International Journal of Application or Innovation in Engineering & Management (IJAIEM)
Web Site: www.ijaiem.org Email: editor@ijaiem.org
Volume 3, Issue 10, October 2014 **ISSN 2319 - 4847**

Scaling Agile Requirements to the Enterprise level through the exploration of Self-Organizing Software Development Teams

P. Ashok Reddy[1], Dr. K. Rajasekhara Rao[2], Dr. M. Babu Reddy[3*]

[1] Sr. Asst. Professor of Comp. Applications LBRCE-Mylavaram, Krishna Dt., AP, India

[2]Director, SriPrakash Educational Institutions, TUNI, AP, India

[3*]Asst. Professor of Computer Science Krishna University, Machilipatnam, AP, India

ABSTRACT

Conventional software development methods are being used in Organizations for more than three decades. Subsequently, the dynamic market conditions and the changing roles of business organizations have pushed the development methodology to 'Agile' , which is a fast, adaptable and cost efficient methodology which suits to the time-to-market conditions very well by preserving the interest of all stakeholders in a project and it promotes transparency and visibility which are crucial aspects for success of any project. Through continuous customer feedback and early release of products, Agile methodology has witnessed acceptable results in producing reliable software in line with the customer business needs. However, to scale the Agile requirements management to the enterprise level, various functionalities associated with project stakeholders need to be thoroughly understood and appropriate protocol need to be designed. In this paper, various issues were discussed in the lines of stakeholder involvement in getting successful results through Agile methodology and an effort has also been made to analyze the specific issues associated with performance testing under Agile development environment; and a thought provoking light has been thrown to encapsulate the agile strategies like: feedback collection and code refinement with performance testing at Enterprise level.

Keywords:- Agile Software Development, Scaling Agile Requirements, Performance Testing, Self Organizing Teams

1. INTRODUCTION

For the past few decades, conventional software development methodologies like 'Water Fall Model' has been widely accepted by the software development community and resulted in development of thousands of software products through a systematic and well planned development phases. But, water fall method is a nightmare to the development team throughout the development process, because, at any time, change request may be received from customer which may disturb the entire schedule and it may incur additional financial burden[1]. The present day time-to-market business environment has influenced the conventional style of software development and a set of new methodologies called Agile Methodologies have emerged and become popular. Agile methodologies emphasize small design steps for incremental development guided by customers through frequent interactions. At every step, customers and developers agree on the next feature and capabilities for the software. These methodologies are thought of as being small and cohesive teams. The key ideology of Agile methodology is "Release Early and Release Often". Agile methodologies works around the ideas of building software that matches the requirements of especially to a class of users, working together with those users throughout the design and implementation phase, combining design and implementation, working in groups, respecting technical excellence, doing the job with motivated people, and generally engaging in constant design/redesign. Agile is a light weight framework for helping teams, giving a constantly evolving functional and technical landscape. There are few other methods like: Pair Programming and Spiral to facilitate the customers and developers to have fast and efficient projects. However, the Pair Programming lay our self open to more cost to the Management and the Spiral is not convenient for development and Quality Assurance teams though it is user friendly[1][5][6].

Agile preserves the interests of all stakeholders of the software project and ensures the transparency and visibility which are essential aspects for success of any project.

It benefits the customers: through frequent releases; Managers: through minimum documentation, no useless meetings, clear and acceptable communication with all stakeholders; Development team: by shortening the learning curve and development time; Testing team: through early access to the software to initiate testing process[13].

1.1. Scaling issues

New challenges are introduced if the team members are geographically distributed or many teams are working for a single product owner. Rather than creating a single 100-person team, the agile encourages to split developers into multiple smaller teams. Planning a project with large, multiple teams may require:

i. Establishing a basis for estimation so that all teams will have a common unit for estimating.
ii. Adding details to user stories to understand the big picture and help the teams to coordinate their work better.
iii. Performing *look a head* planning so that teams can successfully work together. This is useful if teams have interdependencies which require them to coordinate their work for next couple of iterations
iv. Incorporating feedback buffers so that unexpected event in one team won't risk the whole project. Teams can also include the feeding buffer in iterations that deliver capabilities needed by other teams.

