Scrum Software Maintenance Model: Efficient Software Maintenance in Agile Methodology

Fateh ur Rehman, Bilal Maqbool, Muhammad Qasim Riaz, Usman Qamar, Muhammad Abbas
Department of Computer Engineering, College of Electrical & Mechanical Engineering, NUST, Rawalpindi, Pakistan

Agile methodologies gained fame due to the fact of producing high-quality software systems. Maintenance effort is almost more than half of the total effort invested in any software system during its lifespan. A well-discussed issue within the community of researchers and engineers is how to use agile methodologies for maintaining the developed software because agile software development life cycle doesn't have the specifically planned mechanism for maintenance. To bridge this gap, we used the theoretical and empirical technique to formulate factors that should be followed during the agile maintenance including planning for the maintenance; the on-site customer should be present, iterative maintenance, documentation update after each phase and maintenance should be testable.

Keywords – Agile, Scrum, Software Maintenance, Agile Maintenance, Scrum Maintenance, Maintenance Sprints

I. INTRODUCTION

Agile methodologies are increasingly used for the development of software projects [1]. These methodologies are flexible to accept future changes during the life of software development. Scrum [2] is a straightforward, light-weight process for managing and monitoring the progress of software development and it is influenced by the spiral model of Boehm [3]. Maintenance of software is a vital phase of the life cycle of software development, consistent with software evolution law proposed by Lehman; a software system would changed continuously or becomes less useful [4].

Software development and maintenance differ in several important respects [5]. The majority of software projects are building on existing code base [6] and interacts with some instances of the organization and in many cases hard to change too much of the process at once. This factor motivates us to introduce the iterative method of the maintenance to adopt changes in iterations. According to some research studies, software maintenance achieved several benefits using agile methodologies [7,8]. Software maintenance is categorized into four groups – adaptive, corrective, perfective and preventive. Perfective and adaptive maintenance are typically supposed as improvements [9] done to the software structure to increase the usability of the system.

We will try to find out the answer to two research questions in the paper:

- First, how does software maintenance differs with software development?
- Second, how to handle the urgent emergency customer requests during the agile maintenance sprints?

First, these answers were collected from the literature and then we identified and defined the elements that are necessary for agile maintenance process considering the urgent emergency customer needs. We validated identified factors for the maintenance of software system that was working for real customers during the time of this research.

II. BACKGROUND:

Software maintenance is a vital phase of the life cycle of software. Maintenance effort is almost greater than half of the total effort invested in any software system during its lifespan. According to the literature, software system deviates from its original state when maintenance is carried out on it [10] and continuously evolve to a new system [11]. Software maintenance lifecycle has four stages - introduction stage tells initial problems, growth stage exposes deeper level technical difficulties, maturity stage describes significant major enhancement, and during decline stage, the system is patched until it is replaced [12].

Maintenance teams don't have a specific completion date for maintenance project and contain only tasks to be performed. These tasks can overlap and continue to an infinite time [13] until the system is replaced in decline stage. Development projects typically handle the single instance of the software system during the particular period, but maintenance is more complicated by considering discrete versions of a software system and by engaging the customer. The customer initiates maintenance, and each change follows a cycle – request for a change, planning, implementation, verification, validation and re-documentation [14]. The success of software development process is measured by successful completion of scope for a specified time using estimated cost. However, the success of maintenance process can't be measured with these dimensions [13]. Software development and maintenance are different, and we can't use the same procedure for maintaining the software that is defined for the development of software. Agile methodologies share some common characteristics including development in iterations, focusing on delivery, small teams, involving the customer, direct communication, less documentation, test driven development, collective ownership, openness and quality product [15,16]. Since we claimed that development and maintenance are different processes; adopting agile tools and techniques will not necessarily give good results for maintenance [17].

Introducing agile methods for maintenance gave positive results by speed up the process of maintenance [7] and increasing the communication between maintenance team [8]. Agile maintenance also gave team motivation, user satisfaction [15] and improved code quality [7,8,18]. On the other hand, maintenance engineers don't like less documentation because they rely on documentation to understand the system [19]. They can't simplify design because original developers created the design, so all the maintenance is performed on the original design [6]. Maintenance engineers can have individual tasks on different versions of a system [14], and overhead of frequent agile testing can be reduced using automated testing [20,21].

Heeager and Rose [13] studied the challenges and revealed issues while adopting agile scrum for maintenance process. They found that more problems were related to the inconsistency of sprint goals because the emergency jobs were given by the customers. Agile considered that there is no interruption when development is carried out in some iteration [22], but we can't always avoid interruptions [23,24]. During the maintenance, the client can interrupt any time for some urgent emergency work [14], but it's hard to redo the prioritization of interruptions during the specific iteration [20]. We can define two different types of sprints for maintenance process – unplanned or unexpected short sprints and planned or regular long sprints [18]. It is found that customer also sends the urgent emergency request during short or unplanned sprint so these interruptions should be well managed and tackled to evade hindrance and demoralization within teams and achieving goals of the sprint [13].

