Application Lifecycle Management: Marriage of Business Management with Software Engineering

Lovelesh Chawla, Robert F. Roggio
School of Computing
University of North Florida
Jacksonville, FL
Lovelesh.chawla@gmail.com
broggio@unf.edu

Abstract—**This paper discusses Application Lifecycle Management (ALM), its evolution, and its characteristics used to manage the lifecycle of an application through Governance, Development and Maintenance. The paper addresses IBM's tool for Collaborative Lifecycle management (CLM) and its relationship to ALM. Lastly, we cover key tools of CLM: Rational Team Concert (RTC), Rational Requirements Composer (RRC), and Rational Quality Manager (RQM).**

Keywords – CLM; lifecycle; management; rational; application

I. Evolution of ALM

"Application lifecycle management (ALM) is the marriage of business management to software engineering made possible by tools that facilitate and integrate requirements management, architecture, coding, testing, tracking, and release management" according to Wikipedia. [1]

The origins of ALM stem back to many well-known pioneers and related technology originating arguably in the late 70s that include luminaries such as Ed Yourdon, Tom DeMarco, Ken Orr, Constantine, Dijkstra and more, who in various ways discussed methods to build software applications that were functionally complete and correct or--as we say it today. [2]

Some focused on moving structured programming concepts into structured design models. These believed maintaining the design that is properly structured will result in program that will automatically be structured, regardless of the language used.

On the other hand others felt that a structured design did not guarantee that the application produced would meet the needs of the organization or users. Still others argued as to the approach. Should designs be process driven or data driven? Should the designs be programming language independent? Should the focus be on developing tools that generate programs based on design specifications? Should the design be driven from output requirements? The list went on and on. All of these left their mark on the way systems are conceived, designed and developed [2].

A variety of methodologies arose include the Warnier-Orr methodology, which was built around identifying output requirements and the decomposition of those into data and program structures needed to identify the inputs and information required. Ed Yourdan concentrated on a more process-oriented approach through the use of Data Flow Diagrams (DFDs), which remained very popular for many years in capturing data flow.

Software Engineering (CASE) tools arose out of the various structured methodologies and philosophies with arguable ability to support traceability. Perhaps the lone survivor is the Uniform Modeling Language (UML), which has become a widely accepted modeling notation.

The evolution from structured programming to structured design to application lifecycle management is to synchronize technology solutions organizational and stakeholder needs. These need to be tightly woven into business process improvement methods and frameworks [2].

II. Business Management Aspects of ALM

An application's lifecycle may be viewed as beginning with an idea, which is ultimately realized in an application. Once developed, deployment and application maintenance occur over the remainder of the application's lifecycle until the application no longer provides significant business value. The software is then typically either redesigned, replaced or retired as illustrated in Figures 1 and 2.

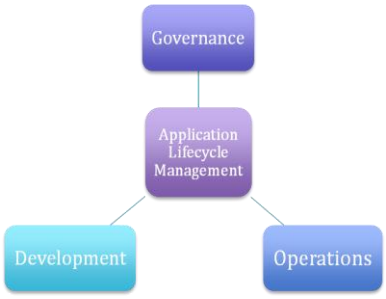


Figure 1. Three Areas of ALM

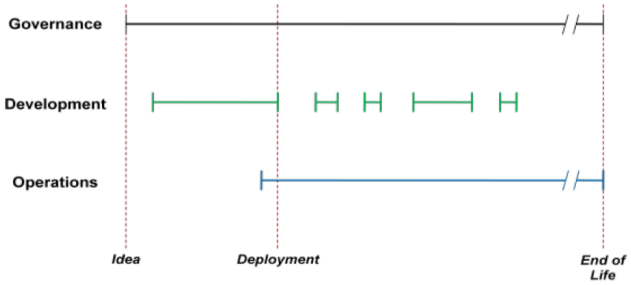


Figure 2. A Closer View of ALM [1]

A. *Governance*

Governance encompasses all of the decision-making and project management considerations for an application and extends over the entire life cycle. Development is essentially the process of creating an application, while Operations is the effort required to run and maintain the application.

