Evaluation of Efficient Requirement Engineering Techniques in Agile Software Development

M.Usman Malik
UET Taxila
Rawalpindi, Pakistan

Nadeem Majeed Chaudhry
UET Taxila
Rawalpindi, Pakistan

Khurram Shahzad Malik
RIU Islamabad
Islamabad, Pakistan

ABSTRACT

Software development is one of the most booming industries of the world at the moment. All the major industrial groups are investing huge amounts of money in software development sector. Although several software development models have been developed in this regard, currently, the most widely used software development model is the agile model, due to its flexibility and change management capabilities. An important aspect of software development is that it is a phased activity which means that before ending up in the development of robust software application, several activities have to be performed. These activities are characterized by software development life cycle. The first and foremost activity of any software development life cycle is requirement gathering and specification. Now the problem is in agile development approach where nothing is fixed and requirements keep on changing all the time. In such scenarios it becomes extremely difficult to gather, analyze and document those requirements. In this paper evaluation of efficient requirement engineering approaches have been presented which could remove these requirement gathering issues in agile application development.

General Terms

Requirement engineering, Agile Application Development, Evaluation matrix

Keywords

Agile methodologies, scrum, software development life cycle, Requirement engineering techniques,

1. INTRODUCTION

Requirement Engineering is the first phase of software development life cycle and is considered corner stone of a software product. Efficiently gathered requirements result in development of a robust and reliable application. Apart from that, cost of requirement change is much higher than cost of gathering correct requirement. Hence, requirement engineering is an extremely important activity in SDLC.

1.1 Requirements Engineering issues in Traditional Models

In traditional software development models like waterfall models, requirements are frozen once decided. It means that once all the stakeholders have agreed upon set of requirements, they are immutable and cannot be changed later. After requirements have been gathered, work on next phase is started.

However, this approach has certain disadvantages [1] as described in the following section.

1.1.1 Requirements Change

Requirements can be changed at any time during the project development life cycle due to several reasons, for example customer might want to add or modify something, technology has changed or there can be any other reason. In such scenarios, in traditional rigid software development models, change in requirement can cause financial loss as well as project may take more time as compared to the decided one.

1.1.2 Business Process Change

Change in business requirement can occur at any time which results in the currently methodology becoming obsolete and outdated.

Hence it is very important to build a system, flexible enough to embrace change in requirement at any point of time during software development life cycle.

1.2 Requirements Engineering issues in Agile Approaches

Agile software development somehow satisfies this requirement but there are still few loopholes in requirement engineering for agile approaches [1].

There has been very little research which shows requirement activities in agile environment [2]. Research work has been done in this regard which shows that agile application development also have some issues in this regard [3]. Requirement engineering issues in agile environment have been discussed as follows:

1.2.1 Lack of unambiguous requirements

Requirements keep on changing in agile approach, therefore it is difficult to gather unambiguous requirement. Agile approaches lack a proper requirements engineering framework which can be used to properly document user requirements.

1.2.2 Lack of Requirement Activities

There are no documented requirement engineering activities which can be followed in order to properly obtain user requirements.

1.2.3 Incompatible Interfacing

Product developed in one phase with certain requirements might not be compatible with the next phase when customer’s need change resulting in the requirements change [4], which leads to interfacing issues between each iteration.

1.2.4 Difficulties in Non-Functional Requirement

Apart from, functional environment, there is no hard and fast approach for capturing and evaluating non-functional requirement in agile development [5]. Usually, whenever iteration is completed, running product is handled over to client for evaluation and client provides feedback regarding the non-functional requirement. Client is often times not able to correctly perceive the non-functional requirement of the product which affects overall quality of the final product.

This paper has been divided into five sections. First section introduces the concept and problem that this research will solve. Second section contains literature review of requirement engineering process and agile methodologies. Third section contains some of the most efficient requirement engineering approaches in agile methodologies. Fourth section contains evaluation of the approaches while the fifth section contains conclusion followed by future work which is last section.