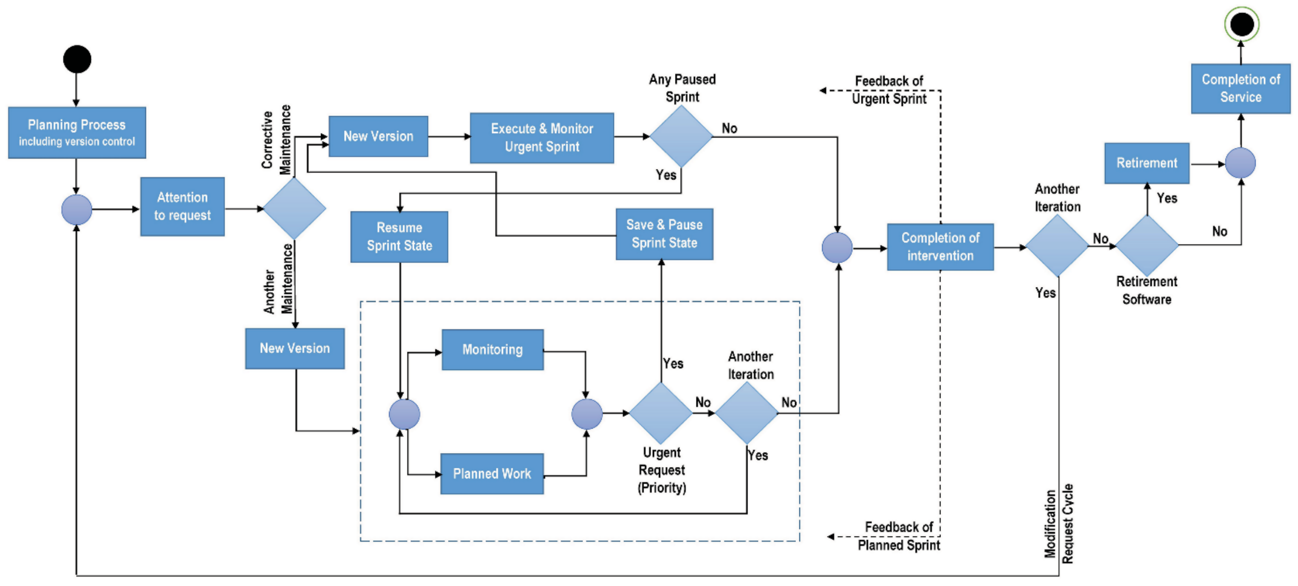
III. STUDY

We used the theoretical and empirical technique to find answers to our research questions. We found that software development community widely adopted agile methodologies and producing high-quality software systems [1]. Due to the fame of agile methods, maintenance teams started practising agile maintenance, but this adoption reveals some issues because software maintenance and development have different characteristics. According to literature, maintenance and development are distinct processes and maintenance team faces challenges while working with agile methodologies [13]. It gives the answer to our first research question – how does software maintenance differ with software development? It also gives strength to our next research question – how to handle the urgent emergency customer requests during the agile maintenance sprint?

3.1 THEORETICAL STUDY

Heeager and Rose [13] conducted a study on agile maintenance and found major problems related to missed sprint goals. They found that customer requested emergency work during the sprints and team lacked planning for this emergency work. Pino et al. [18] advised creating two types of sprints for maintenance - a brief one for unforeseen needs and an extended one for planned tasks. Heeager and Rose [13] determined that client requests came even during the short sprints. They said to create a buffer [22] – some extra time that is not allocated to any task and can be used later for unexpected tasks, but it's hard to predict the size of this buffer. They also suggested creating a well-managed mechanism to avoid the disturbance of sprint goals and to handle urgent emergency customer requests.

We addressed this problem by introducing Scrum Software Maintenance Model to handle urgent emergency customer requests in parallel with regular sprints. Our model is inspired by the findings of Pino et al. [18] and Heeager and Rose [13]



Scrum Software Maintenance Model

Scrum Software Maintenance Model starts with the planning process containing version control for maintenance change tracking. After planning special attention is made to check the type of maintenance request. Corrective maintenance is considered as priority and working is started on it by creating new code branch. Non-corrective maintenance requests are planned normally and a special monitoring is done to check for urgent and emergency sprints during their execution. If some urgent request received, current sprint is paused, its work is stored in version control system and the new version is started for the urgent request. After the completion of the urgent request, this model checks for paused requests and resumes them from where these were left by restoring source from version control.

