Requirements Engineering in Agile Software Development

Saad Abdullatif Alhuzami
435910300

King Saud University
College of Computer and Information Science
Department of Information Systems

Saad.alhuzami@gmail.com

1. Introduction

In today's world, the rapid delivery of software development is one of the most critical factors to mark it as a success or a failure. This factor is derived by the customer continuous need of change, which has been incarnated in a relatively new software development methodology called "Agile". Agile software development have become one of the most popular approaches. it has many applicable methods, All of them are centralized around customer satisfaction, continuous requirements changing, iterated cycles or releases of software delivery and deep involvement and contribution of business users.

In opposite, Requirements engineering or Requirement Analysis is mostly time consuming process. The traditional process of gather, analyze, validate and document requirements is essential for any software development. At this point, we can understand that agile is concerned with fast delivery, and face to face meeting with minimum amount of documentation and requirement engineering is concerned with documenting every detail which will help in sharing the knowledge even after the project is delivered [2]. The incompatibility is obvious, and my goal in this essay is to reduce this gap between agile and requirement engineering. That will be achieved by pointing the critical issues that affect requirement engineering process in agile and providing some state-of-the-art practices which may result in improving requirement analysis in agile method.

This essay will discuss the background of agile methodology, why it became popular, what is requirements engineering, why it is crucial to every software development project and how requirements engineering is being incorporated in agile approach. The second part will criticize the practical side of Requirements engineering in agile, what traditional requirements engineering fundamental practices have been compromised in order to comply with agile methodology. How to reduce the gap between the traditional process of requirements engineering and the velocious process of agile software development by proposing some practices. The last part will conclude what the essay has discussed in general and what are the areas that still have room for improvement.

2. Background and Description

2.1 Agile methods

Agile methods are approaches to develop softwares. They aim to give an organization the ability of delivering fast, with high quality products. From Jim Highsmith words, the meaning of being agile is to "Deliver quickly. Change quickly. Change often" [4]. To exploit what agile means more, let us take a deeper look at agile main values and principles. Agile has something called agile manifesto, which has been written in February 2001 by seventeen independent practitioners of several software development methodologies. Although they had some agree on many things, but they reached an agreement on four main values:

- Individuals and interactions over processes and tools
- Working software over comprehensive documentation
- Customer collaboration over contract negotiation
- Responding to change over following a plan

From these four main values, the following principles are formulated. Those principles represent what does it mean to be agile in detail:

- Working software is delivered frequently (weeks rather than months).
- Working software is the principal measure of progress.
- Customer satisfaction by rapid, continuous delivery of useful software.
- Even late changes in requirements are welcomed.
- Close daily cooperation between business people and developers.
- Face-to-face conversation is the best form of communication.
- Projects are built around motivated individuals, who should be trusted.
- Continuous attention to technical excellence and good design.
- Simplicity.
- Self-organizing teams.
- Regular adaptation to changing circumstances.

Agile approaches focuses on how to deliver high quality product on time under constantly and rapidly changing requirements and business environment. Agile methods success rate is gaining more credibility in the software and IT industries [4]. Fig. 1 shows that about 64% of agile implementations are successful, while 50% only of traditional implementations are successful. This comparison shows the statistics of 2013.

