KOVAIR

Scalable Software Delivery

Through

Integrated Agile ALM

With Elements of DevOps

Rajsekhar Bhattacharjee
VP – Product Deliveries

Kovair Software, Inc.
4699 Old Ironsides Drive,
Unit 190,
Santa Clara, CA 95054
Sales: 1.408.262.0200 Extn. 1
sales@kovair.com
www.kovair.com

Document Version History		
Release	**Date**	**Reason**
Version 1.0 (Initial Release)	02/04/2013	

Table of Contents

Introduction

The unprecedented level of success of Agile methods is causing a wide range of organizations to apply Agile techniques on large project teams with hundreds of people, often geographically distributed, in various environments and various complex situations that involve regulatory compliance, external partners, etc. Agile is getting pushed beyond the small, co-located team environments that it has been typically associated with. Clearly, Agile needs to scale far larger than the techniques were pioneered for.

While there is an unmistakable shift towards Agile as a software development methodology of choice, not all of those who get started meet with success. Not surprisingly, this has led to an industry-wide quest for a better understanding of the success factors in the adoption of Agile.

Visibly improved quality and productivity are the two key reasons behind the acceptance of Agile. The Agile principles have led to the adoption of practices that require software delivery organizations to fundamentally change the way they work to improve team synergy and engage with customers during the development process to deliver working software frequently. Improvement of quality is a natural outcome of the need to deliver working software to the customer frequently while improvement of productivity is a natural outcome of group synergy.

This improvement of quality and productivity was first noticed in primarily software construction projects by small, co-located teams. During that period, tools other than development tools (IDE, Test Automation Tools, and SCM Tools) were not considered important or necessary for successful delivery of software. The first Agile principle of 'Individuals and interactions over processes and tools' was quoted by many in arguing that use of tools was against the principles of Agile because tools reduce agility. The industry-wide interest in Agile changed this perception about tools.

What followed was an industry-wide agreement that ideas of Agile had to be extended from partial methods to a full-fledged, disciplined delivery processes, and had to scale using next-generation ALM tools to work effectively for geographically distributed large teams involving hundreds of people engaged in a full delivery process - from project initiation to deployment and maintenance. Thus, the idea of "Agile ALM" was born.

However, Agile principles of team interaction, collaboration, frequent delivery and responsiveness to change all through the project required the Agile Project Management tools, the Lifecycle Management tools (ALM tools), and the Development tools to seamlessly integrate allowing in-context collaboration among all members of the team. The need for frequent delivery of working software to customer further required elements of DevOps to be integrated into the lifecycle. Hence, the idea of Agile ALM had to be extended further to 'Integrated Agile ALM'.

This whitepaper discusses some key considerations for the successful implementation of a scalable IT delivery process using an 'Integrated Agile ALM' framework. It also discusses the widely used approaches taken by different vendors over time towards achieving Integrated Agile ALM, and the relative merits of these approaches.

Defining an Integrated Agile ALM Framework

Using Scrum as the base, this section proposes a model for a scalable end-to-end Agile delivery framework that can be used to manage delivery of large scale software development projects from inception to delivery into production and maintenance. It shows how Scrum can be extended to scale; how DevOps is a natural extension of the Agile principle of frequent delivery of working software to the customer; and why and how ALM tools need to be used to establish an end-to-end integrated tool chain covering all lifecycle phases in a continuum.

Scaling Agile

The Agile manifesto, as well as the mainstream Agile methods, has an implicit development focus. The principles of customer collaboration, frequent delivery of working software, responsiveness to change, etc. ensure that what's built is what the customer needs. This is great for the customer, but for the delivery organization, how it is built is also important. The delivery organization is faced with the dual challenges of building it right and building it the right way using the right tools.

A delivery process that is not efficient, streamlined, or cost effective impacts the bottomline of the delivery organization. Allowing the team to choose the tools they want to use is great as long as it does not involve significant investment in new software tools that the project budget does not cover. For any project, there is always a pressure to leverage existing tools and infrastructure. So, adopting radically different tools, infrastructure, or processes may not always be feasible.

