White Paper - 3 Approaches to Integrated ALM

KOVAIR

By Kovair Marketing

US Corporate Office

Kovair Software, Inc.
2603 Camino Ramon,
STE 200, San Ramon,
CA 94583, United States
Tel: 1.408.262.0200 Extn.1
Email: sales@kovair.com

India Registered Office

Kovair Software Pvt. Ltd.
PTI Building, 6th Floor, DP-9,
Sector – V, Salt Lake City,
Kolkata - 700091, India
Tel: 91-33-4065 7016/17/18/19
Email: sales@kovair.com

Bangalore Office

Kovair Software Pvt. Ltd.
Samad House, #402, 4th B cross,
7th A Main, HRBR Layout, Kalyan
Nagar, Bangalore- 560043, India
Tel: +91-95350 92589
Email: sales@kovair.com

Table of Contents

Introduction

One of the most desirable aspects of software development process for the last decade has been Integrated Application Lifecycle Management-ALM. Integrated ALM today is also known as End-To-End ALM in some vendor literatures and websites. Fulfilling this promise by various vendors has been patchy at its best. However, in the last 2 years, things are changing as new technologies are maturing in highly critical production systems, there by bringing some of the long overdue benefits of Integrated ALM to real development. Today, if an organization is not thinking about Integrated ALM, it is missing some compelling benefits, which are discussed in a companion Whitepaper from Kovair "10 BENEFITS OF INTEGRATED ALM".

In this paper, we will discuss three widely used approaches provided by different vendors over time towards achieving Integrated ALM and the relative merits of these solutions. We will especially focus on a new integration technology – ESB based ALM Integration for achieving an Integrated ALM for a mixed vendor tools environment.

In software development process, various tools are used both for managing the development process as well as for the actual creation and testing of software code. Following is a list of such tools, some of which are generic (e.g. Document Management) and some are very specific to software development (e.g. Debugger/ Profiler):

Ideation/ Idea Management Tool	**Build Management Tool**
Requirements Elicitation Tool **Requirements Management Tool**	**Configuration Management Tool** **Test Management Tool**
Design and Modeling Tool **IDE (Integrated Development environment)**	**Test Automation Tool** **Debugger/ Profiler**

These tools either generate or manage different objects necessary for managing the ALM processes. The objects include Requirements for Requirements Management, Test cases for Test Management and Source files for Configuration Management. In this document we will refer to these objects as ALM Artifacts or Artifacts.

Since we are still in a nascent stage of Integrated ALM systems and competing vendors are trying to establish the concepts, the best way to define such concepts is to focus on the results of implementing such a system.

An Integrated ALM system provides the following functionalities and benefits:

1. **View artifacts managed by one Tool from another Tool.**
   Examples include a list of Test cases from the Requirements Management Tool and Requirements from the Test case Management Tool, or list of design objects from the IDE. The benefit is even higher when multiple artifacts are accessible from a single tool.
   Each individual stakeholder gets the benefit of accessing the artifacts from other tools without leaving their preferred tool environment. This promotes collaboration and early problem detection, which can minimize errors and save a lot of money.

2. **Create various impacting and non-impacting relationships between any two Artifacts.**
   This can manifest in creating a Traceability relation between a Requirement in Requirements Management Tool and a Test case in the Test Management Tool; so that when the Requirement changes in one tool the Test case in the other tool will be flagged for impact. Another example is creating a dependency relation between a Change Request in an Issues/Change Management tool and the source code in the Configuration Management tool to track what files are modified to implement the Change.
   Without this Traceability, the QA group could be testing an obsolete Requirement without being aware of it. In addition, without this feature, questions like 'why is a single source file affected by so many defects?' cannot be answered.

