I.J. Information Technology and Computer Science, **2014, 06, 61-67**
Published Online May 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.06.09

Impact of Collaborative ALM on Software Project Management

Engr. **Tahmoor Shoukat**
University of Engineering and Technology, Taxila, Pakistan
Email: tahmoor.shoukat@gmail.com

Muhammad Nadeem Majeed
University of Engineering and Technology, Taxila, Pakistan
Email: nadeem.majeed@uettaxila.edu.pk

***Abstract* —** To produce a release of software, ALM is a key for streaming the team's ability. ALM consists of the core disciplines of requirements definition and its management, asset management, development, build creation, testing, and release that are all planned by project management and orchestrated by using some form of process [1]. The assets and their relationships are stored by the development team repository. Detailed reports and charts provide visibility into team's progress. In this paper we will describe how the ALM involves software development activities and assets coordination for the production and management of software applications throughout their entire life cycle.

Index Terms — Application Lifecycle Management (ALM), Service-oriented architectures (SOAs), Software Development Life Cycle (SDLC), Configuration Management (CM)

I. INTRODUCTION

ALM describe management methods for development of software and IT infrastructure by process automation from one end to another, and integration of information from different steps. Accuracy and regularity is attained along with automation opportunities are introduced through integration.

ALM solutions defined by Forrester [2] as:

"Integrated tool sets that support and unite the following life-cycle activities: requirements management, design and modeling, development, software configuration management (SCM), and testing."

The three core pillars of ALM are:

1) *Artifacts relationship and their complete traceability:* This is usually a manual process, where increase in effort required with the increase in size of projects, the number of artifact interdependencies and the varying size and scope. Traceability is necessity for the compliance of new requirements.
2) *High-level processes automation:* Paper-based approvals are commonly used in the development organizations to manage the handoffs between functional domains. ALM increases the worth of handoffs through automation and maintaining the centralized repository for all related documents.
3) *Visibility can be increased with reporting:* Usually managers have limited visibility of development projects to track progress. Limited visibility and lack of reporting hinders the possibilities for process improvement. ALM provide the deep-dive analysis of all activities with integration and automation provides the status in real-time.

Automation, traceability and tracking accurate progress usually difficult to achieve because contrasting tools do not provide adequate integration. For makers, ALM solution has been a junction to build modeling tools, visualization tools, compilers, IDE's, source code and configuration management tools.

Main advantage of ALM solution is that, all the stakeholders of project share the updated information. Here project managers, developers, testers, architects, system administrators, business sponsors and users are considered as stakeholders. Activities typically supported by ALM solution are requirements elicitation, modeling for solutions, visual designs, development, quality control, maintenance and issue tracking. All the artifacts resulted from these activities are linked together through ALM tools.

The remainder of this paper is organized as follows: Section 2 describes the impact on People, process, information and tools. Section 3 describes the knowledge areas for effective ALM and SPM. Section 4 highlights the business issues and Section 5 explains success indicators. Conclusion added in the final section.

II. PEOPLE, PROCESS, INFORMATION, AND TOOLS THAT DRIVE THE LIFE CYCLE

In real world, peoples from the various teams are distributed around the world and they must co-ordinate and collaborate to develop the applications in limited time, while in some cases stick to government regulations. To streamline the team's performance and efficiency without obstructing the progress is major challenge at any point of the software development. Products must be exposed to market to receive the changing response and products must be constantly updated to retain the interest and fresh look. That's why the demand has dramatically increased and life span of software applications has shortened. Software development teams must produce the competitive and meaningful products at un-parallel rate.

In this kind of environment, teams can no longer work in silos where one team throws a solution "over the wall" and the next team to test and deploy after integration. Also the disciplines of software development are not treated as silos where one team member only handles requirements while another develops the source code or only test the software. Business demands software that meets the user needs. Operational teams should know how to manage applications and handle problems. Business analyst is more involved in development process so that the developers understand the stakeholder's needs with more detailed insight of business processes, design and assets reuse to implement and test the system. Business analysts are also closely aligned to testers to ensure the high quality of software applications driven by customers.