2. LITERATURE REVIEW

2.1. Requirement Engineering Process

Requirement engineering is considered first phase and the most important phase in any software development life cycle. Requirement engineering is defined as the process of gathering, processing and documenting requirements of a system along with the context in which software is being developed. Purpose of requirement engineering processes is to only identify what is going to be developed; not how it is going to be developed [6].

Requirement engineering itself is a phased process. Activities in requirement engineering are divided into 5 major phases. In this section, we are going to explain these five activities. Requirement engineering activities flow has been mentioned in figure 2.1.

2.1.1 Requirement Elicitation

The first step in requirement engineering process is to identify the basic requirement and boundaries of the system that is going to be developed [7].This first step is called, requirement elicitation. In requirement elicitation phase, all the major stake holders of the system sit together and identify what are the fundamental requirements of the system and what is the scope of the system. Answer to these questions help in the identification of boundaries and context of the system.

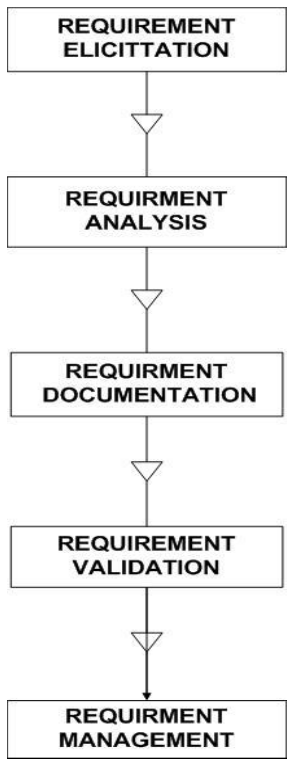

Fig 1: R.E Activities

2.1.2 Requirement Analysis

Requirement elicitation phase results in the list of fundamental requirements. Analysis phase further evaluates requirements on the basis of several criteria. It is verified that if requirements gathered in elicitation phase are actually required and will they play substantial role in the final developed system. It is also verified that no requirement is affecting other requirement i-e there is no contradiction between the requirements. Is there any missing requirement and lastly, does the entire requirements identified fall inside the budget and time constraints.

2.1.3 Requirements Documentation

Requirement documentation is the process of formally documenting all the requirements that have been elicited and analyzed in the previous phases. Purpose of documenting requirement is smooth sharing of requirements among several stake holders. A good requirement document should be readable, unambiguous, consistent, correct, feasible and concise [7]

2.1.4 Requirements Validation

Requirement validation process, as the name suggests, validates that requirements that have been documented are correct and will fulfill the desired functionalities of the system. It is also validated that requirements have been gathered using standard process and models. [7]

2.1.5 Requirement Management

Requirement management is the last phase of requirement engineering process. Requirement engineering is an umbrella activity and it continues throughout the software development life cycle. Requirement management basically refers to managing change in the requirements, requirement status monitoring, requirement tracking and version control.

2.2 Agile Software Development

Agile software development refers to set of software development methodologies based on iterative and incremental model where requirements and solution evolve through consistent collaboration between customer and all other stakeholders. In the year 2001, prominent researcher in the field of software architecture proposed agile manifesto at conference. The theme of agile manifesto is uncovering better ways of software development and helping others do it [8].

Four core features of software agile software development were stated in agile manifesto.

- Individuals and Interactions over process and tools.
- Working software over comprehensive documentation.
- Customer Collaboration over contract negotiation.
- Responding to change over following a plan.

In this section, some of the most widely used and popular agile development methodologies are described. All of these methodologies are based on agile manifesto with slight difference in their implementation strategy.

