Implementation of Change Management in Software Development by using Scrum Framework

Rupali Pravinkumar Pawar*  
Research Scholar, Department of Computer Application,  
Bharati Vidyapeeth University, Pune, Maharashtra, India

Dr. Kirti Nilesh Mahajan  
Department of Computer Application,  
Bharati Vidyapeeth University, Pune, Maharashtra, India

DOI: 10.23956/ijarcsse/V7I7/0194

Abstract: This paper will focus on implementation of change management in Scrum software development framework. Scrum is one of the mostly used software development framework from the Agile software development methodology. Scrum is based on iterative and incremental process. It is suitable for unstable requirements. The use of Scrum proved to be beneficial due to tight schedule and loosely defined user requirements that often changed during the development. The aim of the paper to study implementation process of change management in Scrum. First part of paper gives detailed information of Scrum framework. The middle of the paper presented the organizational process of agile software development using Scrum. Finally, the paper point out key points for managing changes in Scrum implementation. The primary data collection method was interviews of the industry expertise. The secondary source of data is reference books and Internet articles. This paper will help to understand basics of Scrum software development framework and process of change management in developing projects by using Scrum framework.

Keywords: Scrum framework, agile software development, change management, iterative development.

I. INTRODUCTION

There are different ways to implement the software with the help of policies, procedures and processes which is called software development methodology (SDM). There are various software development methodologies and each methodology has its own software development life cycle (SDLC) and each methodology is based on its own policies, procedures and processes. One of well-known traditional software development methods is the waterfall model. The TSDMs are still widely used in industry because of their straightforward, methodical, and structured nature, as well as their predictability, stability, and high assurance.

Scrum is the mostly preferred Agile software development framework. Welcoming Changes is the second principle in the agile manifesto. Within the Agile world scope change is expected and time is considered more important than functionality. So if something has to give to allow change then functionality/scope loses and time wins. To do this the customer must make Requirements Trade-offs. The Customer directs development in Timebox and minor changes are just accepted. Big changes are handled different depending on whether it is a change to the Release Plan or Timebox Plan.

II. SCRUM FRAMEWORK

The Scrum framework is originally proposed by Schwaber and Beedle (2001). Scrum is an iterative, time boxing and lightweight process framework for software development. “Lightweight” means that the overhead of the process is kept as small as possible, to maximize the amount of productive time available for getting useful work done.
A. Roles in Scrum:

There are three distinct roles in the Scrum process: the Product Owner, the Team and the Scrum Master.

1) Product Owner:

The Product Owner is responsible for getting initial and on-going funding for the project by creating the project’s overall requirements, return on investment (ROI) objectives, and release plan. He maintains the Product Backlog (which is the repository for all of this information), keeping it up to date and at the level of detail and quality the Team requires. The Product Owner also sets the schedule for releasing completed work to customers, and makes the final call as to whether implementations have the features and quality required for release.

2) Team:

The Team is responsible for implementing the functionality described in the requirements. Teams should be self-managing, self-organizing, and cross functional to maximize team performance. All of the team members are responsible for both the success and the failure of sub-systems and of entire systems.

3) ScrumMaster:

ScrumMaster is responsible for making the process run smoothly, for removing obstacles that impact productivity, and for organizing and facilitating the critical meetings. The ScrumMaster responsibilities include:

1. Removing the barriers between the development Team and the Product Owner so that the Product Owner directly drives development.
2. Teach the Product Owner how to maximize return on investment (ROI), and meet his/her objectives through Scrum.
3. Improve the lives of the development Team by facilitating creativity and empowerment.
4. Improve the productivity of the development Team in any way possible.
5. Improve the engineering practices and tools so that each increment of functionality is potentially shippable.
6. Keep information about the Team’s progress up to date and visible to all parties.

B. Ceremonies in Scrum:

Scrum framework supports four ceremonies: the sprint planning meeting, Daily Scrum, sprint review meeting, and sprint retrospective meeting.