Because an application is an enterprise asset, the organization needs to maintain an ongoing understanding of its benefits and costs. Meeting this need, Application Portfolio Management (APM) provides a mechanism to avoid duplicating functionality among different enterprise applications. While thus providing application governance, APM addresses activities such as updates and significant revisions that make business sense. APM also includes business case development and project portfolio management for each of the application revision, as can be observed in the Development line in Figure 3. The only ALM activity extending throughout the entire ALM time span is governance, which may be considered the most important aspect of ALM.

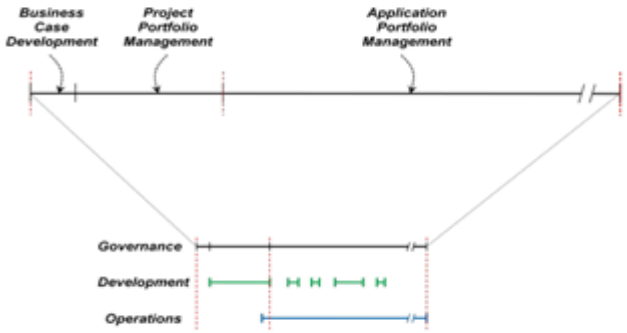


Figure 3 Governance Extends over Entire Application Lifecycle [1]

B. *Development*

While mapping ALM to the software development process isn't entirely warranted, development certainly is a fundamental part of every application's lifecycle (Fig. 4).

Once the business case is approved, and software development starts, development is realized through a series of iterations, where each iteration contains some requirements definition, some design, some development, and some testing and deployment. (The exact activities depend upon the development methodology employed)

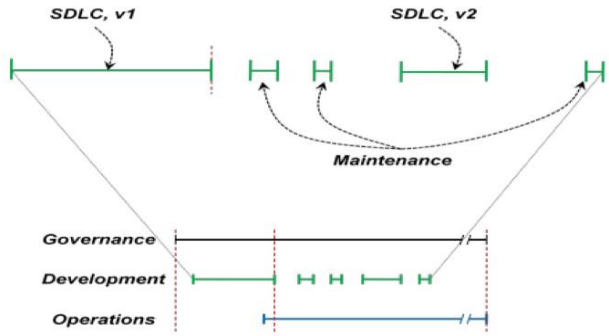


Figure 4 Development within ALM [1]

Once the development process of the application is complete and the application is fully deployed, development still continues via periodic enhancements and bugs fix (adaptive, perfective, and remedial maintenance).

C. *Operations*

Every deployed application must be monitored and managed. The operations line is tightly connected to Development activity (Fig. 5) and is monitored, maintained, and updated throughout its useful lifetime. Similarly, each update to the application must be deployed once it's completed, as the figure indicates.

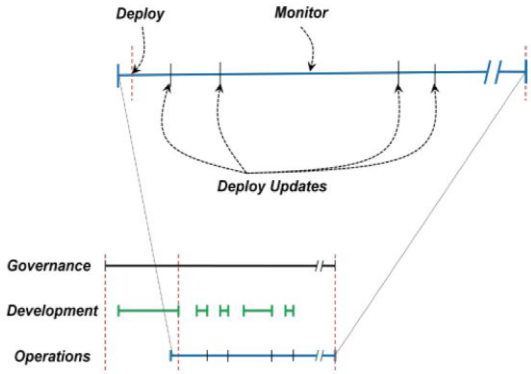


Figure 5 Operations begin shortly before application deployed; continues until application removed from service. [1]

III. CLM: Software Engineering Aspects of ALM

A. *Lifecycle Management*

Collaborative Lifecycle Management (CLM) is an integrated Application Lifecycle Management solution by IBM. It provides integrations across Jazz™-based products to connect the work of analysts with developers and testers. "Jazz™ is built on architectural principles that represent a key departure from approaches taken in the past. Together, these approaches allow teams to "surf the collaborative Web" to seamlessly access teams, processes and artifacts" [5] CLM links among the products support traceability, web-like

navigation, review, commenting, and status tracking across various project repositories." [4] CLM integrations build on the Jazz™ Foundation to provide a common approach to artifact linking, dashboards, security, and user interface frameworks. [7]