2.2.1 Extreme Programming

Extreme programming is an agile software development process with focus on rapid communication, instant feedback and process simplicity [9]. In extreme programming, as the name suggest major focus is on coding and programming rather than spending time on collecting requirement. As the development process continues, requirement changes are embraced during development phase. This eliminates time spend on requirement gathering activities discussed in section 2.1. Dr. Barry Boehm, famous researcher has suggested that cost of requirement change increases exponentially with the progress of the project. In case of XP, this additional cost can be balanced out because there are no hard and fast requirement engineering activities in the beginning rather focus is on adopting change during development.

In extreme programming, also known as XP, increments are extremely small and development process is evolutionary and incremental with a customer's representative always present with the development team and continuously evaluating the product being developed. If customer wants to change anything, it is instantly changed and another short iteration is released and evaluated immediately.

2.2.2 Feature Driven Development

Feature driven development is an agile software development methodology where product is developed in terms of individual features rather than product as a whole [10]. Major phases of FFD include, developing an overall model, developing a feature list, feature wise planning of the product, designing product by feature and then finally building product by feature. Feature list is reviewed and evaluate by all the stakeholders [11]. At the end of each week, a 30 minute meeting is held where status and details of feature being developed is shared among all the stake holders.

2.2.3 Scrum

Scrum is another extremely useful agile software development methodology which focuses on adaptability to requirement changes, flexibility in development approach and iterative application development. In scrum, team members collaborate with each other on consistent base at a single workplace to achieve the desired targets [10, 12]

Scrum methodology has several artifacts. Product backlog is the scrum artifact that contains all product features, mentioned comprehensively in the form of a list. In product backlog, features are arranged in the basis of priority. Sprint backlog is scrum artifact that contains features that are going to be developed in the current sprint. Sprint is a 2-4 weeks iteration in which a part of product is developed. In scrum, product being developed is always working with functionality of product increased after each iteration.

2.2.4 Dynamic Systems Development Method

Dynamic Systems Development methodology is a rapid application development approach following agile practices. Dynamic systems development focuses on quick software application development without stressing any particular requirement engineering technique [13].

3. EFFICIENT RE TECHNIQUES FOR AGILE DEVELOPMENT METHODOLOGIES

Purpose of the paper is to present some requirements engineering techniques that can be efficiently and effectively used to gather requirements in an agile development process. Agile application development is extremely flexible, therefore traditional requirements engineering techniques needs to be chosen carefully and should be modified slightly in order to adapt to the agile development process. Here, some of these techniques and approaches that can be helpful for gathering requirements in agile environments have been discussed.

3.1 Interviews and Direct Discussion

Interviews are one of the most efficient and conventionally used requirement engineering approaches. In agile environments where customer feedback is consistently needed in order to make changes at run time, importance of interviews is even more highlighted.

Interviews consist of direct interactions and discussions with the stake holders. Biggest advantage of interview is that it allows one to one communication between the customer and the development team which is necessary in any requirement engineering process. In agile methodologies, customer or one of the customer representative is always available to the product development team, therefore there is no need for lengthy, well documented and formal interview sessions, rather, short and to the point multiple interviews are more effective in agile application development because requirements are consistently changing and it is not good approach to collect long term requirements during agile development. Interviews are considered critical in requirement elicitation phase.

3.2 JAD (Joint Application Development Sessions)

JAD are similar to daily sprint meetings but they don't necessarily occur on daily basis. They can be weekly or even bi-weekly. During JAD sessions, project managers, executives, technical staff and custome sits together and define their long and short term road maps regarding the product being developed. In agile application development, such JAD sessions focus more on customer interaction and involvement and several such short sessions are more suitable rather than lengthy JADS.

Benefit of JAD is that it results in sharing of information between several stakeholders that will otherwise remain limited to individuals. JAD promotes, mutual understanding between several technical and non-technical teams and stake holders, encourages team work and cooperation.

3.3 Individual and Collective brain storming

Brainstorming refers to creative generation and evaluation of ideas about product. Brain storming plays an important role in agile engineering where features are developed at run time and quick generation of ideas is required to capture changing requirements of the customer.