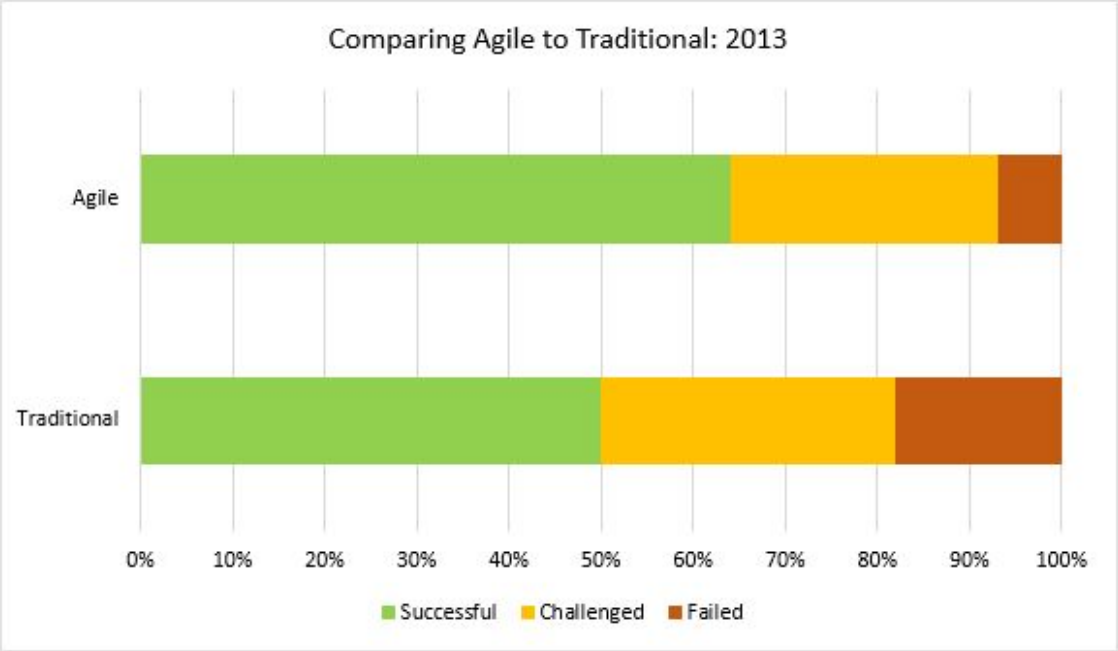

Fig.1: Source: 2013 IT Project Success Rates Survey, Ambysoft.com

There are multiple interpretations of agile methodology, they differ among them but in general they focus on delivering products that matches budget, time with high quality and customer satisfaction, here are the most popular agile methods:

• 1eXtreme Programming (XP)

XP does not explicitly mention requirements techniques in detail, it concentrate more on software development process and what activities to be completed during the process. It works by bringing development team together in the presence of simple practices, with providing the enough feedback which enables them to understand their current state.

• Scrum

Scrum is a method for managing and controlling software development process by applying ideas on flexibility, adaptability and productivity from industrial process control theory. The main Scrum artifacts are the *product backlog, sprints, and daily scrums*. With regard to Requirements Engineering, product backlog is where useful and necessary requirements are listed. It contains a prioritized list of all features, functions, enhancements, and bugs [4].

• Dynamic Systems Development Method (DSDM)

DSDM is a framework for rapid software development. The first two phases of DSDM are feasibility study and business study. During these two phases, the base requirements are

gathered. As the development process goes on, Further requirements are elicited. In DSDM any requirements engineering technique can be used during the development process.

• Adaptive Software Development (ASD)

ASD provides an iterative development framework for large and complex softwares. ASD iterative and incremental development with constant prototyping. ASD replaces waterfall cycle with a repeating series of speculate, collaborate, and learn cycles. This dynamic cycle guarantee continuous learning and adaptation of software development.

• The Crystal family

The Crystal family of methodologies has been founded in 1992, before XP, and ASD. The name "Crystal" came in 1997, before even the agile manifesto. Crystal is one of methodologies that contributed in the founding of agile methodologies. It is different than other Agile methods because it is a family of methodologies, not only one. The Crystal family members are labeled by different colors to indicate their "heaviness": Clear, Yellow, Orange, Red, Magenta, Blue, Violet [6].

2.2 Requirement Engineering

Von Neumann says *"There is no sense in being precise about something when you do not even know what you are talking about",* we cannot ask for something to be done is a certain way if we don't understand what is required. That is one of the simplest arguments that explains why poor requirements engineering is one of the most common failure factor in software projects since the early days of software development. As a result, IT community developed several standards and recommended practices in order to standardize and provide guidance to the processes of requirements engineering.

Requirements are derived by business need, if they are identified and fulfilled correctly, the gap between current and desired state of business will be reduced. The outcome or solution will result in improving the business performance by taking them closer to their business strategic and operational goals. If requirements are ambiguous, ill defined or missing, the result will be nothing but a low quality product, which will not fulfill and match the business need.

Requirements are defined as a solution-oriented statements of need. "Business Analysis Body of Knowledge®" Version 2 from IIBA defines a requirement as [19]:

1. A condition or capability needed by a stakeholder to solve a problem or achieve an
2. A condition or capability that must be met or possessed by a solution or solution objective component to satisfy a contract, standard, specification, or other formally imposed documents
3. A documented representation of a condition or capability as in (1) or (2).