Responsiveness to change is based on sound logic, but it also impacts cost and schedule. Contractual obligations, requirement for upfront planning and budgeting, constraints on cost and schedule are some of the several scaling challenges that delivery organizations adopting Agile face. The other Agile scaling factors are large, geographically distributed teams, mandated documentation and compliance standards, organizational complexity.

The delivery organizations of today are looking at all options of leveraging the benefits of Agile within a delivery framework that allows large, geographically distributed teams to deliver consistently better quality, faster and cheaper while still operating under the contractual constraints of schedule, cost, externally mandated requirements, dependencies on external partners, etc., and organizational constraints of various kinds including but not limited to organization culture, IT infrastructure, enterprise IT disciplines, etc.

Clearly, delivery organization needs to scale agile

Scaling Approach

Scott Ambler and many other contemporary thought leaders and analysts have variously prescribed a multi-dimensional approach to scaling Agile, involving:

- Moving from simplistic and empirical processes to full-scale industrialized agile delivery processes
- Customizing the delivery processes to take into account various scaling challenges, such as integrating DevOps into the lifecycle
- Using the right tools where necessary

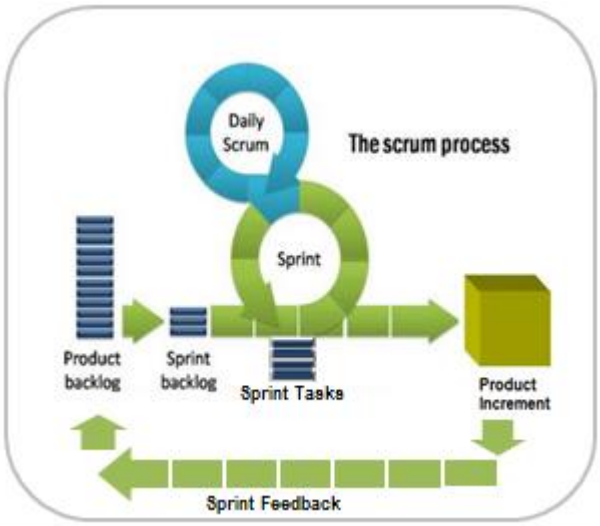
Taking note of the fact that many large-scale agile adoptions would not function without tools, there is a universal consensus on the use of tools where necessary. For example, paper-based face-to-face strategies that work well for small teams must be replaced with techniques and tools appropriate for medium size teams as well as for very large teams. Similarly, collaboration strategies that worked for small & co-located teams must give way to sophisticated collaboration tools and techniques when teams

are geographically distributed. The meaning of collaboration has to grow beyond email and instant messaging; the tools must make it possible to seamlessly collaborate in context.