However, this level of planning is not required if the project contains small teams as long as those teams communicate often. Involvement of whole organization is needed to adopt agile methods in large scale. Management needs to provide enough resources to ensure effective communication. On the business level, large scale agile development affects the other departments such as sales and Quality Assurance. Therefore, for successful agile development, finding of large scale scrum may be the driving force. Different agile scaling factors can be found to understand the challenges in different areas[5][6][12]. The dynamic business requirements drive the selection and implementation of technology and tools. While selecting the required tools as part of scaling initiatives, development teams should focus on present and future needs as well. For supporting larger agile systems, the required infrastructures need to have sophisticated business and performance requirements. In these lines, identifying/ developing a high end light weight management tool will be a great challenge. But, any management tool has to be powerful enough to aggregate and coordinate many artifacts which flows through the delivery lifecycle of the product and the tool must be useful to all the stakeholders.

2. SCALING AGILE REQUIREMENTS TO THE ENTERPRISE LEVEL

Individual project teams focus on their individual areas of responsibilities and they don't operate at an enterprise level. Hence, their primary approach to implement the requirements is same as they were not part of an enterprise. However, a cohesive and coordinated strategy will be executed collectively in a healthy enterprise. Long term product roadmap and short term product backlogs will be managed by product team and product manager in line with the relative priorities of stakeholder goals and market objectives. To scale to the Enterprise level, operational simplicity and complexity of the tasks need to be taken into consideration. In the process of articulating how goals for the product fit into the enterprise strategy, simplicity issue may likely to be raised and incorporated by any team member or role. The product manager will not be managing all the strategies. In the enterprise level, it is quite natural to investigate the requirements of complementary goals and the opportunities to find out alternate solutions and to achieve required additional understanding of the roles that other products play in that strategy. The source of Complexity lies in it only. The product managers can also explore the possibility of rethinking the ownership of the enterprise goals to ensure that the two or more products meet the enterprise objectives. Product delivery synchronization is another key aspect where the impact of transparency and visibility plays the major role. When the goals are coordinated at a high level, delivery must be synchronized in detail. Once the roadmaps of various products were aligned at thematic level by their product managers, the product owners also need to synchronize their deliverables within the schedule to facilitate the suitable interoperability and to get customer feedback. However, the product teams need to have coordination to evolve the required Application Program Interface(API) to minimize code-writing dependencies by allowing them to have liberty in day-to-day operations[13]. When more than two teams are coordinating, the efficiency of one-to-one coordination jumps down, the teams will acclimate by introducing Activity Planner – a daily standup meeting where product owner and Activity planner update each other on the completed and planned synchronization activities, and identify barriers to their teams that impact other teams.

3. PERFORMANCE TESTING

Many organizations have migrated themselves to Agile methodologies, but have not fully integrated performance testing. Few of the approaches which have witnessed significant levels of success are:

3.1. Center of Excellence:

The functionality is quite similar to that of outsourcing the testing task to a dedicated internal group of team members whose regular task is to take up the performance testing issues assigned to them. Most organizations that are trying to incorporate their performance testing into their Agile development cycles as there was a provision for outsourcing the performance testing tasks in the conventional style also. To witness success stories in these lines, there are few things that need to be taken care are:

- Performance targets in addition to the goals and objectives must become an essential part of each stakeholder story.
- Make use of this service at regular intervals not only at the time of product release.
- A full time team member must take the responsibility to ensure the overall effective performance testing activity and the individual developers must take the responsibility of their own module/unit/component.

When performance of the system is very important and the developers are not taking complete responsibility of unit testing and tuning the product continuously at their levels, and when there is no provision to allot full time exclusive members for performance testing, this 'Center of Excellence' method is likely to be insufficient.

3.2. On call/On task basis:
Not like the 'Center of Excellence' method, a specific performance tester is assigned to one or more development projects. In this method, mostly the performance tester will work independently, but can provide some advice periodically based on his performance testing expertise and project level knowledge on each project. For this model to be more successful and useful, the performance tester must be accessible during the critical times of performance evolution. Sometimes, this method may not be in a position to suit to the requirement of performance testing at critical times, as a result, it may fail, because, at that time, the same performance tester might have been assigned and held up with one or more other product performance testing tasks.
3.3. Inclusive Captivation: There is a full time specialist member of the team whose primary responsibility is to coordinate and manage performance related activities throughout the life cycle. However, performance is a part of everyone's responsibility. But, adapting this method requires to have human resource with right skills, test environment and required tools to be assigned to each development team. Performance testing itself is an iterative process and thus Agility can be observed even if the development teams follow conventional development methods and in all the methodologies, performance testing need to be given attention all the way through the development life cycle. However, depending on the product nature, timelines, quality standards expected by the customer and organization, team size, etc., every product team tries to adopt its own approach for performance testing.