3. **Automate a process cutting across the tool boundaries and implement a complete ALM lifecycle without a break.**
   For example: a Change which starts its life in a Helpdesk tool as a Ticket, gets automatically replicated as a Change Request in the Issues/ Change Management tool, gets approved by a Change Control Board, goes to the architect to create/ modify architectural artifacts in the Designer/ Modeling tool, goes to the developer for coding and to QA for creating Test cases in IDE and Test Management tools.
   It is imperative to have an automated process workflow drive this across tool boundaries even when individual tools lack the processing capability. Without this process automation, the manual interfaces between tools are managed typically by emails and unstructured documents, inevitably resulting in dropping the ball at times.

4. **Manage Projects and Resources across the tools.**
   For Development managers, it has always been a vexing question to answer 'For a set of Requirements, Changes and Defects what will be the Release date given a set of Resources?' Alternatively, the questions can be posed as 'Given a set of Resources and a Release date, what Requirements, Changes and Defects can we put in this release?' or 'Given a Release date and a set of Requirements, Changes and Defects, what kind of resources do we need?' We can make it even more complex by incorporating design and modeling objects, test automation and other ALM artifacts in the mix. Since the artifacts involved are managed by various tools, only an Integrated ALM system will be able to handle this question.
   Without an Integrated ALM platform, performing Project and Resource Management against just one artifact, say Requirements, will give an incomplete and wrong picture. The traditional PPM (Project Portfolio Management) software tool fails to take care of this because of a lack of integration with all other individual ALM tools.

5. **Create Cross tools Analytics and Dashboards.**
   A Project manager's assessment regarding the health of a particular development project can be massively flawed unless he/she takes into account the activities in all the tools involved. Analytics and reporting across various tools is very important at different levels including that of the CXO.
   For example, an Integrated ALM system should be able to produce a Test compliance report for the end customer that shows the list of Customer Requirements, Change Requests and Issues, traced to Test cases and individual Test runs. And finally, the Test run results should show that they have all passed. To produce a report like this requires retrieving information from various tools including Requirements Management, Issues/ Change Management, Test Management and Test Automation.
   Now that we have discussed the advantages of an Integrated ALM, we will next discuss three widely used options given by vendors for achieving Integrated ALM, and how these options address the above functionality and benefits.

Challenges of Integrated ALM

There are quite a few challenges to introduce and implement Integrated ALM in a development organization successfully. Briefly, they are:

1. **Multi-vendor tools using various technologies.**
   In practice, different tools from multiple vendors are involved in a software development project. Each of these tools could be using different technologies including command line interface, desktop application, client-server, or web based. In addition, they could be running on different platforms such as Windows, Linux of various kernels, Macintosh, UNIX of various flavors or even Mainframe.

2. **The software being produced uses wide range of technologies.**
   The software being produced can be an embedded system, .NET based desktop application, Java based web application, or a COBOL based mainframe application.

3. **The tools use various data repositories.**
   The data repositories can be proprietary file structures, XML, Excel, or relational databases of various flavors.

4. **The tools are geographically distributed.**
   Today's development groups are often distributed globally. Sometimes, the team members can also belong to multiple corporate entities. As a result, the tools reside at various locations behind different firewalls connected by Internet.

Different Approaches for Integrated ALM

To address the challenges of Integrated ALM and to achieve at least some of the goals, different tool vendors have taken different strategies, which can be broadly categorized in the three alternative approaches.

Approach #1: Point-to-point Integrated Multi-Vendor Tools

Approach #2: Single Vendor Integrated ALM Tools

Approach #3: Multi-Vendor Best of Breed Integrated ALM Tools – ALM Integration Platform

The following sections describe the details of these approaches. Any reference to a vendor is based on the available public information from their websites at the time of writing this paper.

Approach #1: Point-to-point Integrated Multi-Vendor Tools

Traditionally the individual vendors have tried to achieve modicum of Integrated ALM by first integrating their own ALM point tools and then integrating some selection of the popular point tools from other vendors. The advantage of such an approach is the simplicity. You need to create only the integration between a particular pair of tools if necessary. However, to achieve the functionality and benefits mentioned above, it is necessary to have point-to-point integration between most, if not every pair of tools. Moreover, since the integration is often done between tools from two different vendors, it often becomes the users' responsibility to create a workable integration using their own development resources. This results in additional internal tool maintenance and management resources, which are not aligned to the core competency of the organization.