The process of Requirement Engineering can be split into five consecutive activities: Requirement Elicitation, Analysis, Documentation,Validation and Management. In this section we will briefly define each one of these activities and list some of the main techniques that been developed to facilitate the outcome of its activity:

2.2.1 Requirements Elicitation

Requirements elicitation tries to discover requirements and identify system boundaries by consulting stakeholders (e.g., clients, developers, users). System boundaries define the context of the system. Understanding the application domain, business needs, system constraints, stakeholders and the problem itself is essential to gain an understanding of the system to be developed. The most efficient techniques for requirement elicitation are: **1.Interviews**, BABOK Version 2 (Business Analysis Body of knowledge) defines an interview as a systematic approach for eliciting information from a person or a group of people in an informal or formal setting by asking questions and documenting the responses. **2.Use Cases,** in use cases the analyst describe how the user interact with the system, every use case should address what user need from his interaction with the system. Use cases can be helpful if used in the early stages of software development in order to have a mutual understanding between the participant parties. **3.Observation,** in this technique the analyst or investigator will be viewing users in their daily work process, noting every detail that might help in designing the system. This technique is recommended when work processes are described in idealized or oversimplified way by stakeholders. Requirements Elicitation techniques are not limited to what have mentioned, interviews, use cases and observation are examples to give an idea on how an analyst can extract the requirements in the right way [10].

2.2.2 Requirements Analysis

Requirements Analysis comes afterward, this activity determine whether the gathered requirements are necessary or not. Also it should state if there is any contradiction between requirements or they are consistent. Requirements analysis checks ambiguity and guarantee completeness of the requirements, which means the requirements must express the desired state and include all related conditions and constraints. In addition to all previous checks, Requirements feasibility need to be checked, are the gathered requirements feasible to be

implemented considering the available budget, schedule and resources of developed system. One of the most popular techniques to diminish conflicts and prioritize requirements with stakeholders is Joint Application Development (JAD). while developing a software, JAD session is used to gather and analyze business requirements. in JAD sessions, stakeholders is close to each other. Hence, the user participation is enhanced as long as the facilitator of this workshop will prevent discussion to go off-topic. JAD will Also result in accelerated development and increased quality of specifications since customers and developers are face-to-face discussing desired features of their product. In case stakeholders had a disagreement regarding the requirements, JAD can be used to promote a professional facilitator who can find an ending to their disagreement. Last, JAD can improve the trust between customer and development team, which may help in future engagements and joint projects [4].

2.2.3 Requirements Documentation

Documenting the requirements is one of the most important activities in requirements engineering. Although it is not as easy as it sounds, a lot of projects fail due poor documentation of requirements [15]. Documenting requirements is used to communicate requirements between customers and development team. The requirements document is the baseline for the upcoming activities such as designing, testing, verification, validation and requirements management and control. When requirements are documented, the document must be clear, complete, correct, understandable, consistent, concise, and feasible [2].

2.2.4 Requirements Validation

Validation process is to verify and certify the requirements whether they are an acceptable description of the product which being developed or not. Requirements documents, organizational standards and regulations are some of the input for validation process. If validation process found issues, a list contains reported issues with their relative documentations and the required actions to resolve them will be created. Once the validation process is completed and all the issues are fixed, stakeholders will sign-off requirements documents.

2.2.5 Requirements Management

AS the project goes on, a lot of requirements changes will occur, here where the requirements management is needed. Requirements management will gather, store, distribute and manage any information related to requirements. In order to maintain manage and trace your requirements you need to perform activities concerned with change and version control, trace requirements and status tracking (whether it is in progress, completed or hasn’t started yet).

Requirements, design and implementation are correlated, if requirements traceability is ill, the three stage will not be in sync which probably will result in disatisfying product.

2.3 Requirement Engineering in Agile Methods

This section will discuss how agile requirements engineering activities differ from the traditional requirements engineering. The focus will be on the main differences which are the results of adopting agile standards. Requirements elicitation, analysis, documentation, validation and management activities are maintained in agile methodologies but with some changes, due to the constant changing, rapid development and minimized documentation. Some of these differences will be identified in this section, that should give an idea about what kind of obstruction they may produce.