B. *Jazz™ Platform*

First and foremost, it is important to recognize that ™ is an integrated development platform emphasizing team collaboration, communication, and software development processes. [5]

Built on Eclipse technology and focused on integrating tasks across the software lifecycle to improve the productivity of an entire team, the Jazz™ platform consists of an architecture and a set of application frameworks and toolkits, as shown in the Fig. 6 [5].

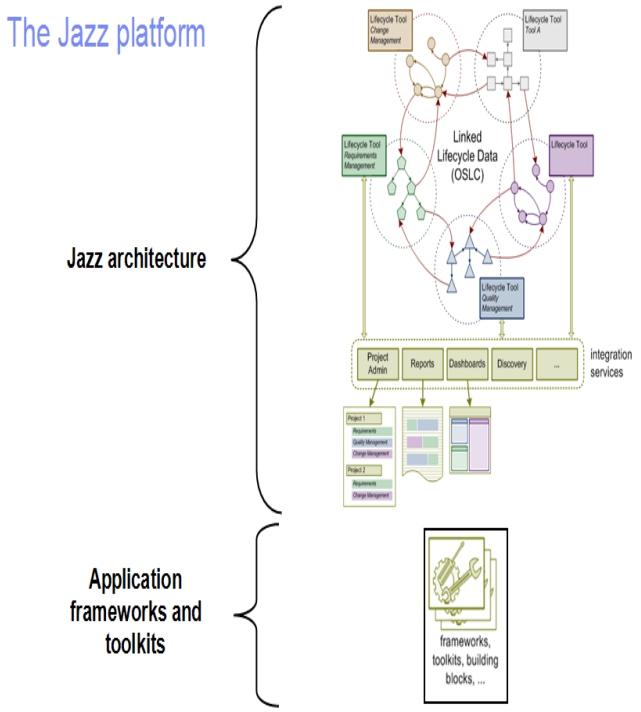


Figure 6. Jazz™ Platform [5]

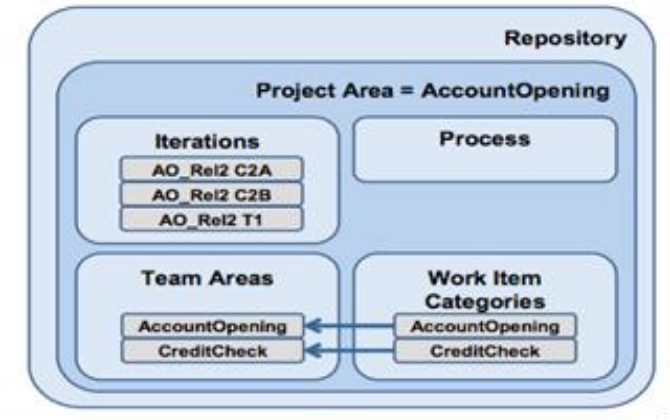
The Jazz™ platform also supports simple artifact exchange such as business requirements, use cases and collaboration between developers by using higher value assets, such as complete change sets (that is, a number of modifications that are relevant to each other in one atomic package" [8]), as compared to the previous more time consuming exchange of source code files or mail with code snippets. Jazz™ also supports instant messaging to enable everyone to be informed of events appropriate to a team or an individual. Conclusions drawn from actual work items are used to assess overall project health [5].

C. *Project and Team Areas*

The Jazz™ platform provides basic concepts to create and manage projects as illustrated in Figure 7 in Rational Team Concert (RTC). RTC is a team collaboration tool by IBM that allows a team to manage all aspects of their work such as plans, tasks, builds and reports.