The primary issues with the point-to-point approach are discussed below:

Complexity of combinations

From high-school algebra, we know that to have a point-to-point integration between every pair of n tools we need n x (n-1) / 2 number of integrations. For a simple case of 5 tools, this amounts to 10 integrations and just by doubling the number of tools to 10 will result in a 4.5 fold increase in number of integrations to 45. Because of the ad-hoc nature of these integrations, it becomes extremely difficult to create and maintain each individual integration code between the pairs of tools.

Handcrafted Integration Business Rules

Since each integration code is custom made for a particular pair of tools, it has the maximum flexibility but at the same time, maintenance becomes very difficult. For example, the business rules of integration like which field maps from one tool to the other, under which condition the data is replicated from one tool to another, or how a change in one tool will be reflected in another - are all hardcoded in the integration code. This means that any change in that logic necessitates changes in that code. Any change in code means a full cycle of development, test, and deployment cycle making it impossible to implement even the smallest change quickly.

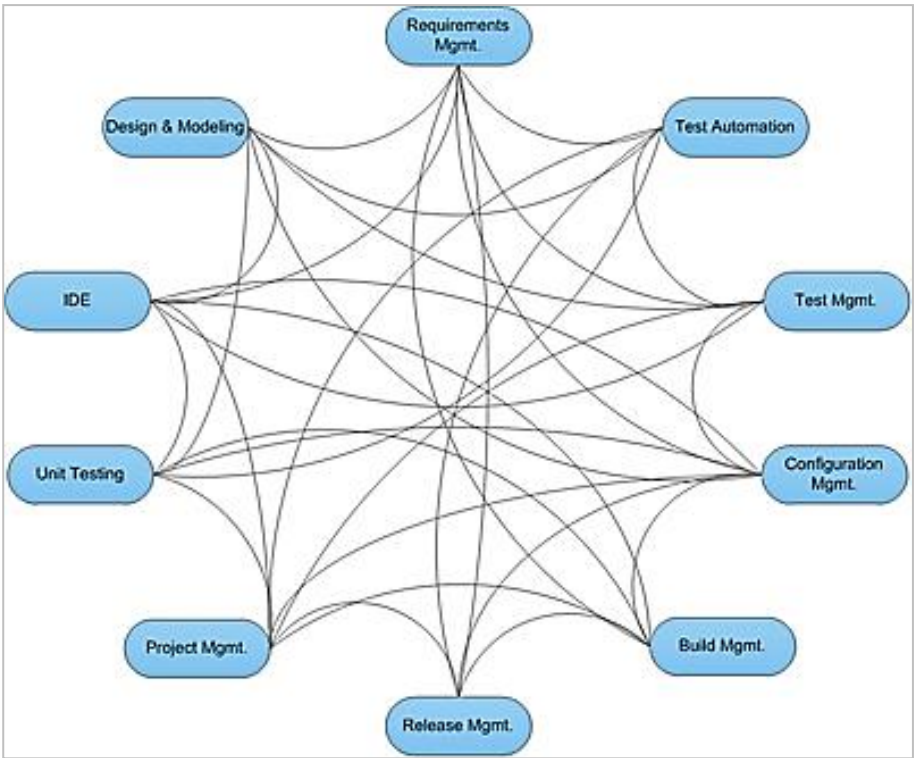

Fig: The Rat's Nest of Point-to-point Integration Solution

Replacement Dilemma

It is unlikely that a development group will use a single tool forever. With changing needs, technology and preferences, existing tools often need to be replaced by more effective alternatives. However, a point-to-point architecture becomes a hindrance to such an upgrade. Referring to our example of 5 and 10 tools scenarios, a single tool replacement will result in re-development of up to 4 and 9 integration codes respectively that often becomes the reason for perpetuation of old and inappropriate tool usage.

