JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
31st December 2017, Volume 2, Issue 3, JSEIS, CADMEI Copyright © 2016-2017
www.jseis.org

A short review of requirements gathering in agile software development

[1]Israr Ghani, [2]Muhammad Irfan Khan, [3]Adila Firdaus Arbain
[1,2]University of Arid Agriculture Rawalpindi, Pakistan
[3]Limkokwing University of Creative Technology, Cyberjaya, Malaysia
Email: [1]israrghani@outlook.com, [2]irfanpitafi@gmail.com, [3]adilafirdaus@gmail.com

ABSTRACT

Collecting, understanding, and managing requirements is a critical aspect in all the development methods. This is true for agile methods as well. In particular, several agile practices deal with requirements in order to implement them correctly and satisfy the needs of the client. These practices focus on a continuous interaction with the client, customer and end user to address the requirements evolution over time, estimate and prioritize them, and deliver the most valuable functionalities first. Understanding and satisfying every individual client necessities have been perceived as a squeezing challenge for programming enterprises. In order to produce high quality software products and meeting stakeholder's requirement is a major challenge in software requirement. Poor pre-requisites and changes to necessities are one of the causes for project overrun and quality issues in the delivered software. The research shares some of the existing studies that deal with this question "How different existing agile methodologies manage requirements and how collaboration issue affects these steps in a project?"

Keywords: requirements engineering; team collaboration; communication; agile software development; software quality; requirement management;

1. INTRODUCTION

Agile development methodology is an approach used in software development which has become more popular during the last few years due to the introduction of agile manifesto [1]. As compared with traditional software development life cycle (SDLC), effective collaboration and requirements gathering with team collaboration is a fundamental component for agile based development [2-6]. Agile software development is iterative and incremental development where you do development in small iterations. It attempts to provide many opportunities to assess the different stages of a project throughout the software development life cycle. There are a number of methods used by agile teams such as Scrum, eXtreme Programming (XP), Feature Driven Development (FDD), Dynamic System Development Method (DSDM), Kanban, Lean and Crystal. These methods aim to deliver software faster and ensure that the software meets customer's changing needs and expectations. Agile methodologies focus on skills, communication and community clarifying the roles of customers, managers and developers for a more satisfying and productive relationship. Requirements gathering & communication is accepted as one of the most crucial stages in software engineering as it addresses the critical problem of designing the right software for the stakeholder. Agile development works in a different manner than tradition methods [2, 7]. Instead of specifying everything before you start developing, one need to take a small portion of the most important features and only specify and implement those. Requirement gathering is different for different methods [8, 9], but user story method is the most popular one used by many agile teams as shown in the Figure 1 below [10].

Figure. 1 User story in agile development [10]

Requirement specification is an iterative process that continues until the customer is satisfied with the product. By using this method, it is easier to react to changes and can have a better estimate and better control on the development leading towards a quality product and customer satisfaction. There are a number of agile methods that offer the ways to collect and analyse requirements. Some of these are discussed in the following section.

2. REQUIREMENTS GATHERING IN AGILE METHODS

This section discussed four most popular agile methods that offer requirements gathering.

2.1 Scrum and requirements gathering

Scrum is an agile approach to software development. It has been found out that Scrum is very beneficial when applied to small and medium projects. Rather than a full process or methodology, it is a framework which instead of providing complete, detailed descriptions of how everything is to be done on the project, much is left up to the team, because the team will know best how to solve its problem. For example, in a sprint planning meeting the discussion about requirements is in terms of the desired outcome (a commitment to set of features to be developed in the next sprint). Scrum relies on a self-organizing, cross-functional team. There is no team leader in a scrum team who decides which person will do what. The development begins with the discussion about requirements and user stories by the product owner. Based on this the developers try to have a clear idea of what to develop. It means product owner gathers the requirements and analysis is performed with the team during the project inception, sprint planning and review meetings. Still there are such issues?

2.2 XP and requirements gathering

One objective for the agile methods is to lower the cost of changing requirements. Requirement gathering in XP addresses this issue by simplifying management tasks and documentation while the traditional software engineering places more emphasis on strict control and extensive documentation. In order to achieve simplicity, XP uses an iterative and incremental software process and very short development cycles. Here are XP practices namely:

- planning game
- small releases
- metaphor
- pair programming
- refactoring
- collective code ownership
- on site customer
- continuous testing
- simple design
- continuous integration
- and coding standard [11, 12].

2.3 FDD and requirements gathering

Discovering list of features is a critical process. The quality of this step largely defines how precise the project will be tracked, how maintainable and extensible the code will be. This process requires full-time participation of customers. FDD includes:

- the principle of least privilege
- the principle of failing securely
- the principle of securing the weakest link
- the principle of defence in depth
- the principle of separation of privilege
- the principle of economy of mechanism
- the principle of least common mechanism
- and the principle of complete mediation [13, 14].