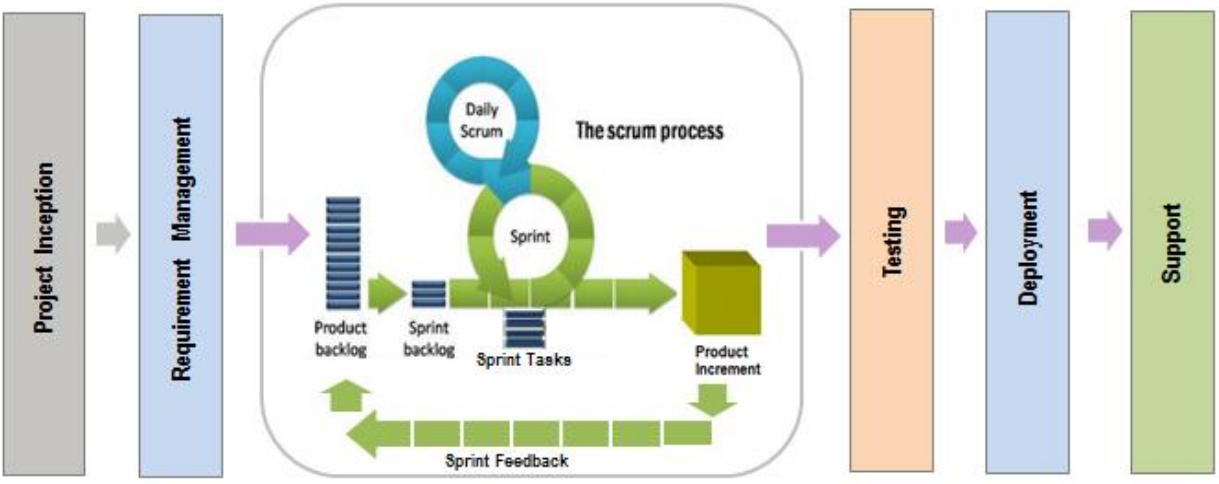
An Agile Delivery Process

The mainstream agile methods are focused on different aspects of the software development life cycle with an implicit development focus. Some focus on the practices (Extreme Programming, Pragmatic Programming, Agile Modeling), while others focus on managing the software projects (the Scrum approach). Yet, there are approaches providing full coverage over the development life cycle, such as Dynamic Systems Development Method (DSDM) and IBM Rational Unified Process (RUP). Thus, there is a clear difference between the various agile software development methods in this regard. Whereas DSDM and RUP do not need complementing approaches to support software development, the others do to a varying degree.
(Wikipedia)

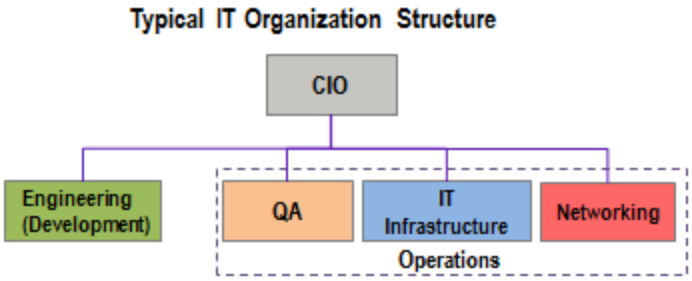
An Agile Delivery Process with Scrum



With its product-centric focus, Scrum provides an excellent foundation on which scalable delivery processes can be built. It is a framework within which various processes and techniques can be employed. While Scrum itself focuses on requirements management and incremental product delivery, it does not specify how the Development team works; how the final product is deployed in production; or how it is maintained. However, Scrum's proven incremental and adaptive development approach can form the nucleus of a project-centric end-to-end delivery process -- from project inception that includes scoping project, creating an architectural framework and a technical strategy, making an estimate of cost and schedule to determine feasibility, etc., to development, which is based on Scrum's principles of managing Product Backlogs, Sprint Planning, Sprint Reviews, etc., and is powered by tools that facilitate and integrate requirements management, architecture, coding, testing, tracking, and release management, to deployment in production that integrates elements of DevOps, to maintenance -- with project tracking and monitoring across the entire lifecycle.

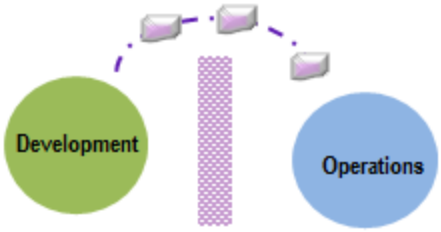


Integrating DevOps into the Lifecycle



In a traditional organization structure with independent departments for development, QA, and IT operations, development team hands over the developed software product to the QA team for testing, and the QA team hands over the tested software product to IT Operations team for deployment in production.

Quite often, the activities in the organization's development and operations processes that affect transitions involve dealing with different priorities, goals, processes, tools, and a variety of other issues. Improving the hand-off of work from development to operations is an age-old goal.