These issues are resulting in some development groups abandoning their custom point-to-point integration projects. Even large service companies with enough in-house software expertise and bandwidth are looking for a replacement of their handcrafted internal integration rats-nest architecture with a more sustainable and scalable solution.

Approach #2: Single Vendor Integrated ALM Tools

Because of the problems faced by point-to-point integration architecture, large vendors like IBM and Microsoft have come up with their own solutions to the Integrated ALM. Though their solutions vary in actual implementation, they can be grouped in the category of 'single vendor solutions'. While these solutions also aim at including third party vendor tools, the current implementations on the market do not meet these needs.

IBM's solution consists of the Jazz integration technology and various tools created on Jazz platform. This includes Rational Team Concert, Rational Quality Manager, Rational DOORS, DOORS Next Gen and Rational Requirements Composer. Jazz has an open API, which allows third party vendors to integrate their tools with Jazz. One of the main challenges for Jazz users is its current incompatibility

to an existing tool – both from IBM Rational and third party vendors. In fact IBM/ Rational has created a new Requirements Management and Elicitation tool called Requirement Composer even when they have two well established Requirements Management tools RequisitePro and Telelogic DOORS. Moreover, Jazz requirements make existing tools from other vendors incompatible unless they retrofit with Jazz compliant interfaces using OSLC (Open Services for Lifecycle Collaboration) web services, an IBM proposed open standard. As of now, no other major tool vendor has adopted OSLC standard for their existing or future tools.

On the other hand, Microsoft's solution does not include any stand-alone integration technology. Centerpiece for Microsoft's Integrated ALM solution is the Team Foundation Server (TFS). Like the Rational Team Concert from IBM, TFS includes project management, source control, continuous build and work item management. Comparable to the Eclipse IDE, Microsoft has their Visual Studio Team System along with the tools for modeling, coding and testing. Like IBM, Microsoft proposes that a development team should use all tools from Microsoft to achieve Integrated ALM, though there are scopes for integration of other vendor tools in future. As a result, Microsoft solution requires point-to-point integration for each 3rd party vendor tool with TFS and/ or Visual Studio. This results in similar problems discussed in the point-to-point integration solutions above.

Any development group starting afresh and focused only on Java based development or .NET based development may benefit from using IBM/ Rational and Microsoft solutions respectively. However, for the vast majority of organizations with major investments in existing tools from various vendors and development in Java, .NET and other mixed technologies, neither the IBM nor the Microsoft solutions can be very appealing.

The primary issues with single vendor integrated ALM tools approach are recapped below:

1. **Rip and Replace**
   The requirement of replacing existing tools by a single vendor tools does not sit well with any development manager. Apart from the difficulty of economic justification for such a move, this results in retraining of the development team members in new tools, which may delay development projects unnecessarily
   .
2. **One Size Fits All**
   It is highly unlikely that built-in tools from a single vendor can serve the needs of a wide range of development groups. However, due to the very nature of the solutions, the users are forced to use these tools even when better and often less expensive (sometime free open source) tools are available and appropriate for their needs. For example, Microsoft's Visual Studio Team System designer is a brand new tool without any maturity whereas, there are quite a few matured design tools available, which cannot be used in the Microsoft only environment. Similarly, there are many organizations using Perforce, Subversion, PVCS, CVS who will not be able to participate in the IBM only environment, since they do not use ClearCase or the new configuration management built-in Team Center.

3. **Technology Islands of Development**
   Single vendor tools create a technology island with little or no chance of working together. Groups developing in both .NET and Java simultaneously using Eclipse and Visual Studio will have severe problems adopting a single vendor approach. It will be difficult to meet any of the functionality and benefits mentioned above in scenarios like that. Though there are

patches of integration bridges like integrating TFS to Eclipse by Teamprise (acquired by Microsoft), they are again essentially a point-to-point integration with all its unwanted baggage.