This model introduced a concept of saving sprint state using version control just to work on urgent emergency sprints those are more important for client's business. It emphasizes on following points:

- **Planning for the maintenance:** Plan the maintenance work into the sprints considering the critical emergency and regular work. It is not necessary to make all the iterations of the same length, depending upon the nature of work in maintenance, small iterations should be allowed. Control each iteration with version control (like Git) to make the management easier and traceable while keeping a full audit path [25].
- **The on-site customer should be present:** Involve customer in sprints planning to get the input about planned and unplanned emergency tasks and to ask about the priority of each task to update the sprint plan.
- **Perform each maintenance task in the iteration:** Either it is a regular task or urgent emergency task, do it in separate iteration and after each stable iteration merges your work with the original system.
- **Documentation update after each phase:** Document the iteration to maintain the integrity of the system. Structured use case [13] or structured test case can be used as a document.
- **Maintenance should be testable:** Make the maintenance work more testable by doing each iteration in separate version and controlling it with some version control. It will allow the precise integration and regression testing and will also enable parallel iteration for urgent emergency tasks.

3.2 CASE STUDY

We picked a case study for the maintenance of a fitness system that was working in production. This system had fitness trainers and their clients; trainers were virtually training their clients using the training plans. The maintenance team was located in South Asia and customer was in the USA and communication was carried out using online communication tools. The arrival of the Internet, electronic mails, and little price global communications has simplified the progress of digital groups, remote work agencies, and remote corporations [26]. The maintenance team was using agile scrum maintenance practices and was facing following issues:

- Urgent emergency sprints were overriding the current sprints, and sprint goals were not meeting.
- Rework required for the sprints that were replaced by urgent emergency sprints because there was no way to save the state of current sprint.
- The complete system's testing naturally impractical [27], and tracking error was confusing because there was no specific code version for the specific iteration.

The derived Scrum Software Maintenance Model was implemented for the maintenance of above system, and maintenance tasks were planned in iterations, having the particular version in version control system. The client was involved in the meetings to decide the priority of unplanned emergency work and to update the overall sprint plan. The analysis covered three months of maintenance work and included six planned and four unplanned sprints. In the start, six sprints were planned by adding buffers [22] in the sprints. The client requested emergency maintenance four times during the three months' analysis. Each time client was involved in revising the original sprints plan, and maintenance engineers switched to emergency sprints by saving their work in the separate version using version control system. After completing emergency maintenance; engineers turned back to the leftover sprint; buffers plus approved additional time helped them to meet the time goals. During this process, each iteration was documented in use cases and test cases format.

IV. FINDINGS

The analysis was done by conducting interviews with scrum master, team members and author also participated in the scrum meetings during which maintenance team discussed and planned the emergency tasks with the client. Maintenance team achieved following benefits using Scrum Software Maintenance Model:

- Client satisfaction increased by allowing small iterations for urgent emergency work, involving her in update sprints planning and getting priority decisions on sprints from her.
- Maintenance engineers' morale improved by reducing the rework and enabling them to save the sprints state in version control and starting the sprint of urgent emergency work. After completing the emergency work, engineers switched back to the regular sprint and started the first sprint from the saved state.
- Separate version for each sprint made integration testing easy and enabled testers to devise and perform the test cases according to the integration of specific iteration. Version control system also increased the errors traceability and audit due to inherited benefits of versioning for agile scrum maintenance – versioning could be done swiftly and proficiently, preserves reliability and trust, imposes responsibility and gives clean core software design [28].
- Simplified minimal documentation in the form of use cases and test cases increased the software system maintainability and testability.

V. CONCLUSION

Software development community widely adopted agile methodologies and producing high-quality software systems. Due to the fame of agile methods; maintenance teams started practising agile maintenance, but maintenance teams face challenges while working with agile methodologies because maintenance and development are different processes. Previous researchers studied agile maintenance issues and identified that major issues were related to not meeting sprint goals because of urgent emergency client requests. We proposed Scrum Software Maintenance Model to handle urgent emergency sprints during the planned sprints based on proper planning, client involvement, iteration handling with version control, testing and documentation. Implementation of Scrum Software Maintenance Model increased the client satisfaction because maintenance team was listening client for urgent emergency tasks and the client was involved in sprint planning and sprint prioritization. Maintenance engineers' satisfaction increased by reducing rework with saved sprint state using version control. Version control also increased the errors traceability, audit and imposed responsibility and clean core software design. Documentation in the form of use cases and test cases increased the software system maintainability and testability. Hence we achieved the maintenance goals in parallel to agile scrum goals.

VI. REFERENCES

[1] Sidky A, Arthur J, Bohner S (2007) A disciplined approach to adopting agile practices: the agile adoption framework. Innov Syst Softw Eng 3(3):203–216

[2] Schmaber, K. and Beedle, J. Agile SoftivareDeveloprlleilt with Scrum, Upper Saddle River. NJ: Prentlce Hall. 2002.