DevOps is an umbrella concept that refers to anything that streamlines communication and collaboration between development and operations. The goal of DevOps is to break down these silos to achieve a better delivery lifecycle.

DevOps and Agile

In many ways, focus on DevOps is a natural consequence of the Agile movement.

Continuous Integration

The Agile principle of frequent delivery of working software to the customer requires the Agile team to build and test the software as frequently as possible. The Agile practice of Continuous Integration, abbreviated as CI, is popular because it addresses this critical aspect of the success of Agile very well. CI makes Integration a non-event! Single source-code repository, daily commit, automated unit testing, automated build, automated functional testing, regression testing are some of the key practices within the practice of CI.

Continuous Deployment

The point of continuous integration is to flush out the problems in the system in rapid cycles. In the end, the system will be judged by how it runs in production. Therefore, continuous integration cannot deliver working software unless the software is continuously deployed and tested in a production environment.

A significant part of how a system behaves in production is linked with the production environment. Technical issues related to hardware, system software versions, networking software, database systems, etc. can arise when software is moved to production system. Validation of non-functional requirements, such as security, performance, and availability must be performed in the target production environment only.

Traditionally, the focus of testing used to be on the functional side of the testing during development, and non-functional aspects of testing, which are also critical, were deferred until after the development phase. To mitigate this risk, the test environment must be as exact a replica of the production environment as possible.

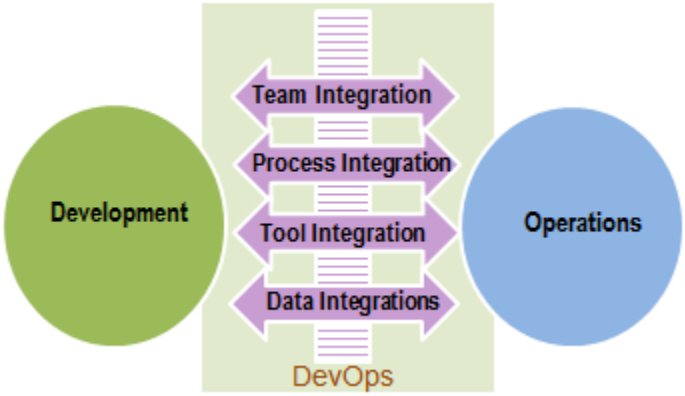
Continuous Deployment is the essence of DevOps. DevOps makes deployment a non-event!

Realizing DevOps

Realizing continuous deployment, in terms of teams involved, would require the development team to hand-off the compiled and unit tested product to the operations team on a daily basis. Now, this is not practically possible as long as the software needs to be 'tossed over the wall' of organizational barriers. Implementing continuous deployment would require:

Team Integration: Members of the operations teams should be part of the 'self-organizing and cross-functional Agile team'

Process Integration: The activities that affect transitions between teams must reflect the realities of the processes of all teams, such as Development, IT Operations, and QA. Examples include both the deployment of software into test, staging, or production and the resolution of problems caused by defects in the software, which must be fixed and a new version of the software must be redeployed.



Tool Integration: Tool integrations allow one tool to leverage the interface provided by the other to automate tasks that affect transitions between teams. For example, to enable the release of a solution into production, a deployment tool could access production configuration information maintained in a configuration management tool. To enable reporting of a problem with an application, a test management tool could register the defect reports into the defect management tool used by the development team.

Data Integration: Data integration allows multiple tools to work with one another's data. For instance, on the development side, a released product may include quality results, configuration information, and proof of regulatory compliance. In operations, usage statistics are tracked and configuration and installation details are maintained for the solution. Integration of this information enables consistency and accuracy between the development and operations teams through single truth across teams.

Scott W. Ambler in his article '2012: The Year of DevOps' has identified these three dimensions as critical to the success of DevOps: Process Integration, Tool Integration and Data Integration.