Approach #3: Multi-Vendor Best of Breed Integrated ALM Tools by ALM Middleware

Taking a cue from the integration solutions in other industries, most notably financial software, the concept of ALM middleware addresses all the Integrated ALM requirements squarely and offers some more while avoiding all issues and pitfalls mentioned with the other two approaches of point to point and single vendor integrations.

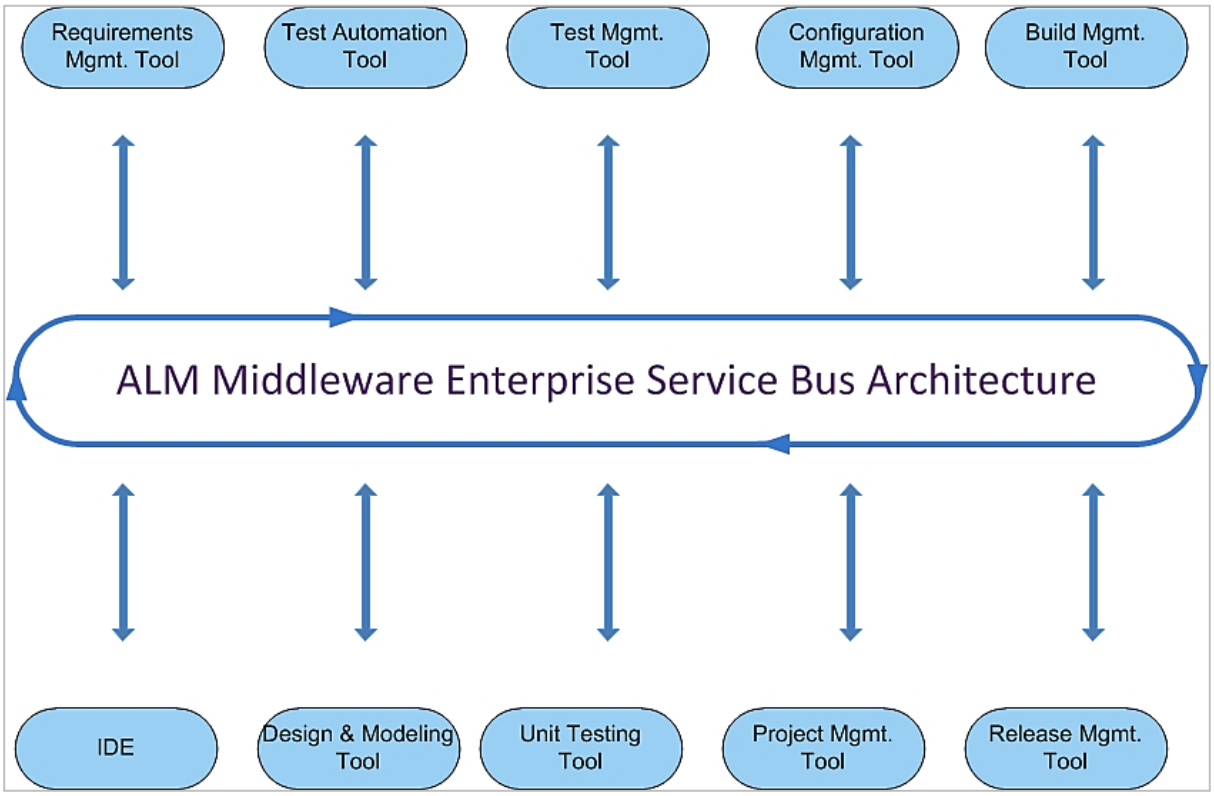

Fig: Enterprise Service based Middleware Integration

The middleware technology is based on **Enterprise Service Bus (ESB)** architecture, which has some distinct advantages over any point-to-point integration architecture. These advantages are discussed below:

1. **Significantly simpler development**
   To use the previous examples of 5 and 10 tools development environment, the ESB architecture needs 5 and 10 adapters respectively, just one per tool! This is substantially less than 10 and 45 custom integrations necessary for point-to-point architecture. Moreover, to replace one of 5 and 10 tools needs replacement of just one adapter, a far cry from the redevelopment of 4 and 9 integration codes respectively.