Agile Projects Contracts: The most important requirements are expressed at the beginning by the stakeholders as clear as possible, this will help in estimating the initial cost for agile projects and guess the later changes costs.

All Development Team members Collects Requirements: in agile, requirements gathering activity is done by the whole team. This means, at one stage, all team members are working as analysts. In this way, the need of documents to share the gathered knowledge is reduced and the probability of misunderstandings decreases. Although, this means poor requirements documentation.

Using Common Language in requirements gathering: Requirements are collected using the language of the customer, not a formal language for requirements specification. This means that developers have to be introduced to the domain of the customer in order to understand him [6].

Direct Interaction Between Developers and Customer: This mean there is no intermediaries. This will minimize the number of documents required and the unnecessary communication layers. On the other hand, the chance of developer misunderstanding the business requirements is maximized.

Non-Functional Requirements: Agile does not provide any technique for gathering and managing non-functional requirements [6]. Non-functional requirements are gathered indirectly during requirements elicitation activity. The need of specifying non-functional requirements in agile is less important than traditional approaches, due to the continuous stakeholders interaction. Usually, the customer doesn't realise the high impact of non-functional

requirements. This may affect the final product deeply, and sometime it results in project failure [12].

Testing: Agile methods are based on iterative releases, which means continuous testing. Hence, testing is fundamental for agile, not only because testing reduce errors and enhance quality, but also it help in identifying new requirements, that is done through prototyping.

3. Discussion and Critical Assessment

Agile requirements engineering differs from traditional Agile requirements engineering since it takes an iterative discovery approach. Agile development take place where developing unambiguous and complete requirements specifications is not affordable or not applicable. These fundamental differences have impacted agile requirements engineering practices.

Instead of following the traditional process to produce complete and clear specifications which accurately describe the product, agile requirements engineering is more dynamic and adaptive. At this stage, it is clear that agile requirements engineering activities aren't centralized in one phase before starting the development; they are spread throughout the product development.

Although agile requirements engineering practices provide benefits such as improved understanding of requirements and being adaptable to the rapidly evolved needs of today's dynamic environments, they pose several challenges. Therefore, using agile methods in your organization is not always safe, and should carefully compare the costs and benefits of agile requirements engineering practices in their environment.

3.1 Requirements Engineering Challenges in Agile

Recent studies have identified several problems that could result from the lack of detailed requirements specifications [2]. Below, list of requirements engineering issues which an software development project may encounter when using agile approaches.

Absence of Requirements Engineering Activities: lack of available techniques which the analyst can pick from and there is no clear specification of activities in the agile requirements engineering.

Incomplete Requirements Elicitation: User-stories or whatever like it are just the tip of the iceberg for both requirements gathering and development processes in agile. Those

requirements are simply just a place to start. since agile is iteration-based, it is expected that more requirements will be added after each iteration, as more is known about the product. This makes software architecture development more difficult. The architecture that chosen by the team during the early cycles may become wrong, as later requirements becomes known [4].

Missing Requirements Documentation: Documenting user requirements is one of the challenging activities. Agile methodologies support creating use-cases and user-stories, but there are no standard processes or notations available to help gather requirements for these. Without strong interpersonal communication and facilitation skills, the process can fail [9].

There are three sources of knowledge in agile about the software to be implemented: code, test-cases, and programmers. If no change occurs for a long period of time, the programmers will forget most of what they wrote and maybe they won't be available at the time they needed. Thus, the only basis for maintenance is code and test cases.

Lack of Focus on Non Functional Requirements: In agile, handling non-functional requirements is poorly defined . Most of the customers when they specify what they want the system or product to do, they do not think about maintainability, portability, safety or performance. Most non-functional requirements should be known in development because they can help determining the database, programming language or operating system. Agile methods need to incorporate explicit techniques to handle non-functional requirements elicitation in a way they can be analyzed before implementation [1].