1) Sprint Planning meeting:

The work to be performed in the Sprint is planned at the Sprint Planning meeting. This plan is created by the collaborative work of the entire Scrum Team including all roles. Sprint Planning is time-boxed to a maximum of eight hours for a one-month Sprint. For shorter Sprints, the event is usually shorter. Items that will be delivered during the scrum sprint are acquired from the Product Backlog based on their priority value. During scrum sprint planning, Scrum Team agrees on a scrum sprint goal that defines what the upcoming sprint is supposed to achieve.

2) Daily Scrum:

Each day at the same time, the team meets so as to bring everyone up to date on the information that is vital for coordination: each team members briefly describes any “completed” contributions and any obstacles that stand in their way. Usually, Scrum’s Three Questions are used to structure discussion. The meeting is normally held in front of the task board. This meeting is normally time-boxed to a maximum duration of 15 minutes.

3) Sprint Review/Demonstration Meeting

Sprint review is a time to showcase the work of the team. The team demonstrates the work finished by them during the sprint and get feedback from the product owner and project stakeholders. The work should be fully demonstrable and meet the quality aspects to be considered as complete.

4) Sprint Retrospective Meeting

After the sprint review meeting, the entire team gets together for the retrospective meeting. During this meeting, the team inspects and adapts its working process. During the meeting team members speak frankly about what occurred during the Sprint and how they felt about it. After all team members thoroughly understand each other, they work to identify what they’d like to do differently the next Sprint, typically focusing only on one or two specific areas of improvement each Sprint. The Scrum Master may also observe common impediments that impact the team and then work to resolve them.

C. Scrum Artifacts:

Scrum Artifacts provide key information that the Scrum Team and the stakeholders need to be aware of for understanding the product under development, the activities done, and the activities being planned in the project. The following artifacts are defined in Scrum Process Framework -
1) Product Backlog
The Product Backlog is an ordered list of features that are needed as part of the end product and it is the single source of requirements for any changes to be made to the product.

The Product Backlog lists all features, functions, requirements, enhancements, and fixes that constitute the changes to be made to the product in future releases. Product Backlog items have the attributes of a description, order, estimate, and value. These items are normally termed as User Stories. The Product Owner is responsible for the Product Backlog, including its content, availability, and priorities of items.

2) Sprint Backlog
The Sprint Backlog is the set of Stories planned for implementation in a Sprint (the “Sprint” being the standard 2—4 week development period). The Sprint Backlog is the set of Product Backlog items selected for the Sprint, plus a plan for delivering the product Increment and realizing the Sprint Goal.

The Sprint Backlog is a forecast by the Team about what functionality will be made available in the next Increment and the work needed to deliver that functionality as a working product Increment.

3) Sprint Burn-Down Chart
At the start of the Sprint, the team breaks down each Story into a set of tasks, with a time estimate for each, such that executing the tasks results in a completed Story. During the Sprint, the team member who completes a particular task marks that task as complete. Plotting the amount of uncompleted work against time from the start of the Sprint produces a Burndown chart.

The diagonal line from the top left to lower right shows the ideal “burn down” of work versus time, and ends with zero remaining work on the last day of the Sprint. The blue bars in the sample chart show the amount of work actually remaining each day.

4) Increment
The Increment is the sum of all the Product Backlog items completed during a Sprint combined with the increments of all previous Sprints. At the end of a Sprint, the new Increment must be a working product, which means it must be in a useable condition. It must be in working condition regardless of whether the Product Owner decides to actually release it.

III. PROCESS OF CHANGE MANAGEMENT IN SCRUM
In scrum, if project requirements are changed throughout the project due to change in some business requirements, the same method continues to be effective. Before beginning a Sprint— the product owner create Prioritized Product Backlog or Groom Prioritized Product Backlog processes and assigns the highest priority requirements based on urgency of requirements in the Prioritized Product Backlog with are typically selected to be completed in that Sprint. Because changes have been accounted for in the Prioritized Product Backlog, the team only needs to determine how many tasks they can accomplish in the Sprint based on time and resources provided. Change management is executed in the ongoing processes of prioritizing and adding tasks to the Prioritized Product Backlog. If there is a Change Request that may have significant impact on a Sprint in progress, the Product Owner, after consultation with relevant stakeholders, decides whether the change can wait until the next Sprint or represents an urgent situation which may require ending the current Sprint and starting a new one. Scrum framework clearly specifies that the scope of a Sprint cannot be changed once the Sprint begins. If the required change is so important that the results of the Sprint would be worthless without it, then the Sprint should be terminated. If not, then the change is incorporated into a next Sprint.