2. **Protect Investments**
   ALM Middleware is based on a standard set of web service based APIs. Without any special requirement on the tools, ALM Middleware can integrate tools from different vendors, including internally developed tools. This means all the tool investments by a development organization are protected.

3. **Best Tools for Best Functions**
   ALM Middleware allows integration of multiple tools from different vendors for the same function. For example, Requirements Management may be covered by any combination of tools from IBM/ Rational (RequisitePro, DOORS, Rational Requirements Composer), Microfocus/ Borland (Caliber-RM). What is even better, it can support simultaneous usage of multiple tools from multiple vendors in a single tools ecosystem. This allows organizations to select the best tools available in the market without locking themselves in a single vendor solution. The table below illustrates the increased flexibility with tool choices using ALM Middleware.

Tools Choices with Two Integration Approaches

	Single Vendor Integration Approach		**ALM Middleware Approach**
Functions	**IBM Tools**	**Microsoft Tools**	**Multi Vendor Tools**
Requirements Elicitation	Rational Requirements Composer		Blueprint IRise Rational Requirements Composer
Requirements Management	Requisite Pro DOORS	TFS	Caliber-RM DOORS Integrity Kovair MKS Requisite Pro TFS
Design and Modeling	Rational Software Architect/Modeler	VSTS Designer	Enterprise Architect Rational Rose Rational Software Architect /Modeler Visual Studio
IDE (Integrated Development Environment)	RAD	VSTS	Eclipse RAD Visual Studio

Tools Choices with Two Integration Approaches

	Single Vendor Integration Approach		ALM Middleware Approach
Functions	**IBM Tools**	**Microsoft Tools**	**Multi Vendor Tools**
Project Management	Rational Team Concert	Team Foundation Server (TFS)	Kovair MS Project Rational Team Concert Team Foundation Server (TFS)
Build	BuildForge	MS Build	BuildForge Electric Cloud MS Build
Configuration Management	ClearCase	TFS	ClearCase Perforce Subversion TFS Visual SourceSafe
Test Management	Rational Quality Manager	VSTS Tester	Kovair Quality Center Rational Quality Manager Visual Studio
Issues/ Change Management	ClearQuest	TFS	Bugzilla ClearQuest JIRA Kovair Teamtrack TFS
Helpdesk	Tivoli		CA Service Desk Kovair Remedy Tivoli

Note: The tools shown here are for illustration of the capability. At present not all tools mentioned here are integrated in IBM, Microsoft or ALM Middleware solutions.

4. **Flexibility of Integration Business Rules**
   Integration business rules do change over time for various reasons, including changes in business conditions, group dynamics, and development methodologies. For example, how the Requirements will be replicated from IBM RequisitePro to HP Quality Center so that the

Traceability relation created in Quality Center may change over time. Integrated ALM platform allows creation and management of these rules independent of the individual tool adapters. Unlike point-to-point and single-vendor integrations where the logic is hard coded in the integration codes, middleware adapters do not have any embedded business rules. The business rules are managed in middleware administrative module with visual interface for the empowered end users. This eliminates the necessity of development resources for changing the integration codes and reduces the change implementation time drastically from weeks to hours.