Let's look at the following terms from Figure 7 used to describe components within a Project Repository. The Project Area provides definition for the project deliverables such as team structure, process, and schedule. Team Areas refers to the management of team membership, roles assignments, and team artifacts. Work Item Categories, categorize group work items by the various components or functional areas of the project and are associated with a team area. A work item is a mechanism used to keep track of tasks and issues a team needs to address during development. One may view the status and number of work items as an indicator of project health.

A CreditCheck Team Area that brings development team together on collaborative development in Rational Team Concert is shown in Figure 8. Some key Team Area properties are highlighted in the figure. For example, the Team Artifacts and Team Central views provide transparency to team health, the team members and their assigned roles, and the iteration structure with the current iteration plan that is selected. [4]



. Figure 7 Project and Team Areas in Rational Team Concert [4]

More terminology is used for agile planning in Rational Team Concert (Fig. 8):

Development Line

A development line represents an area of activity within a project that typically has its own objectives, deliverables, team, process, and schedule. In a bit more detail, an ongoing project that has both current maintenance underway but is also working on new release development might choose to define the work efforts in two separate development lines because they have different values for objectives, schedule, processes and more. Further, project intervals or phases defined within a development line may be associated with a hierarchy of iterations [9]

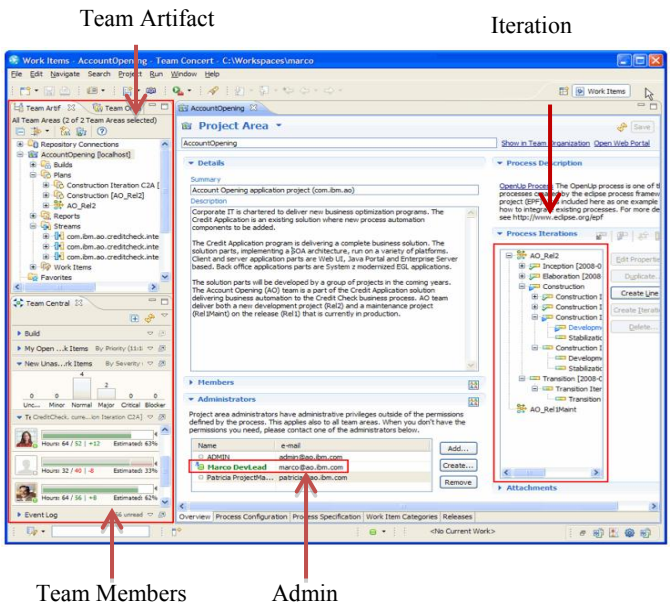


Figure 8. The CreditCheck Area in Rational Team Concert [4]

Iteration

Iterations are defined within a development line and are used to express the details of an iteration in terms of work items. Projects are organized into a series of development periods referred to as iterations.

Work item:

- A work item is a representation of the tasks and issues that a team needs to address during the development cycle. Work items are essential for iteration planning and indicators of the health of the project. [4] Work items are activities the team must do.

D. Rational Team Concert

In looking at the prime components of Rational Team Concert, we see the three main components: Rational Requirements Composer, Rational Team Concert, and Rational Quality Manager. Each contributes to an overall transparent environment in support of the Rational solution for Collaborative Lifecycle Management built on Jazz™ to help teams integrate tasks across the software life cycle. [6]

According to [10], CLM is an integrated Application Lifecycle Management solution comprising three main products: IBM Rational Requirements Composer, IBM Rational Team Concert, and IBM Rational Quality Manager.

Rational Requirement Composer (RRC)

Rational Requirement Composer (Figure 9) is a tool used to define, collaborate and manage requirements for any size development effort or project team. Project teams can manage their requirements, write good use cases, improve traceability, strengthen collaboration, reduce project risk, and increase quality. RRC also provides improvement in the following key areas:" [4]

- Improved requirements definition, validation, and management of requirements change through the software development life cycle
- Increased and clearer communication among business stakeholders and IT delivery teams wherever they are located, and
- Less project rework, faster project execution, and lower-cost delivery