4. AGILITY THROUGH SELF ORGANIZING TEAMS

The culture of self organizing teams is unfamiliar concept to majority of the software professionals. In the recent software development scenario, there is a pressing need to have a close look into the roles of team members and managers while shifting to Agile methods. When the Agile team is considered as Self Organizing, team members will work independently with required collaborations and share a common goal[2][3]. Rather than the managers having complete responsibility over various activities, the team members feel responsibility for managing their own work and gradually emerge as decision makers and conflict eliminators. However, the role of Managers can still be witnessed to have liaison with the rest of the Organization, to build trust, facilitating and supporting team decisions, expanding team capabilities and to influence change[4]. Though the process of transition to self organizing teams be scary, Agile practitioners can grab benefits by understanding the team dynamics clearly.
4.1. Initiating the Team activity with appropriate and effective beginning: During the initial stage, the team members may not be fully committed in working environment and often they will feel independent and excluded from the rest of the team members. They will spend most of their time in gathering required inputs for paradigm shift to agile practices. At this stage, the traditional role of Managers need to be transformed as a facilitative leader for extending necessary support to the team members so as to assist them to gear up with their work[1].

4.2. Initiating best practices towards team Self Organization by way of encouraging effective team dynamics: For the success of any team work, proper understanding of the role and effective communication plays crucial role and this helps in transforming in to truly Self-Organizing status. Building emotional trust among the team members is an essential element of efficiency. Through effective team cohesion, teams ability to resolve conflicts can be developed which in turn will lead to full self-organization with true agility and acceptable performance[6][10].

4.3. Continuous tracking of team's progress towards Self Organization:

The secret to any team work depends on the understanding of team dynamics and circumstances that promote the team dynamics. As the teams move through the well documented stages, the various issues associated with team activities will be changed [12]. In every stage, team interacts with six teamwork components like: responsibility sharing, sagacity of reason, performance, required communication among the members, commitment towards the target and functioning agreements [12]. The stages may not be always in a sequence; up and down movements will be observed due to the influence of many internal and external forces which include: reorganization of teams, lack of supporting administrative systems, nature of task to be completed, etc., Moreover, teams may exhibit the behavior associated with multiple stages.
4.4. Choosing appropriate strategies to ensure greater productivity and customer satisfaction: During the course of development, majority of the agile teams may encounter challenges which may influence the performance. To ensure team's ability towards the goal, appropriate interaction skills are required. Moreover, team members need to understand each other's commitment and stress in getting their job done. At this stage, the managers need to recognize it as progress in right direction and they need to extend their support in all aspects. To accelerate the momentum towards

final results, team cohesion need to be promoted through the acknowledgements and celebrations even for early minute success.

5. BEST PRACTICES FOR MOVING TO AGILE DEVELOPMENT

To ensure the smooth transition to Agile development, the initial step that need to be carried out is to involve testers and customers at each stage of iterative process in order to minimize the future risks. To align the testing in agile team, a team member should be assigned formally the duty of the testing and testing should be carried out in the same iteration as the coding, or else, the basic collaboration will not be working fine and must be revisited. Compared to that of waterfall model, here, testers are being introduced little bit early in the product life cycle. This helps the testers to gain better understanding of the agile methodology and can directly start determining the suitable levels of testing.

- One more success factor lies in according same access to the tester in team activities as other team members. Though such kind of access is not provided in conventional development methodologies, there is no reason for not allowing the access to testers. Gaining familiarity by the testers will help in realization of the results of Agile methodology.
- Making testing activity as part of team communication when testing need to be carried out on the outcomes of various stages of development life cycle. For example, while designing and finalizing user interfaces itself, the suitable test scenarios should be identified and finalized.
- It is a common practice and responsibility of the development team/exclusive testers to ensure that the product is error free and suits to the customer requirements before it is passed to the customer for acceptance testing. Under the Agile methodology, customer is also involved in the testing process so as to ensure the successful acceptance test and there by little effort can be wasted. While including the customers in the testing process:
- Customer should be given freedom to filter the objectives and should receive the testing objectives and standards [6].
- Assigning formal responsibilities to customers may yield better results.
- Customer should have the access rights to testing results as other members of project team [11].
- Sufficient documentation and testing is necessary to ensure adequate testing. Sophistication of application will help in determination of appropriate level of documentation and team's strengths and weaknesses will also help as guidelines in determining the sufficiency of documentation and level of testing.
- Complicated algorithms will most likely require sufficient documentation, or else, the team may find itself continuously revisiting the same algorithm which is waste of effort.
- Identifying what is necessary and what can be neglected can also helps as better guideline and check list is probably the more appropriate option in this situation.
- Just pushing the whole product to an exclusive testing portal is a waste of resource when it is not being used by the team. To ensure the focus on particular area of the code, usage of test instance is advisable.
- For documenting sophisticated design challenges, preparation of one-time document is essential; whereas, it provide very little value to the agile teams.
- In view of the fact that testing is part of the process in agile development, team should be functioning with more collaboration. Thus, large guidebook testing sets are not good sign.