As a result, teams take a much different approach to software development using agile development, mock-ups, faced iterations with potentially shippable increments and constant review meetings and interaction session with stakeholders. Development teams should be aware of and responds to customer feedback. In addition, teams have no longer relaxation of 18 to 24 month development cycles where each detail is keenly planned and designed [3]. Waterfall approaches to software development have given the way to short iterative agile approaches. Feedback loops in agile are necessary for producing software that could meet the users requirements.

ALM not only brings these disciplines and teams together; it also clears the ambiguity of their progress and inspects the working of teams. Now a day it's difficult for software development teams to function without it.

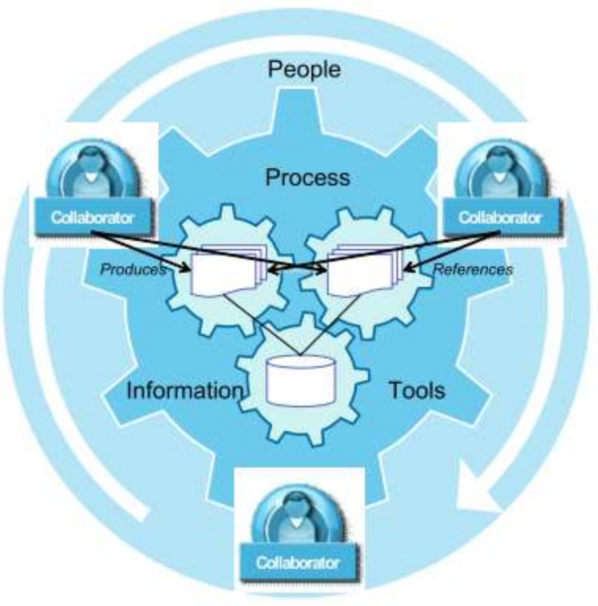

Fig. 1. The coordination of people, process, information and tools in ALM [1]

Coordination and contributions between team members lies at the heart of ALM. This involves people, process, information and tools with transparency aspects and shared responsibility for success. Fig1 provides a simplified view that involves a series of contributions between team members; what one team member produces, another team member consumes. Understanding the relationship helps to manage their efforts, rationalize the ability to produce software and produce a healthy software development environment.

The term *Application Lifecycle Management* comes to define the market demands for a suite of integrating tools that help teams to manage the assets in software project. ALM can also be defined with the set of roles associated with specialized set of tools. With this definition, analysts, architects, testers, build engineers, deployment resources and project managers were all considered as key members and this approach is successful at broadening the definition of software development team [4]. A by-product of this role-based approach led to silos of capability.

Key business challenges are:

- Reduce the cost of development
- Increase the productivity of Staff
- Decrease time of product to reach market
- Improve Product Quality

A. Awareness Barriers

There is no doubt that the biggest challenge in software development is awareness. Each area managing their assets with their own set of tools and repositories. So it's quite difficult for the teams to collaborate and merge their effort for the common cause. So the information about the project, the teams and the outcome is crucial to manage the solution through complete life cycle.

1) *Project awareness:* If the project teams are distributed around the globe, it's extremely difficult to collect and manage project metrics. Management and multiple releases and immediate tool selection helps to fix a high priority defect in production system. Tracking the work of team individuals who are distributed around the globe and collaborating on the multiple releases at a time is critical for project success.
2) *Team awareness:* Individual team members should learn to effectively switch the context as they needs to move from one project to another. They should be aware about culture, policies and practices of the team they are going to join. They should also know how project is configured, how to report changes and how whole team is progressing.
3) *Build awareness:* In general, build present the functional features irrespective of internal details. Testers are ready with test scripts and test cases when build is ready if they know what is planned for coming build [7]. Also the whole team should know about the changes that are implemented in the build and what defects are handled. That will help them to catch the ripples produced as a result of some defect fix and also help the test team to target their effort

and apply meaning full test cases. When build face major problem, the team troubleshoot the context and efficiently fix the problem.

B. *Organizational barriers*

Organizational boundaries prevent the teams to collaborate as free as they need to be successful. Because of those boundaries visibility of cross-team and their interaction is limited. Different organizational cultures, embedded politics and management support affect the progress. Teams should overcome those barriers to produce successful products.