5. **Traceability with Change Impact Analysis**
   ALM Middleware makes it simple to create Traceability relation between artifacts from various tools. In case of Point-to-point integration where without a central framework only two tools are integrated at a time, there is no way to visualize and manipulate Traceability relations of three or more Artifacts in a single interface. But for ALM Middleware, a central framework allows flexible ways of creating and managing these relationships among multiple (more than a pair of) tools. Moreover, due to its flexibility in relationship management, ALM Middleware promotes multi-tool proactive and reactive change impact analysis. This is impossible to do both in point-to-point and single vendor integration solutions.

6. **Process Automation without Boundary**
   Typically, Implementing SDLC Processes for multi-tool ALM requires discipline of individual participants. Such a manual implementation of the process fails after a short time when individuals do not follow it due to high overhead and high cost of retraining for each small change. ALM Middleware, with its built-in process automation capability has a unique advantage of creating a cross tools process and automating that for a transparent no-overhead implementation with a much larger success potential. The typical example is a Requirement that starts life in a Requirements Management tool, is reviewed and approved by stakeholders in a Project Management tool, is implemented by a developer in an IDE, and tested by testers in a Test Management tool. ALM Middleware automates the whole process for all these users even when they are using different tools and may be based at different locations.

7. **Analytics and Dashboards**
   Once ALM Middleware has all the artifacts and meta information about the artifacts (e.g. not just the Requirement but the information about the Requestor, Type, Priority, Approval status, Approver, Lifecycle status…) in its repository either by replication (where the artifact information is replicated to ALM Middleware repository) or federation (where the tool data is retrieved on demand, avoiding replication), one can create all sorts of dashboard metrics and reports for those data in real-time. These reports give valuable insight about the whole cross-tool process, which is often impossible or difficult to get.

A Comparison of the 3 Approaches

Item	Point-to-point Integrated Multi Vendor Tools	Single Vendor Integrated ALM Tools	Multi Vendor Best of Breed Integrated ALM Tools –ALM Middleware
View artifacts managed by one Tool from another Tool	**Fair** Only artifacts of the paired tool	**Good** Artifacts in the IDE and other paired tools	**Excellent** All artifacts through the ALM Middleware interface
Create various impacting and non-impacting relationship between any two Artifacts.	**Fair** Only artifacts of the paired tool	**Good** Artifacts in the IDE and other paired tools	**Excellent** All artifacts through the ALM Middleware interface
Automate a process cutting across the tool Boundaries and implementing the complete ALM lifecycle without a break	**Poor**	**Poor**	**Excellent** With built-in process automation in the ALM Middleware interface
Manage Projects and Resources across the tools	**Poor**	**Fair** Through integrated project management tool (if such an integration exists)	**Excellent** With built-in project and resource management in the ALM Middleware interface or external tool
Create Cross tools Analytics and Dashboards	**Fair** Through reporting tool with custom integration	**Fair** Some limited built-in analytics based on individual tools' reporting capability.	**Excellent** With built-in reporting and analytics tool on all artifacts and meta information available in the ALM Middleware interface

A Comparison of the 3 Approaches

Item	Point-to-point Integrated Multi Vendor Tools	Single Vendor Integrated ALM Tools	Multi Vendor Best of Breed Integrated ALM Tools –ALM Middleware
Effort to create and maintain integration with a tool	**Poor** Point to point hand-coded integration code with hardcoded business logic. This involves custom coding, testing and support services. To replace any tool by another tool of same function or even by a new version will involve high effort maintenance of all the integration code for that tool.	**Good** Since existing tools are to be modified for integration in this model, unless the tool vendors are willing to make those changes such tools cannot be integrated by any third party effort. In case of IBM, needs the creation of new tool and adapting existing tool compliant to OSLC standard.	**Excellent** Needs to create an adapter per tool without changing the tool behavior or code. Since tools adapters are created without any changes in the actual tools, the adapters can be created independent of the tool vendors, even by the end user organizations, using an open standard based API.
Utilizing existing tools	**Excellent** Can integrate any existing tool.	**Poor** Integration of existing tools needs modification in existing tools.	**Excellent** Can integrate any existing tool.
Migration and Training Effort	**Excellent** Since existing tools are involved, no migration or training effort is necessary.	**Poor** Since the existing tools are to be replaced by new tools a major data migration and training is required.	**Excellent** Since existing tools are involved, no migration or training effort is necessary.