2.4 DSDM and requirements gathering

Agile methods can be beneficial when combined with distributed development. This helped in reducing the cost involved and access to skilled resources. The main objective is to develop high quality products at lower cost than co-located developments by optimizing the resources sometimes, [15-17] the search for competitive advantage forces organizations to search for external solutions. Project and process management issues as high organizational complexity, scheduling, task assignment, and cost estimation becomes more problematic in distribute environments as a result of volatile requirements, changing specifications, cultural diversity, and lack of informal communication. Managers must control the overall development process, improving it during the enactment and minimizing any factors that may decrease productivity.

3. COMPARISON OF APPROACHES

This section summarizes the strengths and weaknesses of existing agile approaches in relation to requirements gathering. Table 1 highlights the comparison of different approaches in the domain of agile software development.

Table. 1 Comparison of approaches

Approach	Strengths	Limitations
[11] Online serious games in distributed requirements gathering.	1. Enhances individuals' creativity, and therefore customers are able to provide more innovative ideas about the software to be developed. 2. Rich ICT mediated tool facilitates collaboration and off-site communication between software stakeholders, which might lead to effective requirements communication in distributed projects	1. Still need excessive involvement of different customers until an agreement is achieved.
[13] Hybrid user-requirements and interface evaluation.	Using prototypes and User interfaces to navigate the client in using the desired system and the gather the real requirements that the clients really need.	1. Users must be familiar with the prototypes navigation and that will take some effort and time. 2. Different navigation results from different types of users. Do not solve the issues of distributed stakeholders.
[15] Groom and prioritises.	1. Product backlog continuously evolving. 2. Priorities based on set of experiences that product owners have.	1. Rely on human skills and experience especially the role of product owner, not much on the set-up approaches.
[12] Method for Elicitation, Documentation and Validation of software user requirements (MEDoV)	1. Helps to prioritize the gathered requirements. 2. Requirements gathering is based on the business objectives. 3. No unwanted features will be created based on the method of requirement gathering that this approach propose.	1. Excessive formal documentation that needed in order to gather the requirements. 2. Wasted efforts in understanding the whole individual Key Performance Index (KPI) for each requirement.

3.1 Critical analysis of solutions in agile requirement gathering

Table 1 provides comparison of approaches investigated in the perspective of their strengths and weaknesses. Identification of agile approaches for requirement gathering needs the solution of problems related with the agile requirement gathering.

a) Collaboration

A number of stakeholders involve in the agile and require the collaboration on every phase of iteration. Customers collaborate with the stakeholders and discuss the concerns of every stakeholder before reaching an agreement. Communication and coordination between stakeholders create a common vision also known as collaborative activity. Thus, socio-technical issues are resolved by discussion between stakeholders [18].

b) Requirement dependencies

Requirement dependency is an issue that is not explored in the literature. Dependencies are considered to be the risk for the cost-effective execution of projects. Success and cost reduction of a project highly depend upon the identification of dependencies. Detecting the dependencies as early as possible is key for high gains. For a small sized project with the limited number of requirements, every member in a development team knows the dependencies between requirements. However, a large sized project with many stakeholders have a large number of dependencies. This is an area which requires more attention of researchers to find the solution of this issue. In [19] researchers presented their findings in an empirical study on dependencies for agile requirements. Some of the recommendations they made in the study included as that continuous collaboration and communication for an agile method were found to be critical to reduce the risk produced from requirements' dependencies.

c) User stories

User story in agile methodology promotes the collaboration between software development team and customers. Storytelling is employed where a customer tells about the capability or needs based on an acceptance criterion. Thus, storytelling is a good source to establish the link between software stakeholders. Demonstration and iterations of storytelling continue until a customer tests and accepts the requirements. Storytelling has created more understanding and clear requirements as customers are in the immediate access. Acceptance criteria is set when a story is created. This acceptance criterion provides information that when a software is completed. It also tells that when a story is added to a sprint to adjust the acceptance criterion. Acceptance criterion also includes the specific performance, usability requirements, validation requirements and metrics. Research [20] proposes to add these requirements to a story is as to define the testable and measurable criteria for customers.

In [21] it has been stated that requirement gathering processes are easier and reliable because customers meet directly with the developers in the agile software development. An individual's memory keeps the information about discussion between the customers and team developers. In a traditional requirement gathering, a developer is not directly involved in process of requirement gathering. There may arise issues of lack of complete information between customers and requirement engineers. This issue has been resolved by clarifying and evolving the requirements in the agile software development.