C. *Functional Barriers*

Software development involves different roles. As a result, multiple tools are used for each discipline. Different tools and vendor choices are done to facilitate the needs for each discipline. Role of Managers is quite critical because they are responsible for integrating multiple teams using different tools provided by multiple vendors.

D. *Geographical Barriers*

Multiple challenges rose when teams are distributed around the globe. In order to complete their work all team members coordinate and collaborate efficiently and they should have access to each other's work. High speed networks are necessarily in all locations for their communication [5]. Even those barriers are conquered, team members must gain respect for other's cultures and develop understanding how and when they communicate to each other.

E. *Technology Barriers*

New architectures like SOAs demand new skill set and new approach to software development. Also organizations faced the challenge to retraining workforce which can be pricey and time consuming. With the offshore development and outsourcing, new coding practices and techniques are evolved. Mangers need to set up the coding practices and analysis of source code to ensure the source code consistency. As enterprises looks to develop enterprise architecture so the need to be consistent in all business units is crucial. If multiple teams implement the same solution in different ways is not only time wasting but this inefficient way also confuse the stakeholder as well. Users who got multiple approaches recognize the enterprise as disorganized. Such practices can lead to a loss of business.

III. EFFECTIVE ALM AND SPM

ALM coordinates the flow of people, processes and information in an iterative cycle of software delivery activities [6]. The coordination is made possible by applying the business management capabilities which are also addressed in the PMI's knowledge areas:

- Planning and change management
- Requirements definition management
- Architecture management
- Software configuration management
- Build and deployment management
- Quality management

IV. KEY BUSINESS ISSUES

Generally numbers of common issues are faced by our technical and business decision makers.

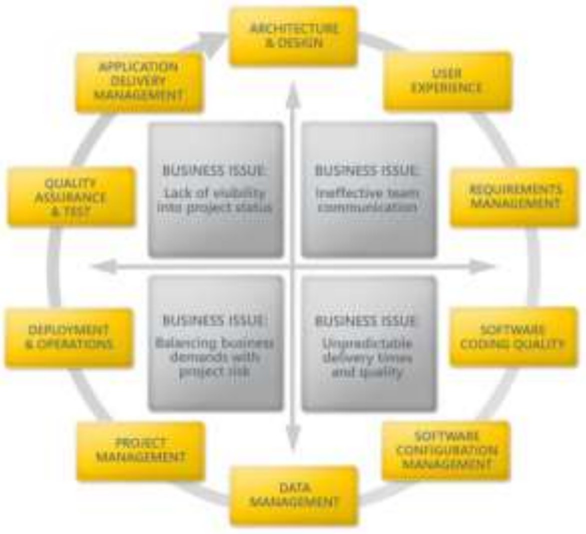
ARCHITECTURE & DESIGN
USER EXPERIENCE
REQUIREMENTS MANAGEMENT
SOFTWARE CODING QUALITY
SOFTWARE CONFIGURATION MANAGEMENT
DATA MANAGEMENT
PROJECT MANAGEMENT
DEPLOYMENT & OPERATIONS
QUALITY ASSURANCE & TEST
APPLICATION DELIVERY MANAGEMENT
BUSINESS ISSUE: Lack of visibility into project status
BUSINESS ISSUE: Ineffective team communication
BUSINESS ISSUE: Balancing business demands with project risk
BUSINESS ISSUE: Unpredictable delivery times and quality

Fig. 2. Business issues addressed by organizations [8]

Four common challenges faced by every organization are:

1) *Limited visibility of status:* If organization not able to enforce the accountability, responsibility, check points and sign-offs, then this would be the totally project management issue. If organization not able to involve the stakeholder, not able to provide precise estimation and not able to bend the timeline of project then these are the signs of management issues.
2) *Effective communication between teams:* Merging the work across organizations and provide effective medium for communication is extremely critical.
3) *Matching business needs with risk:* Change in requirements, unrealistic estimates, ambiguous business objectives and rapidly developing technology contain this issue.
4) *Unpredictable delivery times and quality:* Maintaining functional requirements, requirements for quality of service, schedule and budget is a major challenge. Bugs found at the last moment during testing in production occurred too frequently.