6. CONCLUSION

Though agile methodologies have proved to be the better choice to meet the dynamic customer requirements and to ensure high customer satisfaction, using agile methodologies in larger scale has many challenges and can introduce many unpredicted events and constraints. Agile methodologies like Scrum and Kanban works well at team level but not on the organization level. In spite of the advanced technology, communication among the distributed teams is still seen as a major challenge. As face-to-face communications method is the most effective way, teams should be co-located whenever possible. However, global distribution can introduce cultural challenges which can be blocker for successful agile scaling. It is difficult to point out legitimate objective indicators to assess the success of agile and most of the times, decision is based on the opinion of different stakeholders. There are no fixed/rigid strategies or best practices for scaling agile methods, but context based decisions are always advisable. Without giving noteworthy thoughtfulness given to proper training, inter-departmental communication, refined administrative practices, adequate infrastructure and business process consistency, the spread of agile in organization can be initiated with slow pace, if not completely disallowed.

REFERENCES

[1] Diana Larsen - "Team Agility: Exploring Self-Organizing Software Development Teams", 2004 Diana Larsen Industrial Logic and the Agile Times newsletter.
[2] Leffingwell, Dean. 2007. Scaling Software Agility: Best Practices for Large Enterprises, Boston, MA: Addison Wesley.
[3] Architecting for Large Scale Agile Software Development: A Risk-Driven Approach Ipek Ozkaya, SEI , Michael Gagliardi, SEI , Robert L. Nord, SEI
[4] Rashina Hoda, James Noble and Stuart Marshall – "Organizing Self-Organizing Teams"
[5] N. B. Moe and T. Dingsoyr. Understanding shared leadership in Agile development: A case study. HICSS,0:1–10,2009.
[6] N. B. Moe, T. Dingsoyr, and T. Dybå. Understanding self-organizing teams in agile software development. In ASWEC'08:, 76–85, Washington, 2008. IEEE Computer Society.
[7] Leffingwell, D. Scaling Software Agility. Upper Saddle River, NJ: Addison-Wesley, 2007.
[8] Grant, T. "Navigate the Future of Agile and Lean." Forrester, January 10, 2012.
[9] Rashina Hoda, James Noble and Stuart Marshall- "Self-Organizing Roles on Agile Software Development Teams",March 2013 (vol. 39 no. 3) pp. 422-444
[10] Hoda, Rashina – "Self-Organizing Agile Teams: A Grounded Theory", 2011 URI: http://hdl.handle.net/10063/1617
[11] http://www.versionone.com/White_Papers/Five_Success_Factors_for_Scaling_Agile- Webpage visited on 25-10-2014.
[12] http://www.futureworksconsulting.com/resources/TeamAgilityAgileTimesFeb04.pdf, Webpage visited on 25-10-2014
[13] The classic four stages are Forming, Storming, Norming and Performing from: Tuckman, B.W. "Developmental Sequence in Small Groups." Psychological Bulletin, 63(6). 1965

AUTHORS

Mr. P. Ashok Reddy, has received his Master's Degree from JNTU Kakinda with Computer Science & Engineering specialization and at present he pursuing Doctor of Philosophy from Acharya Nagarjuna University, AP, India in the area of Software Engineering. He has been actively involved in teaching and research for the past 9 years and now he is working as Sr. Asst. Professor in LBRCE-Mylavaram, AP, India.

Dr. K Rajasekhara Rao, has received his Doctor of Philosophy in Computer Science & Engineering from Acharya Nagarjuna University, AP, India. He has been actively involved in teaching and research for the past 30 years and now he is working as Director, Sri Prakash Educational Institutions, Tuni, AP, India. His research interests include: Embedded Systems, Software Engineering, Algorithm Complexity analysis and Parallel Computing. Three of his scholars have successfully earned their PhD degrees in Computer Science & Engineering domain and 8 more scholars are pursuing their PhD work under his guidance from various Universities.

Dr. M. Babu Reddy, has received his Master's Degree and Doctor of Philosophy from Acharya Nagarjuna University, AP, India with Computer Science & Engineering specialization. He has been actively involved in teaching and research for the past 15 years and now he is working as Asst. Professor of Computer Science, Krishna University, Machilipatnam, AP, India. His research interests include: Machine Learning, Software Engineering, Algorithm Complexity analysis and Data Mining.