An Experimental Study on Requirement Gathering Methods of Agile Methodology

Mani Sahore, Harikesh Bahadur Yadav

Abstract – Agile methods form an alternative to waterfall methodologies. Agile software development method is an iterative mechanism for developing a software. Agile methods focus on accommodating change even late in the development lifecycle. Agile teams commonly use User Stories, Use Cases, interaction with On-Site Customers to provide more detail to the Requirements. This paper investigates the fact that how requirements gathering using Use Cases, User Stories and, interaction with On-Site Customers are beneficial for developers and how it increases customer satisfaction Also this paper presents the result that what are the impact of the methods like Use Cases, User Stories, interaction with On-Site Customers on the requirements gathering that enhances the customer satisfaction while using agile methodology. We found that subjects using Use Cases spent less time understanding requirements in comparison to subjects not using Use Cases. We conclude that the involvement of Use Cases in gathering requirements could benefit agile teams.

Keywords – Agile Practices, Customer Satisfaction, On-Site Customers, Rational Unified Process, Requirement Gathering, User Stories, Use Cases

1 INTRODUCTION

Currently, the number of organizations adopting agile practices is increasing .The main reasons for the adoption of agile practices are (1) Regular adaptation to changing circumstances (2) Enhanced customer relationships, (3) Greater return on investment (4) Shorter development periods.5)Rapid delivery of software. 6) Iterative development.[1].Now the question arises how agile methodology achieves these benefits. The reasons are it "lightens" the entire software development process in terms of the quantity of generated intermediate software products, models, documentation and similar things. It focuses primarily on principles, practices, and direct personal communication. These factors increase satisfaction among stakeholders and end users. One of the most indicative differences between agile and traditional software development approaches for the project implementation are a way to manage requirements. The agile methodology put emphasis on personal communication, which is very efficient, and the simple techniques for recording requirements to writing documents. The techniques are use cases, user stories described later in this paper. The aim of the development team is not to write a perfect requirement specification document, but to deliver functional software on time and in an effective way that meets the needs of the customer.

Mani Sahore has done B-tech/M-tech(CSE) from Lovely Professional University, India. Email: sahoremani6@gmail.com

Harikesh Bahadur Yadav is Assistant Professor in Lovely Professional University, India. E-mail: Harikesh.14573@lpu.co.in

But in traditional approaches there were requirements specification which made them "heavy" processes. Another beneficial point about agile methodology which proves that it is better from traditional software development methodologies is that it put emphasis on the development of intermediate products. And then it studies the feedback on the intermediate product received from the client [2].

We want to investigate the impact of Use Cases [3], User Stories [4], chatting with On-Site Customers [5] on requirements gathering in agile methodology. With this goal, we conducted a controlled experiment involving a small number of subjects. In the experiment, subjects had to make modifications to an existing software system with the requirements for the changes specified as Use Cases, User Stories and access to an On-Site Customer, or both. For each session, we recorded screen interactions, videos of subjects as they worked and audio of think-aloud verbal protocols. We also record and save chat transcripts for those subjects who had access to the On-Site Customer.

2 HISTORY

In this section, we review the concept of use cases, user stories and let's discuss the strengths and weaknesses and then their impact on agile practices.

2.1 User stories

User Stories have been gaining popularity in industry. User stories proves to be quick way of handling customer requirements without the need of creating formalized requirement .User stories basically includes a description of features from customer's point of view. User stories are 2-3

sentences long. User stories provide greater flexibility by adding use cases, Graphical user interface sketches.

User stories are often written on index cards or sticky notes, and arranged on walls or tables which facilitate planning and discussion. User stories strongly shift the focus from writing about features to discussing them. In fact, verbal discussions are more important than the written text .