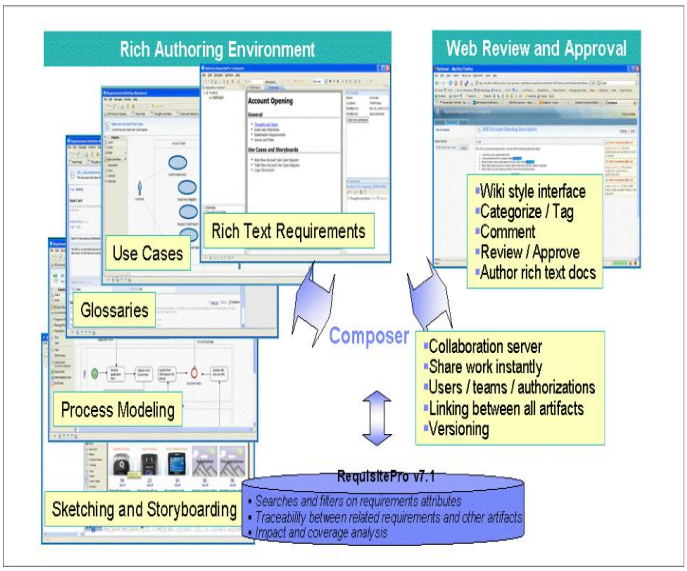


Figure 9. Rational Requirements Composer [4]

Team members require a number of techniques in order to gather requirements information. RRC helps in these activities by providing features to capture drawing, storyboard, use cases, glossaries and other models critical to effective requirements capture. These features of RRC are significant helps to analysts for requirements capture.

Rational Team Concert (Fig. 10) is a Rational product on the Jazz™ team collaboration platform, first released on 2008. [11] Available in both client versions and a web version, RTC provides the following key capabilities to support collaborative development:

- Integrates seamlessly the development task across the delivery life cycle for the team
- Facilitates team collaboration and coordination and helps the team develop applications more effectively and with less risk
- Supports team collaboration across co-located and globally distributed teams
- Establishes and maintains traceability and audit trails, and automates bookkeeping so that teams are accountable

- Integrates into Eclipse for developers and provides Web access for external contributors
- Makes collaborative development more enjoyable [4]

Rational Team Concert supports and provides for seamless integration of workflows, for the application life-cycle assets managed by the Jazz™ repository and for assets managed by other repositories not part of the Jazz™ platform. [11]

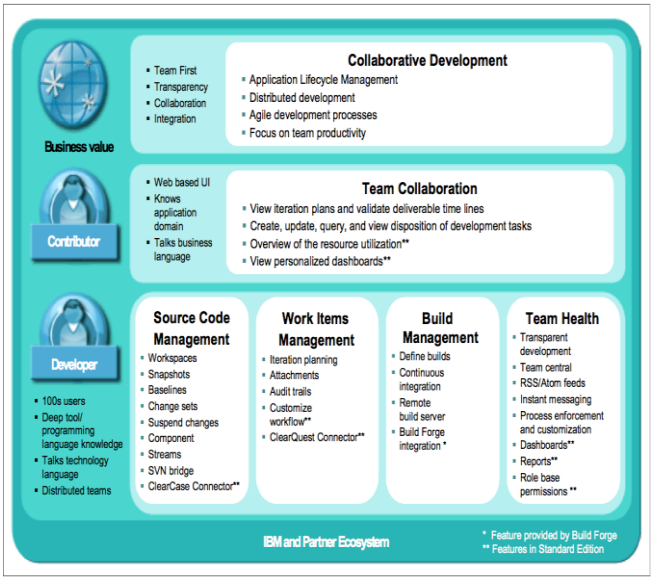


Figure 10. Rational Team Concert [4]

It is important to note which ALM domains are supported by RTC. These are shown in Figure 10.