A Comparison of the 3 Approaches

Item	Point-to-point Integrated Multi Vendor Tools	Single Vendor Integrated ALM Tools	Multi Vendor Best of Breed Integrated ALM Tools –ALM Middleware
Cost	**High** For creating all point-to-point integration code by development resources the cost of development and maintenance is very high.	**High** Since existing tools are to be replaced by new tools the cost is very high both in terms of new license cost as well as cost of deployment, training, documentation and lost opportunity.	**Reasonable** Since no tools are to be replaced and deployment, only the cost of middleware and adapters
Availability	Almost all ALM vendors offer limited point-to-point integration with selected tools. Individual development groups can develop their own custom point-to-point integration as well.	At this time IBM and Microsoft are two vendors offering Integrated ALM using their own tools only. IBM's solution is developed around Jazz technology and Team Center project management tool. Microsoft's solution is around Visual Studio Team System and Team Foundation server tools.	Kovair's Omnibus integration technology is the only ALM Middleware available today based on Enterprise Service Bus (ESB) architecture. All integrations use web service based interface so existing tools can be integrated to Omnibus using adapters independent of their locations and technology platform.

Conclusion

In this paper, we have discussed three approaches to Integrated ALM by comparing them for technology, business values and cost. It has been shown that the method of ESB based ALM integration is the most suitable method for implementing Integrated ALM in organizations having or needing multiple vendor tools.

Kovair has the ALM Middleware technology called Omnibus Enterprise Service Bus, which is the leading integration technology in the ALM industry today. For more information about Kovair's Omnibus technology please visit www.kovair.com or contact sales@kovair.com.

Each tool name used in this paper is the registered trademark of the corresponding tool vendor.

About Kovair

Kovair Software is a Silicon Valley based software product company specializing in the domain of Integrated Application Lifecycle Management (ALM) solutions and supports global software development and management. Kovair's focus on integrating third party best-of-breed ALM tools enables creation of applications in a synchronized tools environment.

Kovair has partnered with leading technology brands like Microsoft, IBM, CA, BMC and more to provide customers a wide range of integration solutions.

Product Portfolio: Kovair's flagship products Omnibus Integration Platform, ALM Studio, QuickSync and Integrated DevOps are highly preferred solutions by some of the major corporations globally.

Recognitions: The SD Times 100 has recognized Kovair as one of the top 100 software innovators in the domain of Application Lifecycle Management. Kovair's Innovations in ALM Tools and ALM Integrations are well recognized both in the industry and by analysts at places like Gartner and Forrester.

Business Focus: Application Lifecycle Management Products and Services, Integration Platform

Industry Verticals: IT Consulting and Services, Banking and Financial Services, Telecom, Manufacturing, Networking, Healthcare, Defense and Government.

Contact: For more information about product and services contact sales@kovair.com. You may follow Kovair updates on Facebook, LinkedIn, Twitter, Google+, Slideshare and YouTube.

Important Links: Why Kovair| Management | Product Updates | Tool Integrations | Product Brochure | Videos | Datasheets | White Papers | Case Study | Technical Documents| Presentations | Services | Blog | Press Releases | Events | Customers | Partners | Support | Contact | Site Map

Global Technology Partners

Microsoft Visual Studio Industry Partner Program

Ready for IBM Rational software

bmc TAP Premier Partner

ca technologies

RALLY SOFTWARE

Cognizant

Capgemini CONSULTING.TECHNOLOGY.OUTSOURCING

Tech Mahindra

salesforce

Memberships and Associations

NASSCOM®

ESC Electronics and Computer Software Export Promotion Council

एसटीपीआई STPI

SIATI