V. SUCCESS INDICATORS

In Collaborative Application Lifecycle Management (CALM), Assets are produced by certain roles in life cycle. To succeed in this orchestration of people and assets, CALM relies on the following key points to effectively develop software:

- Collaboration
- Distribution
- Traceability
- Automation
- Continuous improvement

A. Collaboration

CALM focuses on the management of entire team and their handoffs working across domains. These points of integration make development process efficient and if they are missing they create problems between team members. Using CALM, it is not satisfactory if a team member works in isolation, because CALM demands collaboration in order to complete their own tasks.

Decisions are made that will help the whole team to progress, and also letting all team members aware of what other team members are doing. In Soccer, all team members collaborate to move the ball forward and stop the opposition to score. If each team member plays alone irrespective of other players on the field, their success chances are very limited. To increase their chances of winning, they should collaborate according to the situation on the field.

Similarly development of software is a team sport. Being successful does not mean that you can act alone, the best team members are collaborators. Like a successful soccer team, collaborators work each other and respond according to the situation of the project. The agilists were the first ones who discover the need of collaboration and now everyone realize the need.

Software product is the result of many conversations. Teams must be able to coordinate in many ways described below:

- Team awareness enables team members to know who is available online so that they can interact.
- Instant messaging enables the team members to interact each other instantly irrespective of their geographic position. A good example is a developer who communicate with tester over a bug to produce it again.
- When team members subscribe to RSS feeds, they are notified about the events occurred within the repository. User can also subscribe for the feeds on single artifacts to get notifications whenever some changes has been made.
- Ratings enables the team members to rate the quality of work, thus enabling the community to judge the value. Poor quality assets can be improved, while good quality assets are utilized often with confidence, thus improving the quality of overall assets.
- Tagging provides the option to tag resources and find them instantly without relying on the search. User can tag things to their work thus making it easier to locate when required.

An ALM solution provides support to the user irrespective of what they are and where they are. It must provide support for collaboration that results the creation of assets.

B. Distribution

To support the decisions for agile business and flexibility, enterprise ALM platform are required to integrate roles, teams, workflows and repositories.

Now a days in software development, teams are distributed geographically. No matter team members are in same country or different countries, are working in home or a third party outsourced, development teams are hardly at same place. An ALM solution will be effective by providing the high-speed connection to all members. High security standards on the assets are also introduced with the rise of outsourcing work. Visibility to the assets must be controlled with in the repositories.

With team’s distribution, systems are also distributed that manage the assets throughout the enterprise. An ALM solution must connect the team irrespective of where they are located and where their assets are stored.

C. Traceability

Tagging other resources both internal and external to the assets provides traceability. Now a days in software development, linking the related resources becomes an essential need in completing tasks. For example, while writing the test case, tester must review the requirements that the test must validate. Requirement link added with the test case, streamlines the ability to complete the test case writing.

Traceability also enables to traverse the linkage between the resources. For example, traceability in artifacts enables the architect to check how many test cases are applied to validate the specific requirement. Another example is to check how many bugs exist for meeting the specific requirement.

Traceability also enables the team to provide details answers to a regulatory audit. By having the traceable repositories, a team can respond to auditor’s questions relating the number of changes done into a release and how changes are validated. Traceability helps to complete our work by providing the ability to access resources in context of work. Also helps to understand the dependencies and relationships between resources.

D. Automation

Fundamentally ALM involves management of cycle. However, it is interesting to note that there are interim cycles within cycle for a software project. Some tasks are creative in nature and require attention to resolve. Other tasks, however are repetitive and not creative. Those noncreative tasks can be disposed with automation. Testing is the major example and automation makes it much easier.

To streamline the lifecycle, many other forms of automation can be utilized. For example, build creation brings value to team members by automating the processes of build system, applying validators (baselines) to source control systems, conducting and running build verification tests, and staging and packaging the distribution store for test team. Thanks to automation, kicking off the build now involve the full process.

“Scanning” such as static analysis of source code can also be done with automation. Developers usually perform static analysis before delivering their changes or it can be added during the build process before the compilation. Source code quality and consistency in code can be improved by this type of analysis. Security scan is also performed with a predefined criteria. This scan works in same manner as virus scanning on home PC. The definitions are created by vendors and organizations use the scanning tools perform the job by keeping the definitions up to date. The scanning tools produce the results in the form of threat identification and then threats can be treated before releasing the application to production. Security scans test the application with the predefined definitions and set of tactics usually used by hackers. Organizations can identify the threats by using this type of automation. Compliance scan also works in same manner where test are performed with certain regulations. The scan highlights the noncompliance and team respond to those violations.