User Stories basically consist of three aspects:

a) A written description of the feature
b) Conversations about the story
c) Tests cases that convey the details

Typically, User Stories are written on 3"x5" index cards. User Stories are used by agile teams and especially in Extreme Programming (XP)[6].It is one of the agile software development methods. XP is based on 12 practices i.e. Planning Game, the On-Site Customer etc. Basically what happened during the Planning Game is customers write what functionality they want from the system to perform in the form of User Stories. Then, developers estimate the time for the implementation of that User Stories. Based upon the estimates the customers prioritize the User Stories and choose which User Story will be first completed in an upcoming iteration. For their implementation, they will be assigned to a developer or a pair. During implementation, developers are expected to have questions regarding the User Stories. They give answers to them by talking face-to-face with the On-Site Customer who works side by side with the development team. With the help of Test-Driven Development, implementation in XP is done. In test driven development automated test cases are created. The test cases are based on user stories before the source code is written. Developers implement necessary code to pass the tests. User Stories should be written without using any technical words .They should be easily understandable by the business people. Their content should fit on an index card. It should be possible to explain them in 30 seconds .The completion of user stories will be done in less than one week. They should be easy to translate into a test.

One example of a User Story: "As a user, I can saw a passport size photograph in the online address book of the person whom I looking for." As we can see in the example, there are no details about validations, error messages. However, subjects who were given User Stories also had access to an On-Site customer via chat. With the help of chat they could ask any questions related to the features requested.

2.2 Use Cases

Introduction of use cases in support of object-oriented software engineering have an explosive growth in both development methods and development practice. Use cases have proved to be versatile conceptual tools for many aspects of design and development. Use Cases are used extensively in plan-based software processes, such as the Rational Unified Process [7]. Use cases includes list of steps, which defines interactions between an actor and a system.

Each use case captures: The actor, the interaction (what does the user want to do?), the goal (what is the user's goal?).

Actor: An actor might be a person, a company or organization, a computer program, or a Computer system — hardware, software, or both. Actors are always stakeholders, but many stakeholders are not actors, since they "never interact directly with the system.

Goal: Action that actor wants to accomplish.

What it tells: Use cases are the ways in which a system can be used (the functions which the system provides to its users).

Benefits**:** It includes many benefits with it. Use cases help us discover/document requirements. It provides shortest summary of what system will contribute. It includes list of things programmers have to watch for. They hold functional requirements in easy-to-read text format. They make a good framework for non-functional; requirements & project scheduling. A good use case involves as many things as possible. It starts with a request from an actor to the system. It defines the interactions (between system and actors) related to the function .It takes into account the actor's point of view, not the system's. It focuses on interaction, not internal system activities. It is easy to read. A Use Case contains the name, goal, preconditions, success end condition, failed end condition, primary and secondary actors, trigger, description of each step. In this case, the Use Case has specific information about the steps the user should follow to successfully use the new feature ("Click to see").

Also it included the details such as the names of buttons. There are basically two types of Use Cases: Informal use case and Formal use case. Formal use cases can also be written as a table.

Table 1 illustrates the example of formal use case which corresponds to the User Story elaborated in the previous sub-section.

2.3 Comparison

Table 2 explains the difference between Use case and User Stories: Use Cases are longer than User Stories; they can vary between two paragraphs and ten pages. Usually, User Stories will not be sufficient in an organization where formal documentation is a necessity. Their main difference with Use Cases or Scenarios is that User Stories have the goal of capturing the perspective that the user has about the

system. Now we have a brief comparison between the Use cases and User Stories.

TABLE 1
USE CASE EXAMPLE

USE CASE 1	Add item	
Goal in Context	Add passport size photograph in the address book of every person.	
Preconditions	User has pressed the "click to see" button for the photograph.	
Success End Condition	User successfully sees the picture of the person (he/she searches for) in the address book.	
Failed End Condition	The user could not see the picture of the person he searches.	
Primary, Secondary Actors	User	
Trigger	"Click to see" button is clicked	
	1.	Display table with columns: Name, Address, Picture ,etc.
	2.	"Click to see" button is pressed.
	3.	Use "Use Case"
	4.	View Address