[3] Boehni, B. "A Spiral Model of Software Development and Maintenance," IEEE Computer (21:s); 1988, pp. 61-72.

[4] Lehman MM (1980) on understanding laws, evolution, and conservation in the large-program life cycle. J Syst Softw 1:213–221

[5] Charette RN, Adams KM, White MB (1997) Managing risk in software maintenance. IEEE Softw 14(3):43–50

[6] Svensson H, Host M (2005) Introducing an agile process in a software maintenance and evolution organization. In: Ninth European Conference on Software Maintenance and Reengineering, Manchester, United Kingdom. IEEE, pp 256–264

[7] Choudhari J, Suman U (2010) Iterative Maintenance Life Cycle Using eXtreme Programming. In: International Conference on Advances in Recent Technologies in Communication and Computing (ARTCom 2010). IEEE, pp 401–403

[8] Rudzki J, Hammouda I, Mikkola T (2009) Agile Experiences in a Software Service Company. In: 35th Euromicro Conference on Software Engineering and Advanced Applications, SEAA'09. IEEE, pp 224–228

[9] Pigoski, T. M., “Practical Software Maintenance – Best Practices for Managing Your Software Investment”, John Wiley & Sons, New York, NY, 1997

[10] Stachour P, Collier-Brown D (2009) You don’t know jack about software maintenance. Commun ACM 52(11):54–58

[11] Sukumaran S, Sreenivas A (2005) Identifying test conditions for software maintenance. Ninth European conference on software maintenance and re-engineering. IEEE Computer Society, Manchester. doi:10.1109/CSMR.2005.32

[12] Kung H, Hsu C (1998) Software maintenance life cycle model. International Conference on Software Maintenance. IEEE Computer Society, Bethesda

[13] Heeager, L.T. & Rose, J. Empir Software Eng (2015) 20: 1762. doi:10.1007/s10664-014-9335-7

[14] Bennett KH, Rajlich VT (2000) Software maintenance and evolution: a roadmap. In: Proceedings of the Conference on the Future of Software Engineering. ACM, pp 73–87

[15] Prochazka J (2011) Agile Support and Maintenance of IT Services. In: Information Systems Development, Prague, Czech Republic. Springer, pp 597–609. doi: 10.1007/978-1-4419-9790-6_48

[16] Abrahamsson P, Salo O, Ronkainen J, Warsta J (2002) Agile software development methods: review and analysis. VTT, Finland

[17] Polo M, Piattini M, Ruiz F (2002) Using a qualitative research method for building a software maintenance methodology. Softw Pract Experience 32(13):1239–1260

[18] Pino FJ, Ruiz F, Garcia F, Piattini M (2012) A software maintenance methodology for small organizations: Agile_MANTEMA. J Softw Evol and Process 24(8):851–876

[19] de Souza SCB, Anquetil N, de Oliveira KM (2005) A study of the documentation essential to software maintenance. In: Proceedings of the 23rd annual international conference on Design of communication: documenting & designing for pervasive information. ACM, pp 68–75

[20] Poole C, Huisman JW (2001) Using extreme programming in a maintenance environment. IEEE Softw 18(6):42–50

[21] Poole CJ, Murphy T, Huisman JW, Higgins A (2001) Extreme maintenance. In: Software Maintenance. Proceedings. IEEE International Conference on, Florence, Italy. IEEE, pp 301–309

[22] Schwaber K, Beedle M (2001) Agile software development with Scrum. Prentice Hall, Upper Saddle River

[23] Heeager LT, Nielsen PA (2009) Agile Software Development and its Compatibility with a Document-Driven Approach? A Case Study. In: Scheepers H DM (ed) Australasian Conference on Information Systems, Melbourne, Australien, 2009. p 205

[24] Pikkarainen M, Haikara J, Salo O, Abrahamsson P, Still J (2008) The impact of agile practices on communication in software development. Empir Softw Eng 13(3):303–337

[25] Ram, K. Source Code Biol Med (2013) 8: 7. doi:10.1186/1751-0473-8-7

[26] O’Brien JA (2002) Management information systems managing information technology in the

business enterprise, 6th edn. McGraw Hill, New York

[27] Junio GA, Malta MN, de Almeida Mossri H, Marques-Neto HT, Valente MT (2011) On the benefits of planning and grouping software maintenance requests. In: 15th European Conference on Software Maintenance and Reengineering (CSMR), Oldenburg. IEEE, pp 55–64

[28] Jon Loeliger (2009) Version control with Git,1st edn. O'Reilly Media, Inc, USA, ISBN: 978-0-596-52012-0