Noncreative and repetitive tasks should be performed with the automated processes of ALM solution in a traceable and meaningful manner.

E. Continuous Improvement

If software development team seeks the effectiveness of ALM then this team must seek areas of improvement. Continuous planning, integration and testing are the key areas for consideration [11].

1) *Planning and feedback:* Planning for project is a continuous activity that involves evaluating the current system, identifying the weak areas and making corrections. Project planning includes following dimensions:
   - Cadence
   - Transparency
   - Application health
   - Retrospectives
2) *Integration:* Continuous integration occurred at various levels, between the developers, team build and solution level build. Usually developers code and test their changes before committing the changes to team repository so the integration starts with the developers. In some cases, build scripts and build environment that are used by developers are different from the team environment. Which causes problems for the developers when they commit their changes. An effective integration through ALM solutions ensures that the consistent build environment should be used by all the team members.
3) *Testing:* It is no longer acceptable to perform testing at the end of development. Rather reactive teams involve testing throughout the development cycles as specified by the following types of testing which contributes software quality improvements.
   - Developer testing
   - Build verification testing
   - Test planning

Testing is added as part of each iteration in iterative models. Each iteration involves unit testing, build verification, functional, integration and system testing and testers are involved in multiple phases of testing.

VI. Steps to Implement Application Lifecycle Management

Application lifecycle management (ALM) is different from Software development cycle (SDLC). ALM which is a part of SDLC provides more details how application is developed. Now days each company manage its application development in different way. Department and manager have its own method of supervising and conforming standards during application development. For example, a company which follows waterfall SDLC is different from a company that follows agile SDLC.

Now days, almost every tool provide an integration options with ALM and each company will use it according to its needs.

Through the software development methods which are used by companies, ALM is indicative of how well its end product will meet the milestone for being successful.

All external and internal stakeholders should be agreed upon on the ALM processes during and after the application development. A standalone role or person can’t take the place of a multidimensional process. Sometimes project manager dictates the project process based on his or her priorities of what process should be—though he or she is more concerned about time, scope and cost. The impact of ALM and process is the responsibility of quality assurance department.

There are 11 steps to successfully achieve the application lifecycle management [9].

1) *Defined roles should be part of the ALM:* Define the responsibility of each role in Software technology, Information technology, Quality assurance, Software testing and the business stakeholders on customer side.
2) *Goals and objectives should be defined for each role:* If this step is not followed then the details for each role will not be unstated and the question of “what will I do?” will never be lucid. This step helps to avoid finger pointing each other.
3) *Foresee the dependency of internal-processes:* Mark the outline and explain which roles and what work will depend on your deliverables and actions. If this is not mentioned clearly, other people on the project assume that time is not crucial or milestones can be postponed or covered later if done improperly. A proper ALM process ensures that all work is done correctly and follow standards for quality.
4) *Develop a risk and contingency plan:* Each step in the process has the potential to give impression if something goes wrong. You have to think about the possibilities how a missed deadline and objective could be maintained without a major interruption to the overall ALM process. If the risks are not understood and maintained from the start of the project, then you will face the catastrophic effects later in the project. Problems distressing time, functionality, outcome, quality, performance improvement, security concerns and budget all add to the risk of the end product.