TABLE 2
COMPARISON

Use Cases	User Stories
expressed using a constrained (semi-formal) syntax	expressed using natural language prose
specifications of object interactions	descriptive and expressive of human desires
Deal with "how"	They contain "what" and "why"
Longer than user stories	Shorter than use cases
Provides detail	Not enough detail
Complex to write and to understand, for both end users and developers.	easy for users to read
Use Cases are needed for implementation when formal documentation is required.	User Stories will not be sufficient in an organization where formal documentation is mandatory.
Use cases are about the behavior you'll build into the software to meet those needs	User stories are about needs.
A use case covers a much larger scope	Smaller in scope
Use cases are often permanent artifacts that continue to exist as long as the product is under active development/maintenance	Not possible in the case of user stories. Discarded after use.
generally written as the result of an analysis activity.	written as notes that can be used to initiate analysis conversations.

3 EXPERIMENT

We conducted an experiment with a small sample of software engineers to know whether or not the Uses Cases, User stories and requirements elicitation by chatting with On-Site Customers could be beneficial for teams using Agile methodology. We asked our subjects to modify an existing feature and to add two new features to "An Online Address Book" web-based recruitment system for users who want to search the address of their friends, their dear ones.

3.1 Design

We had basically three conditions in our experiment.
In the first condition, our subjects were given requirements as Use Cases. We will refer that case as the UC Group. In the second condition, our subjects were given requirements as user stories. This condition will be referred as the US Group. In the second condition, our subjects were given requirements as User Stories and On-Site Customers. This condition will be referred as the US&OC Group At last, subjects in the fourth condition were using both of the above requirement formats. This last condition will be referred as UC+US &OC Group from here onwards. We expected that subjects using the Use Cases (UC+US&OC Group) would perform the best among the four groups. We believed that because those subjects would have more information and more details. Also it would result in a

better understanding of the requirements which results in better performance. We also felt that subjects who spent more time trying to understand the requirements would perform better than the others. Because once requirements are clear and easy to understand, rest work related to development becomes easier .This would include reading the requirements from the Use Cases or User Stories or eliciting details regarding the requirements from the On-Site Customer.

3.2 Activities

The experiment consisted of four activities: a questionnaire, tutorials tasks, the maintenance tasks and the design, and finally the debriefing interview (report includes the completion of the tasks). Only the maintenance tasks and the design were time taken. Table 3 shows the schedule of the experiment including time duration per activity. Graphical representation of the overall time duration of all the activities are as drawn below.

Questionnaire: In this task **w**e asked our subjects to fill out a questionnaire based on their

- Education
- Software development experience
- Familiarity and preferences of different requirement formats.

Tutorials: The goal of these tutorials was to familiarize our subjects with the requirements format. Hence, we provided a Use Case tutorial to the UC Group, a User Story tutorial to the US Group, and both the above tutorials to the UC+US Group.

Maintenance Tasks and Design: Both the tasks took 2 hours for its completion. Both the tasks were the largest time taken tasks among the all.

Debriefing Interview: After all the completion of all the tasks, we proceed to conduct a debriefing interview. In this interview we asked our subjects open ended questions regarding their actions and their experiences while using the requirement formats during the whole experiment. We also asked them for their opinions and their feedback regarding the experiment.

TABLE 3

OVERALL TIME DURATION OF THE ACTIVITIES

Activities	Time Taken (minutes)
Questionnaire	10
Tutorials	10
Maintenance Tasks (3 tasks) Design	120
Debriefing Interview	10

3.3 Participants

Total of 6 subjects took participate in our experiment. 3 among them are post graduate students, 3 among them are graduate students. We placed the 1 post graduate students and one graduate student in the same group to counterbalance the background. Thus, the UC Group had 1 post graduate students and 1 graduate student, US group had 1 post graduate student and 1 graduate student, US&OC group each had 1 post graduate students and 1 graduate student and the UC&US&OC had only 1 post graduate student. Details about the subjects are as given below in Table 4.