Brain storming sessions are divided into two types, one is individual brainstorming where individuals generate and implement requirement ideas at run time and immediately evaluate its output. The second type of brain storming is collective brain storming session where requirements evolve when two or more than two stakeholders sit together and discuss their ideas.

There are two phases of any brain storming session. The idea generation phase where idea is actually created after lots of thinking and brainstorming. The second session is idea analysis session where idea produced in generation phase is analyzed and its feasibility is evaluated. The later phase is more rigorous in collective brain storming sessions.

3.4 Similar Systems Analysis

Analysis of similar systems is one of the oldest and tried and testing techniques for requirement gathering. In this technique, systems that are similar to the system being developed are analyzed and observed and their requirement documents are evaluated and modified to fit the needs of the current systems. This technique is extremely successful in agile development since there is not much time to think and brainstorm requirements, therefore whenever clients requirement change, compare that to already built similar system and evolve final requirement through these evaluations. This technique is less time consuming, more cost effective but it may not have much impact in the final product because it is highly unlikely to have exact similar requirements in two or more than two systems.

3.5 Use Case Scenarios and User Stories

Use case scenarios and user stories are the scenarios in which a user can use a software product. Use case scenarios are an effective way to capture requirements of the systems since they represent actual stories in which users are going to use the product. Use case diagrams are often using to capture such requirements. Use case scenarios are drawn with the help of customer where customer is asked how he wants to use the software or what are the cases in which he will interact with the system. Use case scenario are bit lengthy technique to gather requirement but can be effective in agile modeling where you have customer readily available onsite.

3.6 Surveys

The last and final technique presented in this paper is surveys. In agile methodology a survey can be presented to all the stakeholders in daily scrum meetings where feedback regarding the yesterday's task can be obtained. Also questions regarding what are the requirements for that day's task and what the requirements for the whole sprint are, such questions can also be included in these surveys. Usually surveys are done in ordered to gather requirements and feedbacks of large audience in shorter period of time.

4. EVALUATION OF REQUIREMENT ENGINEERING TECHNIQUES

In this paper, analysis and evaluation of all the six requirement engineering techniques have been done with respect to agile methodologies. Several criteria have been taken into account, in order to evaluate these requirement engineering techniques with respect to agile application development and in the end these techniques have been prioritized according to their effectiveness.

Three metrics have been defined for each criteria, LOW represented by L, MEDIUM represented as M and HIGH represented as H. One point is awarded for L, 2 for M and 3 for H. However weightage of last criteria i.e. Feasibility for Agile Methodology has double weightage since our major focus is on requirement engineering techniques in agile methodologies. Hence L fetches 2 points; M fetches 4 points while H fetches 6 points.

For research purposes, more than 20 software companies employing agile practices, and 18 university professors doing research in agile practices, have been consulted The result of this research is presented in analysis table. L, M and H values have been assigned to criteria on the basis of voting from the results obtained from software companies and professors.

4.1 Evaluation Criteria

In order to evaluate the requirement engineering techniques which have been explained in section 3, six different criteria have been chosen. These criteria have been selected after consulting software development companies and researchers. These criteria are the key for evaluation of requirement engineering techniques, particularly in reference with agile application development. There are total six criteria that have been explained in this section.

4.1.1 Cost Effective

Signifies that how much cost is spent on this technique. Represented as C.E in analysis table. For example L means low cost effective and will fetch one point for the technique.

4.1.2 Time Effective

Signifies that how much time does the technique takes. Represented as T.E in analysis table. For example H means highly cost effective and will fetch three points for the technique.

4.1.3 Final Impact
Final impact states that how much the technique impacts the final requirements of the system after the system has been developed. This is represented as F.I in analysis table.

4.1.4 Resource Effective
This shows that how effective is the technique in terms of resource consumption such as human resource and other technology resources. This is represented as R.E in analysis matrix.

4.1.5 Audience Reached
This criterion signifies that how many people were involved in the evolution of a requirement. For example interview may involve one or two persons while survey involves large audience. This is represented as A.R in analysis matrix.