Application Lifecycle Management Tools

Application Lifecycle Management (ALM) refers to the capability to integrate, coordinate and manage the different phases of the software delivery process — from development to deployment. ALM is a set of pre-defined processes and tools that includes definition, design, development, testing, deployment and management. Throughout the ALM process, each of these phases are closely monitored and controlled.

Categories of ALM Tools

The categories of ALM tools include, but are not limited to:

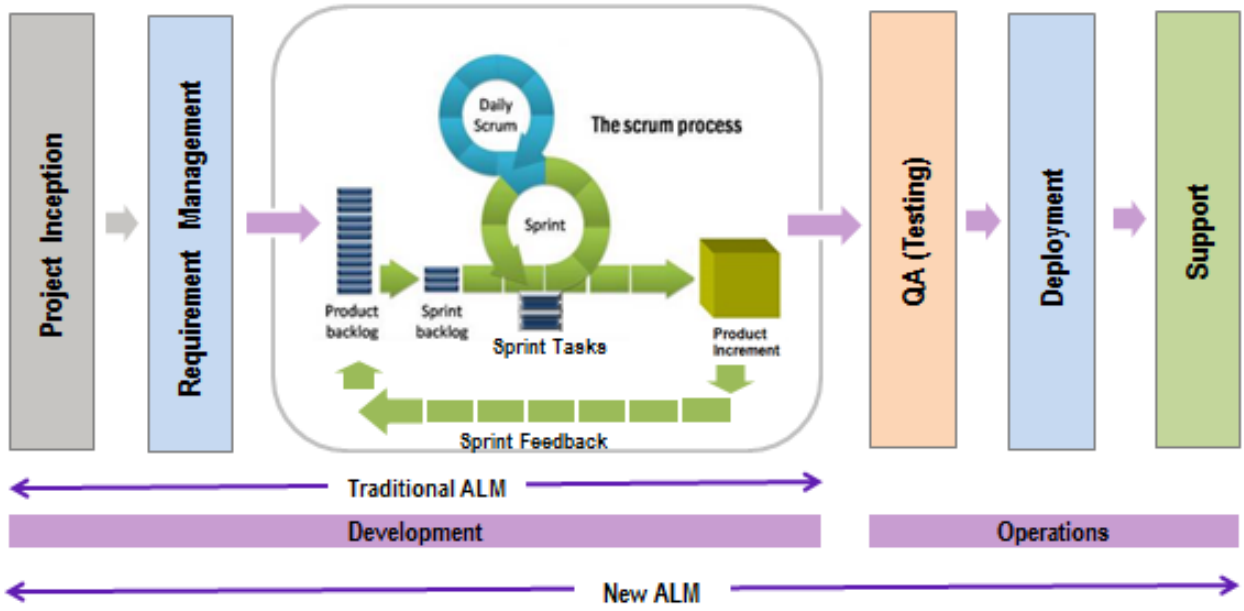
- Requirements management
- Modeling
- Design
- Integrated Development Environment (IDE)
- Software configuration management
- Change management

- Build management
- Automated unit testing
- Test automation (Automated functional testing)
- Test management
- Software deployment
- Issue/Defect management
- Project management

ALM being one of the fastest growing areas in the IT space in recent times, more and more ALM tools are continuously appearing in each category. With Agile becoming mainstream, new approaches focused on mixing and matching the right methodologies will emerge, and will need to be supported by ALM tool vendors. Demand for Cloud deployment will drive the growth of SaaS-based ALM tools. With large-scale virtualization of server infrastructure becoming a reality, organizations will move more and more of their IT infrastructure to virtualized environments with increased focus on further reduction of spending on hardware. Thus, ALM has entered a new phase in its evolution as it has to deal with the impact of agile methodologies on software development on one hand, and the impact of DevOps on software deployment and support on the other.

DevOps in ALM

In many ways, DevOps is a natural extension of ALM. Large scale delivery of software projects needs processes and tools for managing full delivery lifecycle from project initiation to deployment into production and maintenance.