TABLE 4

BASIC INFORMATION ABOUT THE PARTICIPANTS

Average Age	23
Occupation	3 Post Graduate Students. 3 Graduate students.
Degree	4 in Computer Science 2 in Information Technology
Years of Experience in Software development	1-2 years

3.4 Technology used for Implementation

The project used in this study was a web-based System based on the online addresses called" An online Address Book". We chose this system because it was of medium complexity .It used typical ASP.Net technology .It did not use an external database. In addition, it has an XML (extensible Mark-up Language) file with configuration information. We asked our subjects to update a feature in the system and add two new tasks.

Task 1: This task involves addition of a feature. Subjects were asked to add a field to display a passport size photograph of the person whom he wants to search with their addresses.

Task 2: This task also required subjects to add feature that allows the users to delete his/her old address after adding the new address. The addition of this feature means that there should not more than one address of a single person.

3.5 Demerits of the Experiment

We believe that every work done has some limitations with it. We also have few limitations which are as following: First drawback is that there were limited number of subjects i.e. only 6 subjects. However, this number is sufficient for a preliminary study. Second, our subjects did not have enough experience with the technology used in the experiment. Only 2 out of our six subjects were able to finish the implementation task. The rest of them were unable to complete the implementation task due to a lack of

some concepts in ASP.Net technology. Although our study has some limitations, the results obtained from the experiment provide us with some useful empirical data to evaluate the benefit of Use Cases, User Stories, and Chatting with On-Site Customers.

4 FINDINGS

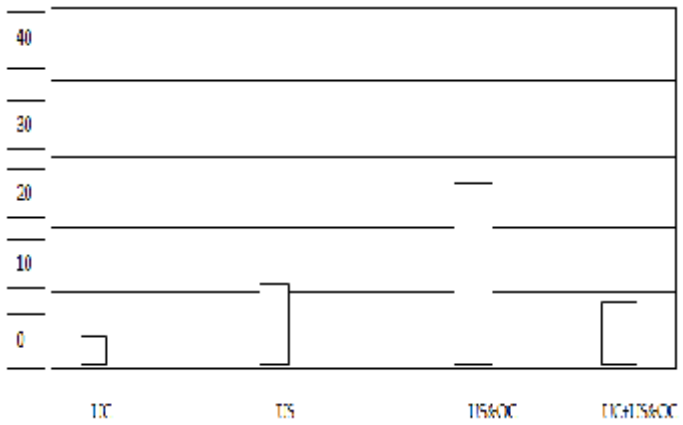
The results of the experiment indicate that Use Cases complemented elicitation of requirements by helping subjects to spend their time on understanding of requirements.

4.1 Comparison of time between all methods of gathering requirements

We compared the total time spent by the four groups understanding the requirements. In the case of the UC Group, we considered the total time understanding requirements as the time spent reading the Use Cases. In the case of the US Group, we considered the total time understanding requirements as the time spent reading the User Stories. In the case of the US&OC Group, we considered the time spent reading the User Stories and chatting with the On-Site Customers.

For the UC+US&OC group we considered the time spent reading the User Stories, Use Cases and chatting with the On-Site Customers. Graphical interface makes everything clear. So let's see the graphical image of the subjects spending time on understanding the requirements. The average time spent by each group in understanding requirement is shown in table 5 which is given below. We found that on average time spent by the subjects in the US&OC group is 24 minutes and 50 seconds, time spent by the subjects in the US group was 10 minutes and 10 seconds, time spent by the subjects in the UC+US&OC Group was third (9 minutes and 9 seconds), and the UC Group the least (4 minutes and 8 seconds).Overall, subjects spent little time reading User Stories when it is mixed the chatting done with the On-Site Customers. The second entry in Table 5 shows that on an average subjects in the US&OC Group spent more time (2 minutes and 10 seconds) reading the User Stories than subjects in the UC+US&OC Group (45seconds). Also in the second entry US Group spent more time in reading User Stories (10 minutes and 10 seconds) when done individually .We observed that subjects in the UC+US&OC Group spent more time on an average(9 minutes and 54 seconds) reading the Use Cases than subjects in the UC Group (4 minutes and 8 seconds).