4.1.6 Feasibility for agile Development
The last and the most important criteria is feasibility of a requirement engineering technique in agile application development. This criterion has double weightage in this research due to the fact that our main focus is on requirements engineering for agile application development. Here double weightage denotes emphasize on agile development. This is represented as F.A.D in analysis matrix

4.2 Analysis Matrix
Analysis matrix based on evaluation criteria discussed in last section and metrics decided have been presented in table 4.1. In first column of the analysis matrix, all the R.E techniques discussed in section 3 are listed while first row contains all the criteria discussed in section 4.1. According to our criteria and metrics, an ideal requirement engineering technique should get 21 points; 3 Points each for first 5 techniques and 6 for last i-e feasibility for agile development.

Table 1, Analysis Matrix of RE Techniques

	C.E	T.E	F.I	R.E	A.R	FAD	Total Sum
Interviews	M (2)	M (2)	H (3)	M (2)	L (1)	H (6)	**16**
JAD	L (1)	M (2)	H (3)	L (1)	M (2)	H (6)	**15**
Brainstorming	H (3)	M (2)	M (2)	H (3)	M (2)	M (4)	**16**
Similar System Analysis	M (2)	L (1)	M (2)	M (2)	L (1)	M (4)	**12**
Use Case Scenarios	M (2)	M (2)	H (3)	M (2)	M (2)	H (6)	**17**
Surveys	L (1)	L (1)	M (2)	L (1)	H (3)	L (2)	**10**

From the evaluation matrix, it can be observed that Use Case scenarios, with highest score of 17 are the best technique for requirement gathering in agile application development where user of the product is consistently involved and is readily available. Use case scenarios and stories can be immediately discussed and robust requirements can be gathered and documented at run time with the help of the user. Interviews and JAD have same scores of 16 each that show that interviews and Brainstorming sessions are equally suited for requirement gathering with a total score of 16, whereas JAD is not too far with sum of 15.

The result of this research and evaluation of R.E techniques have been presented graphically in Fig 2.. On x-axis, techniques have been located. It can be observed that use case scenarios have the highest bar with score of 17.

All these techniques are extremely useful with different purposes for example surveys might not look good with a score of 10, but if you look at audience reach criteria, it has the highest. Therefore in case you want to get feedback of large audience before finalizing the requirements, surveys can be a good technique. Hence apart from overall score, individual scores are also important in case you are interested in evaluating technique on any particular criteria.

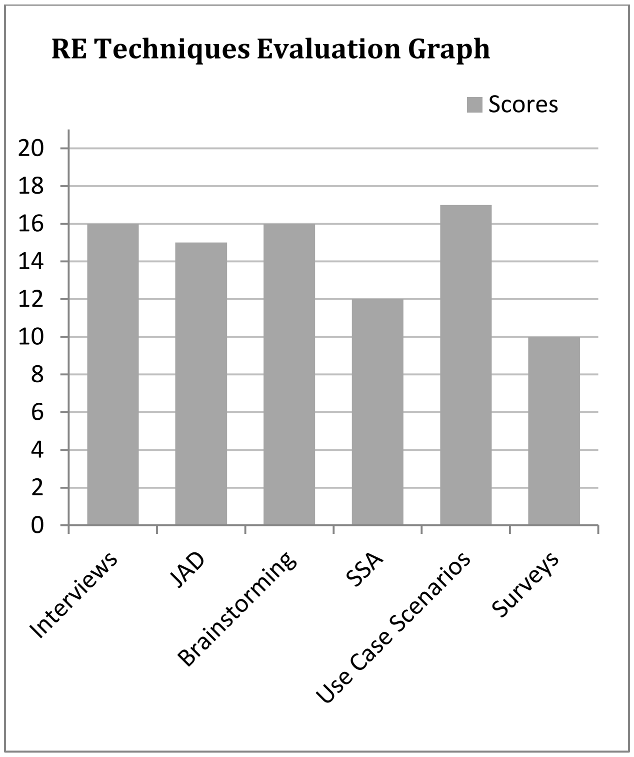

Fig 2: R.E Techniques Evaluation Graph

5. CONCLUSION
In this paper, several requirement engineering techniques have been presented that can be particularly useful for requirement gathering in agile environments. Several techniques have been presented along with suggestions on how each technique can be modified in order to fit the flexible nature of agile development. All these techniques have been evaluated on the basis of