5) *Process preparation:* Defining the flowing path of documentation and communication is quite important for ALM. That's the point where the SDLC comes to play and company or department defines how they will operate.
6) *Predict the potential output from each individual:* Recognize the capabilities of each person in your team and expect the outcome accordingly.
7) *Plan again and again:* In order to make ALM successful and beneficial, then detailed planning is a must. At the start of each project each role must be defined and their role's objectives, needs and dependencies should be known.
8) *Effect of communication:* Communication with the customer is crucial because client expects something whenever you interface with him/her. As earlier your clients are involved, they will feel a part of the process and accept your ideas and thoughts.
9) *Apply quality assurance checkpoints:* Usually quality analyst set up the checkpoints to ensure that the ALM process is working. (Here quality analyst is different from the software tester, Quality analyst ensure that all the processes are running). When project is being executed by a project manager, they also want to manage ALM; in that case it's hard to give good quality assurance since there is no opportunity for outside entity to perform evaluation. Checkpoints added by the quality analyst helps to balance the process.
10) *Work for continuous improvement:* It is QA's responsibility to highlight what went right and what went wrong in order to correct any problems in the process at the end of each lifecycle attempt. A document "lessons learned" should be created and offered to all the major stakeholders to rectify the problems and improve for the next attempt. If this doesn't happen then the process truly not exist and shows that one person on team is "running the show" instead of collective group [10].
11) *Provide control:* When the baseline is set, all process activities must go through the formal plan to ensure new rules within the company's adoption of ALM process. That's why the quality assurance checkpoints are so important. QA monitors the processes that are applied and control works with in-house regulatory and audits for standards.

VII. ALM Quality Dashboards Designing And Implementation

Managers' needs to deal with different software projects at the same time and there are a lot of challenges they face each day. Identify the nature of projects and the measure make sense across projects, implication of data collected from the various sources, handling them across different project sizes, choosing meaningful ways to present information and effective utilization [12]. Targeting the analysis, design and execution formulate the dashboards actionable, intuitive and effective.

A. ALM quality dashboard analysis

1) *Metric:* Metrics are required for the dashboards that incorporates all the phases of application lifecycle – requirement elicitation, design, development, testing and maintenance. Also should care about the efficient collection and effectiveness of metrics. For example, effect measure for requirement elicitation should be balanced with the effective measure for end user satisfaction. If the addressed requirements are not prioritized and work in the same systematic manner for the reasonable period of time, end user would be not satisfied and may feel that they are part of lengthy accumulation.
2) *Normalization:* Applications may vary in size; mainframe-based, enterprise-level, client server or mobile. They may encompasses multiyear, large-funding projects or short-run, simple projects. The dashboard needs to have same measures on all projects so that they can be comparable in meaningful manner. Often for meaningful comparisons, weighted projects based on size and complexity.
3) *Actionable metrics:* If the metrics are a part of an ALM dashboards, they should be actionable and the actions taken from the metrics should be practical and implementable to improve the ALM quality.

B. ALM quality dashboard design

1) *Minimum no. of metrics:* If too many metrics are provided, ALM quality dashboards would be useless. The main idea of the dashboard is to provide comprehensive picture of quality through the metrics without spending a lot of time. It is always a good practice to classify the metrics as primary (mandatory) and secondary (Optional) on the basis that how much they affect the quality of ALM. And the primary ones should be provided on dashboards.
2) *Detailed display:* Try to keep all quality information on a single dashboard screen, whether it's webpage or a pop-up on a computer or mobile device. Mobile devices like smartphones and tablets are becoming ideal for dashboards.
3) *Filter capabilities:* Dashboards just provide you summary for analysis. Advance actions might not be feasible without the in-depth metrics analysis. Filtering will help user to extract the variances with convenient accessible statistics.

C. Execution and Follow-up

Various test management solutions provides the data for quality metrics. This info is managed in centralized manner so that desired analysis and action could be done. Feeds are managed on the dashboards coming from db queries, data in files, sheets and reports [13]. Monitors should be applied on such sources to make sure that the data have been received from all sources before generated on dashboards.

1) *Comparing timeline:* The timelines presented in the data should be comparable because data sources are

directly linked with dashboards and timelines coming from files and reports are managed in synchronized manner.

2) *Actions to make it correct*: Actions which are taken to correct should be linked with the dashboards to people those are responsible for quality metrics. If queries or notes can be directly emailed to the liable person that will provide/improve the sequence of corrections at the level of drill-down.
3) *Follow-up and repeat:* Reporting tools should provide feedback and discussion options as feedback loop fashion. Metrics which are not easily accessible and actions which needs to be taken should be available on the dashboard for convenient accessibility.

ALM quality dashboards provide a large amount of details for their tracking, analysis and decision needs to be taken. In product lifecycle, development teams might be geographically dispersed and applications vary in size and complexity [14]. Creating a single ALM quality dashboard in those cases is difficult and challenging, but if we design and implement the dashboard for efficiency and effectiveness by focusing on key characteristics.