During the experiment, subjects in the second and third conditions had the opportunity to ask the On-Site Customer Questions.



0-40 is scale for minutes

Fig. 1. Time Spent by Each Subject in Understanding the Requirement Format

TABLE 5
TIME SPENT IN UNDERSTANDING REQUIREMENTS

Group	UC (mm:ss)	US (mm:ss)	US&OC (mm:ss)	UC+ US&OC (mm:ss)
Time reading Use Cases	04:08	-	07:20	
Time Reading User Stories	-	10:10	02:10	00:45
Time spent on chatting with the OC			22:40	09:09
Total time understanding requirements	04:08	10:10	24:50	09:54

5 DISCUSSION

In this section, results presented in the previous sections will be discussed and interpreted .We observed that subjects in the US&OC Group spent more time asking questions than the subjects in the UC+US&OC Group. We believe that this took place because subjects in the US&OC Group did not have enough detail in the User Stories and they needed to ask the On-Site Customer to elaborate on what was required. We also believe that there may be a chance that questions asked to On-Site Customers are irrelevant and of no use. Subjects in the US&OC Group not only spent more time asking questions to the On-Site Customer than the UC+US&OC Group, but also they asked a greater number of both relevant and irrelevant questions. When a proper feedback is taken from the subjects that what they think about the requirement gathering process by chatting with on-site customers, most of the subjects said

that face to face interaction is better than chat. One thing is also observed that subjects in the UC+US&OC Group did not spend too much time reading the User Stories because they preferred to read the Use Cases, which had more detail than the user stories [8].

6 Future Work

The experiment done here have made a contribution to know about the impact of Use Cases, User Stories and chatting with On-Site Customers for the elicitation of the requirements for agile methodology But this experiment would be a huge success with more number of subjects. Other improvements include the use of subjects who were proficient in the technology Asp.Net.

7 Conclusion

Agile software teams occasionally use other requirements specification formats for requirements elicitation. We conducted a preliminary experiment to determine that addition of other things was a success or not. More over we wanted to know how the Use Cases, User Stories and On-Site Customer, or both could affect the understanding of requirements or not and results are very much satisfactory. We found that Subjects using Use Cases, User Stories and Chatting with on-site customers spent less time understanding requirements than subjects who had access to only user stories and chatting with on-site customers. We found that subjects in the UC Group spent least time in understanding the requirements. The on the second position, subjects in the UC+US&OC Group spent the least time understanding the requirements In the last, subjects in the US&OC Group were the ones who spent the most time of all the groups in understanding the requirements.

Acknowledgement

We thank our subjects from the bottom of our heart for their time and patience in participating in our experimental study.

References

[1] S. Soundararajan and J.D. Arthur, "A Soft-Structured Agile Framework for Larger Scale Systems Development," M-tech dissertation, Dept. Of Computer Science and Applications, Virginia Polytechnic Institute and State University Blacksburg, USA.

[2] M. Kohlbacher, E. Stelzmann, and S. Maierhofer," Do Agile Software Development Practices Increase Customer Satisfaction in Systems Engineering Projects?" *Systems Conference (SysCon), 2011 IEEE International*, pages168-172.

[3] A. Cockburn,"Writing Effective Use Cases," Addison-Wesley, 2001.

[4] M. Cohn, "User Stories Applied: For Agile Software Development,"Addison-Wesley, 2004.

[5] K. Beck," Embracing Change with Extreme Programming," 1999, vol. 32, Issue 10, October 1999, pages 70-77.

[6] K. Beck, "Extreme Programming Explained: Embrace Change," Addison-Wesley Professional, October 1999.

[7] P. Kruchten, "The Rational Unified Process: An Introduction," Addison-Wesley, 2003.

[8] R.E. Gallardo- Valencia, V. Olivera, and S.E. Sim, "Are Use Cases Beneficial for Developers Using Agile Requirements?, "*Fifth International Workshop on Comparative Evaluation in Requirements Engineering*, 2007.