4. CONCLUSION

This research concentrates on the collaboration practices around requirements engineering and team collaboration within an agile time specifically to perform in depth analysis of issues related to communication in requirement gathering in agile-based software development. The most crucial skill is effective communication for agile-based requirements engineering methodology as it plays a vital role to deliver and accept information from both parties; software project team and stakeholders. This research also includes requirements envisioning, strategies, and modelling of a creative teamwork in agile-based development, the structure that the team adopts and specially the different roles that the methodology advises to define to validate the proposed model for effectiveness of communication, although communication analysis is an information system but this research defines a requirement gathering with communication method that proposes a model for effectiveness of communication strategies in agile with a flow of activities and a requirements structure. It also discusses the issues on communication in requirement gathering in agile software and disparity between the stakeholders which affects the agile-based software development project as a whole. The outcome of the requirements gathering process will be reported and analysed for future improvements and enhancements in agile-based software development.

ACKNOWLEDGEMENT

Authors gave special thanks to University of Arid Agriculture, Rawalpindi, Pakistan and Limkokwing University of Creative Technology, Cyberjaya, Malaysia for providing support and facilities to complete this research.

REFERENCES

1. Alliance, A., *Agile manifesto.* Online at http://www. agilemanifesto. org, 2001. **6**(1).
2. Mahalakshmi, M. and M. Sundararajan, *Traditional SDLC Vs Scrum Methodology–A Comparative Study.* International Journal of Emerging Technology and Advanced Engineering, 2013. **3**(6): p. 192-196.
3. Helmy, W., A. Kamel, and O. Hegazy, *Requirements engineering methodology in agile environment.* International Journal of Computer Science Issues, 2012. **9**(5): p. 293-300.
4. Vithana, V., *Scrum Requirements Engineering Practices and Challenges in Offshore Software Development.* International journal of computer applications, 2015. **116**(22).
5. GILLANI, S.M., S. QADRI, and M. FAHAD, *Customer Oriented Requirement Engineering By Using Scrum Methodology.* International Journal of Natural & Engineering Sciences, 2014. **8**(3).
6. Carlson, D. *Practical Agile Requirements Engineering*. 2010 [cited 2017 05/04/2017]; Available from: https://ndiastorage.blob.core.usgovcloudapi.net/ndia/2010/systemengr/WednesdayTrack1_11106Matuzik.pdf.
7. Elshandidy, H. and S. Mazen, *Agile and traditional requirements engineering: A survey.* International Journal Of Scientific & Engineering Research, 2013. **9**.
8. Kennaley, M., *SDLC 3.0: Beyond a Tacit Understanding of Agile, Towards the Next Generation of Software Engineering*. 2010: Fourth Medium Press.
9. De Lucia, A. and A. Qusef, *Requirements engineering in agile software development.* Journal of Emerging Technologies in Web Intelligence, 2010. **2**(3): p. 212-220.
10. FAQs, A. *Two Days Authoring User Stories Workshop*. [cited 2017 23/06/2017]; Available from: https://agilefaqs.com/services/training/user-stories.
11. Ghanbari, H., J. Similä, and J. Markkula, *Utilizing online serious games to facilitate distributed requirements elicitation.* Journal of Systems and Software, 2015. **109**: p. 32-49.
12. Dragicevic, S., S. Celar, and L. Novak. *Use of Method for Elicitation, Documentation, and Validation of Software User Requirements (MEDoV) in Agile Software Development Projects*. in *Computational Intelligence, Communication Systems and Networks (CICSyN), 2014 Sixth International Conference on*. 2014: IEEE.
13. Bias, R.G., *The HURIE Method: A Case Study Combining Requirements Gathering and User Interface Evaluation.* User-Centered Design Stories: Real-World UCD Case Studies, 2010: p. 163.
14. Anwer, F. and S. Aftab, *SXP: Simplified Extreme Programing Process Model.* International Journal of Modern Education and Computer Science (IJMECS), 2017. **9**(6): p. 25-31.
15. Bass, J.M., *How product owner teams scale agile methods to large distributed enterprises.* Empirical Software Engineering, 2015. **20**(6): p. 1525-1557.
16. Al-Zewairi, M., et al., *Agile Software Development Methodologies: Survey of Surveys.* Journal of Computer and Communications, 2017. **5**(05): p. 74.
17. Shrivastava, S.V., *Distributed agile software development: A review.* arXiv preprint arXiv:1006.1955, 2010.
18. Inayat, I., S.S. Salim, and Z.M. Kasirun. *Socio-technical aspects of requirements-driven collaboration (RDC) in agile software development methods*. in *Open Systems (ICOS), 2012 IEEE Conference on*. 2012: IEEE.
19. Martakis, A. and M. Daneva. *Handling requirements dependencies in agile projects: A focus group with agile software development practitioners*. in *Research Challenges in Information Science (RCIS), 2013 IEEE Seventh International Conference on*. 2013: IEEE.
20. Huckabee, W.A., *Requirements engineering in an agile software development environment.* Defense Acquisition Research Journal, 2015. **22**(4): p. 394-415.
21. Abdullah, N.N.B., et al. *Communication patterns of agile requirements engineering*. in *Proceedings of the 1st workshop on agile requirements engineering*. 2011: ACM.

AUTHORS PROFILE