VIII. Conclusion

In this paper, we provide the blueprint for Collaborative Application Lifecycle Management (CALM) and demonstrate how the CALM solutions support the development organization by relating with the basic aspects of Software Project Management. In this paper, we provide the steps for deploying ALM solutions into an existing enterprise environment. Our primary focus is to highlight the value of CALM by explaining a user view of the solution that supports distributed enterprise development team that incorporates the critical aspects of "Agility-at-Scale" approach.

References

[1] Göthe, M., Pampino, C., Monson, P., Nizami, K., Patel, K., Smith, B. and Yuce, N. (2008) Collaborative Application Lifecycle Management with IBM Rational Products, An IBM Redbooks publication, December 2008.
[2] Schwaber, C. (2006) The Changing Face of Application Life-Cycle Management, Forrester Research Inc., White Paper, 18 August.
[3] Jalali, S. and Wohlin, C. (2010) Agile practices in global software engineering – a systematic map, International Conference on Global Software Engineering, ICGSE 2010, Princeton, NJ, USA, August 23–26, 2010, pp. 45–54.
[4] Goth, G. (2009) Agile Tool Market Growing with the Philosophy, IEEE Software, Vol. 26, No. 2, pp. 88–91.
[5] Battin, R.D., Crocker, R., Kreidler, J. and Subramanian, K. (2001) Leveraging resources in global software development, IEEE Software, Vol. 18, Issue 2, March-April 2001, pp. 70–77
[6] Driving Your Business Forward with Application Life-cycle Management (ALM), Microsoft, White Paper, August 2007.
[7] "Eleven steps to kickoff application lifecycle management" by John Scarpino, D. Sc.
[8] Doyle, C. and Lloyd, R. (2007) Application lifecycle management in embedded systems engineering, Embedded System Engineering (ESE magazine), Vol. 15, No. 2, March, pp. 24–25.
[9] Carolyn Pampino (2011) Five imperatives for effective Application Lifecycle Management, IBM December 2011.
[10] Ebert, C. and De Neve, P. (2001) Surviving global software development. IEEE Software, Vol. 18, Iss. 2, March–April 2001, pp. 62–69.
[11] Booch, G. and Brown, A. (2003) Collaborative development environments, Advances in Computers, Vol. 59, Academic Press.
[12] Nari Kannan (2012) Designing and implementing ALM quality dashboards. ALM and Agile Strategies - July 2012, Vol. 1 Iss. 3
[13] Dearle, A. (2007) Software deployment, past, present and future, Future of Software Engineering (FOSE '07), 23–25 May, Minneapolis, MN, USA, pp. 269–284.
[14] Murta, L., Werner, C. and Estublier, J. (2010) The Configuration Management Role in Collaborative Software Engineering, In: Mistrík et al., Collaborative Software Engineering, Springer-Verlag, Berlin, Heidelberg, pp. 179–194.

Author's Profiles

Engr. Tahmoor Shoukat is MS Scholar in Department of Software Engineering at University of Engineering & Technology, Taxila. He has done his BSc Software Engineering (with distinction) from the same the university in 2009. He worked in various Software organizations in Pakistan. Currently he is working in Ministry of Defense, Pakistan. His areas of interest are Digital Image Processing, Software Quality and Software Project Management.

Muhammad Nadeem Majeed is PhD Scholar in Department of Computer Engineering at University of Engineering & Technology, Taxila. He holds a MS degree in Computer Engineering from Center for Advance Engineering, University of Engineering & Technology Taxila and has 11 years teaching experience. He is currently serving University of Engineering & Technology as Assistant Professor and working on Optimized Vertical handoff algorithms in vehicular Ad-hoc network. His research related to Risk management in IT industry of Pakistan which is aimed to aid different managers and team leads to manage the risk in their software development

How to cite this paper: Tahmoor Shoukat, Muhammad Nadeem Majeed,"Impact of Collaborative ALM on Software Project Management", International Journal of Information Technology and Computer Science(IJITCS), vol.6, no.6, pp.61-67, 2014. DOI: 10.5815/ijitcs.2014.06.09