research in software companies and the academia and final result have been proposed in the analysis matrix. Overall, use scenarios, interviews, brainstorming and JAD are suitable techniques for requirement gathering in agile development.

6. FUTURE WORK

In order to further research in the evaluation of requirement engineering techniques in agile development methodologies, further metrics, criteria and techniques can be added. For example, technology specific requirement techniques can be evaluated such as requirement engineering techniques in agile application development for handheld and mobile devices. Also, modifications in existing requirement engineering techniques can be proposed.

7. REFERENCES

[1] S.Ambler, "Agile Requirements Modeling", 2012 available at http://www.agilemodeling.com/essays/agileRequirements.htm

[2] J. Erickson, K. Lyytinen, and K. Siau, "Agile Modeling, Agile Software Development, and Extreme Programming: The State of Research" J. Database Management, 2005, vol. 16, no. 4, pp. 88-99.

[3] J. Nawrocki et al., "Extreme Programming Modified: Embrace Requirements Engineering Practices," Proc. IEEE Joint Int'l Conf. Requirements Eng. (RE 02), 2002, IEEE CS Press, pp. 303-310.

[4] . Beck, "Extreme Programming Explained: Embrace change", Addison-Wesley,1999.

[5] A. Eberlein, F. Maurer, F. Paetsch, , "Requirements Engineering and Agile Software Development", Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises.

[6] Alan M. Davis: Software Requirements Revision Objects, Functions & States, Prentice Hall PTR, 1994.

[7] Eberlein, A., Maurer, F., Paetsch, F., "Requirements Engineering and Agile Software Development", Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, 2003.

[8] Murauskaite A., Adomauskas V., "Bottlenecks in Agile Software Development using Theory of Constraints(TOC) Principles", Master's Thesis, Gothenburg, Sweden 2008.

[9] Kent BeckExtreme Programming explained, Addison-Wesley, 1999.

[10] Pekka Abrahamsson, Outi Salo, Jussi Rankainen & Juhani Warsta : Agile software development methods - Review and analysis, VTT Electronics, 2002.

[11] Peter Coad, Eric Lefebvre, Jeff De Luca : Java

[12] Modeling in Color with UML, Prentice Hall PTR, 1999, Chapter 6.

[13] Ken Schwaber, Mike Beedle: Agile Software Development with Scrum, Prentice Hall, 2001.

[14] Linda A. Macaulay: Requirements Engineering, Springer Verlag, 1996.

8. AUTHOR'S PROFILE

M. Usman Malik has completed his Bachelors in Software Engineering from University of Engineering & Technology in July, 2012. Currently he is enrolled as part time MS Scholar of Software Engineering at the same institute. He provides consulting services, business planning solutions to various IT firms. He also provides research services to IT Industry. His areas of interest are Operating Systems, Mobile Application Development, Digital Image Processing, Agile Software Development and Software Requirements Engineering.

Nadeem Majeed Chaudhry is serving as an Assistant Professor at University of Engineering & Technology Taxila. He is also a PhD scholar at the same institute. His areas of interest are Computer Networks, Operating Systems, Software Requirement Engineering and Mobile Application Development.

Khurram Shahzad Malik is MS Scholar of Project Management at Riphah International University Islamabad. He has spent around 10 years in industry managing several projects. He also holds a Master in English Literature from University of the Punjab. His Areas of interest are Project Management, Engineering Management, Agile Project Management and Requirements Engineering.