Traditionally ALM tools were focused on phases from inception till development cut-over. This included tools for Requirement Management, Modeling, Design, Development (IDE), SCM, Build Management, Automated Unit Testing, Automated Functional Testing, etc., and Project Management. Tools necessary for managing application lifecycle beyond development, i.e., QA, Deployment, and Support were largely part of IT Operations and Services Management tool set. ALM has since expanded to include, in a seamlessly integrated way, tools for Deployment Automation, IT Services Management, Quality Metrics, etc.

Integrated Agile ALM

ALM is universal and methodology agnostic. ALM applies to Agile as much as it does to any other methodology, e.g., Waterfall. So, what's needed for 'Agile ALM'? In reality, it is about the tools, but with a difference. Agile ALM refers to the ability of using tools in a way that people want in an agile development project, and not the other way around.

Scaling agile needs a tool chain across the life cycle that supports agile way of planning, collaborating, managing task based development, continuous integration, frequent releases, ability to create and maintain traceable artifacts across the lifecycle, and so on. Tools can no longer enforce how team should work; tools that do not enhance productivity and agility, while allowing people to do their work the way they want, are no longer acceptable. Thus, Agile ALM is implicitly 'Integrated Agile ALM'.

In Agile, the more seamless an integrated product development framework is, the more an agile team can focus on building customer value. The integration becomes an enabler for delivering customer value. At the same time, there is a need for flexibility and customization so that an ALM tool framework doesn't drive the team's interaction.

Summary

A working definition of an Integrated Agile ALM is thus proposed as:

- It is an end-to-end process framework that can be used to manage the delivery of large scale software development projects from inception to delivery, into production and maintenance, without compromising on the core Agile principles.
- It is an extension of Scrum.
- It has elements of DevOps in it.
- It can be implemented using an integrated ALM tool chain across the life cycle that support agile way of:
    - Planning
    - Collaborating
    - Managing task based development
    - Continuous integration
    - Frequent releases
    - Ability to create and maintain traceable artifacts across the lifecycle
- It is generic enough to be customized in a way that Agile team wants to use the tools.

About Kovair

Based out of California, Kovair is an innovation leader in ALM and ITSM solutions supporting global software development and management. Kovair's Global Lifecycle Management Platform and the Omnibus Integration Platform offer customers a wide range of choices to implement robust, scalable, and collaborative IT Delivery and Management systems.

Customers can use Kovair's fully configurable and extensible Global Lifecycle Platform or one of its three available out-of-the-box solutions, 'ALM Studio', 'ITSM Studio', and 'Kovair Agile' to implement their process. Customers can also use Kovair's Global Lifecycle Platform or one of its three available out-of-the-box solutions and Kovair's Omnibus Integration Platform to integrate best-of-breed ALM tools to create a scalable, Integrated Agile ALM solution. Alternatively, customers can use just the Omnibus Integration Platform to integrate best-of-breed ALM tools to create their custom delivery processes. Omnibus offers integrations with over 35 industry leading ALM tools that include tools from IBM, HP, Microsoft, and other commercial vendors as well as open source tools.

Resources

Agile software development, Wikipedia, http://en.wikipedia.org/wiki/Agile_software_development
Manifesto for Agile Software Development, http://agilemanifesto.org/
Scott Ambler (October 4, 2011): "Agility At Scale"
(http://www.projectsatwork.com/article.cfm?ID=267575&authenticated=1)
The Scrum Guide, Developed and sustained by Ken Schwaber and Jeff Sutherland
http://www.scrum.org/Portals/0/Documents/Scrum%20Guides/Scrum_Guide.pdf
Continuous Integration, Martin Fowler, http://martinfowler.com/articles/continuousIntegration.html
DevOps, Wikipedia. http://en.wikipedia.org/wiki/DevOps
What is DevOps, Damon Edwards, http://dev2ops.org/2010/02/what-is-devops/
2012: The Year of DevOps, Scott W. Ambler,
http://www.cmcrossroads.com/article/2012-year-devops?page=0%2C0