- Source code management
- Work items management, including agile iteration planning
- Build management
- Team health and collaboration

Rational Quality Manager (RQM)

"Rational Quality Manager (Fig. 8) is one of the three main Rational products on the Jazz™ team collaboration platform. Rational Quality Manager provides the development efforts a test management system rendering the testing in total control of testing efforts. Test planning, construction, and execution can be completed in Rational Quality Manager. Decisions are managed in a database that enables the team to track against the plan." [4]

One of the most critical questions the test team needs to answer is: "When are we done testing?" Teams are better equipped to answer that question by placing an emphasis on the test plan, which enables a team to track their progress against the plan with Rational Quality Manager. A team can do as much or as little as they need.

The Rational Quality Manager product provides the following features:

- Collaborative Web-based quality management solution
- A central repository for test planning, construction, deployment, and execution
- The ability to align the test effort with project requirements
- Quantifiable metrics for project tracking and decision making
- Keyword-driven manual test authoring and execution [4]

Rational Quality Manager can work for both simple testing and very comprehensive testing to the extent that testing can scale to large teams sharing test assets. Most of the work items managed by the system can be assigned to a team member. Unlike some other testing environments, in RQM a test plan can easily be decomposed into different components and delegated to different stakeholders. All activities can be assigned and tracked in order to assess the level of effort in building and executing tests. .

This assignment of work in all aspects of the test effort helps the team to ensure that all expected work is completed. It also gives them insight into their progress against the work effort.

IV. SUMMARY

The authors have shown that ALM joins together both business management and software engineering in a very powerful development environment through IBM's Collaborative Lifecycle Management using Team Concert on the Jazz™ platform. It is our belief that ALM is a significant advancement in the quest for coalescing into a transparent integrated environment the many activities, roles, and artifacts characterizing modern software development in the business enterprise.

REFERENCES

[1] D. Chappell, "What is Application Lifecycle Management?" Microsoft Corporation, 2008, http://www.microsoft.com/global/applicationplatform/en/us/RenderingAssets/Whitepapers/What%20is%20Application%20Lifecycle%20Management.pdf, downloaded 11/2012.

[2] M. Wood, "The Evolution of ALM," Projectmanagement.com, 2010, http://www.projectmanagement.com/articles/255472/The-Evolution-of-ALM, downloaded 10/2012.

[3] IBM Corp., "IBM Rational Quality Manager," IBM Software, 2009, http://publib.boulder.ibm.com/infocenter/rqmhelp/v2r0/index.jsp?%20topic=/com.ibm.help.common.jazz.calm.doc/topics/c_calm_common.html, downloaded 10/2012.

[4] M. Göthe, C. Pampina, P. Monson, N. Khurram, K. Patel, B. Smith, N. Yuce, "Collaborative Application Lifecycle Management with IBM Rational Products," IBM Redbook, 2008, IBM Corp.

[5] Jazz Inc, "About Jazz Platform," Jazz, 2012, https://jazz.net/story/about/about-jazz-platform.jsp, downloaded 10/2012

[6] IBM Corp. "Collaborative Lifecycle Management," IBM Software, http://www-01.ibm.com/software/rational/alm/collaborate/, 2009, downloaded 11/2012.

[7] A.Meneely, L.Williams, L. Hayward, "Jazz Basics," Software Engineering, North Carolina State University, 2009, http://realsearchgroup.org/SEMaterials/tutorials/jazz/jazz_tutorial.html, downloaded 10/2012

[8] A. Buehlmann, "ChangeSet," Mecurial, 2010, http://mercurial.selenic.com/wiki/ChangeSet, downloaded 10/2012

[9] IBM Corp., "Development Line," IBM Infocenter, 2012, http://pic.dhe.ibm.com/infocenter/rtc/v20m0/index, downloaded 11/2012

[10] C. Babcock, "IBM Preps Team Concert, A Collaborative Development Tool," InformationWeek, 2008, http://www.informationweek.com/ibm-preps-team-concert-a-collaborative-d/206905042, downloaded 11/2012.

[11] C. Babcock, "IBM Sharpens Rational Tools," InformationWeek, 2009, http://www.informationweek.com/development/tools/ibm-sharpens-rational-tools/219401501, downloaded 11/2012