There is only one exception to this rule about not changing the scope of a Sprint once a Sprint begins. If the Scrum Team determines it has heavily overestimated the effort during the Sprint and has spare capacity to implement additional User Stories, the team can ask the Product Owner which additional User Stories should be included in the current Sprint. By locking down the scope of every Sprint, the team is able to efficiently optimize and manage their work and effort.

An additional benefit is that the team does not have to worry about managing changes once they start working on a Sprint. This is a big advantage of the Scrum framework when compared to traditional project management. Because changes are not allowed during a Sprint, the impact and frequency of expected changes may have an impact on the decision related to the length of the Sprint when it is determined during the Conduct Release Planning process.

If project requirements are generally stable and major changes are not expected in the near future, then length of a Sprint may be set to be longer, 4 to 6 weeks. Team members to work on the Prioritized Product Backlog requirements for lengthy periods of time without having to go through the Create User Stories, Approve, Estimate and Commit User Stories, Create Tasks, Estimate Task, and other related processes that are conducted for every Sprint.

However, if project requirements are not very well defined or if significant changes are expected in the immediate future, the Length of Sprint may be relatively shorter, 1 to 3 weeks. This provides stability to the Scrum Team members to work on shorter Sprints and deliver results, which can be evaluated by the Product Owner and stakeholders at the end of the Sprint. This also provides enough flexibility for them to clarify requirements and make changes to the Prioritized Product Backlog at the end of each Sprint.
IV. KEY POINTS FOR MANAGING CHANGES IN SCRUM

1. Change management does not mean no prioritization

If change is required then product owner assigns priority to that changes and add that changes in the product backlog. Based on the highest priority of product, product selected for sprint. It means in change management changes are assigned priority and added into product backlog so that changes are selected for next sprint.

2. The goal of Agile is to enable change management, not prevent it

The goal of Agile manifest is able to handle changes if required ,this goal is based on principal of agile is "Welcoming changes". Scrum framework follows this principal. In scrum project development, if changes are required that would be managed but can't prevented.

3. There should be no change management during the sprint

Change management is to be avoided in the middle of an sprint, since scrum may have preceded the implementation in that iteration. In such cases, changes themselves need to be scheduled as stories or items for the next sprint. They are best avoided during an iteration but can be carefully considered at the beginning of next sprint.
4. Change management does not change highest value story scheduling

In scrum project development always includes the highest value (to the product owner or end user) stories in the earlier iterations or sprint. This can also be for technically challenging stories (or those with high development and technically feasibility risks) to be attempted for earlier iterations. If the changes carry high development risks, they need to be addressed in earlier sprint also. The product owner gives the priorities to products by considering urgency or high development risk.

5. Agile does not eliminate change management challenges but provides a disciplined, streamlined way to manage them

Change management challenges are not eliminated when using scrum framework. They may be necessitated by the external environment, not something within the development team. Scrum provides a disciplined, streamlined way to manage them. Agile iterations provide working software earlier, enabling owners and users to recognize and address needed changes earlier. Change management in non-Agile environments assumes that changes come at the end of the development cycle and at most can be minor deviations from the original set of requirements. Scrum gives you a way of recognizing that change management is inevitable and to facilitate its incorporation in a natural and non-intrusive way.

V. CONCLUSION

Change management has always been a challenge in software development, whether you use scrum or not. If changes are needed, in scrum, they can be recognized earlier and interleaved with earlier sprint. There should be no change management during an iteration. This also provides a smooth way of development of the project and it minimizes the risks earlier in the development cycle. Rapid change management is the rationale for using Agile methods and can add significant value to resulting software.

In scrum change management is performed within regular activities some minor changes in process. It handles change management in a better way if above change management's key points will be considered. Scrum framework is suitable for unstable software projects. In scrum, change management is smoothly handled as compared to other software development methods.

REFERENCES