3.2 Proposed Practices to overcome Requirements Engineering challenges in Agile

As we understood what kind of obstructions an agile approaches may develop when requirements engineering activities is compromised. This section takes us further, we will go over some of the practices to overcome three main challenges of requirement engineering in agile software development:

1. *Requirements documentation*
2. *Requirements Traceability*
3. *Non-functional Requirements in agile*

3.2.1 Requirements Documentation:

The purpose of requirements documentation is knowledge sharing. In agile, no formal requirements specification is produced since agile doesn't care about documentation. The requirements are gathered on story boards, cards and paper prototypes. The lack of

documentation might cause problems on the long-term for agile teams , so, here are some suggested techniques to help solving this problem:

1) The agile team leader assigns two or three members to produce documentation in parallel and concurrence with development. Those team members will be responsible for handling functional and non-functional requirements, documenting, revising and maintaining consistency with development. In addition to that, we can involve some useful practices like "peer interviews" which will ensure accuracy and quality of documentation.

2) Automate the process of documenting certain agile practices, processes and requirements using UML modeling tools and project management tools.

3) Treat Documentation Like a Requirement One of commonly suggested philosophies to overcome documentation issues in agile is to treat documentation like any other requirement. This means documentation should have estimation and prioritization, and put it as an activity in your work item stack. Think of it like this, the need of writing a clear document is a requirement just like the need of writing a new feature. By treating documentation as a requirement you make its creation a visible and explicit decision for your stakeholders to consider, since it will consume time and it is part of your product delivery [12].

3.2.2 Requirements Traceability:

One of the most major upsets in software development is ensuring that system design meets the current requirements. Agile projects will work more efficiently if they incorporated requirements traceability tools. One good approach is to identify a traceability matrix in the test environment. This traceability matrix should map test cases to related code, this mapping should be identified and evolved to control changes in both sides. Hence, if the code is refactored, agile team will be able to update the traceability matrix again and determine what test cases they need to re-run.

Traceability mapping between requirements, code and test-cases can also drive software development by identifying which requirements have not been yet implemented, since they don't have related test-cases and/or code [7].

3.2.3 Non-Functional Requirements (NFR):

As mentioned earlier, there is a need and an opportunity for agile approaches to include techniques to improve the specification of non-functional requirements and identify them earlier, so they can be analyzed before the implementation start. Non-functional requirements

should be dealt with at the same stage of scoping user-stories. An early arranged meeting between stakeholders and agile team to discuss non-functional requirements can be a good start to identify and document the initial non-functional requirements, which the development can start with.

4. Conclusion

In this essay, I tried to explain the concepts and some major aspects of requirements engineering and agile methodology. And how agile can affect requirements engineering when you try to adopt agile manifesto. Some of requirements engineering essential activities appear to be ignored in the agile movement. The idea was to highlight what can go wrong, and how requirement engineering practices can be improved or changed to overcome agile challenges.

5. References

[1] W. Helmy, A. Kamel and O. Hegazy "Requirements Engineering Methodology in Agile Environment" IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 3, September 2012 ISSN.

[2] Frauke Paetsch, Dr. Armin Eberlein and Dr. Frank Maurer, "Requirements Engineering and Agile Software Development"

[3] A. Randell, E. Spellman, W. Ulrich and J. Wallk, "Leveraging Business Architecture to Improve Business Requirements Analysis" - Business Architecture Guild Whitepaper, March 2014.

[4] A. De Lucia and A. Qusef "Requirements Engineering in Agile Software Development" University of Salerno, Italy. JOURNAL OF EMERGING TECHNOLOGIES IN WEB INTELLIGENCE, VOL. 2, NO. 3, AUGUST 2010

[5] Sunaina "Analysis of User Requirements Gathering Practices in Agile and Non-Agile Software Development Teams" International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 9, September 2012.

[6] Alberto Sillitti and Giancarlo Succi "Requirements Engineering for Agile Methods", Italy.

[7] Lorena Delgadillo, "Story-Wall: Lightweight Requirements Management for Agile Software Development"

[8] S. Powell, F. Keenan and K. McDaid, "ENHANCING AGILE REQUIREMENTS ELICITATION WITH PERSONAS", IADIS International Journal on Computer Science and Information Systems Vol. 2, No. 1, pp. 82-95 ISSN: 1646-3692.

[9] J. Nawrocki, M. Jasiński, B. Walter, A. Wojciechowski, "Extreme Programming Modified: Embrace Requirements Engineering Practices" Poznań University of Technology, Poland.

[10] T. Aschauer, G. Dauenhauer, P. Derler, W. Pree, C. Steindl, "Could an Agile Requirements Analysis be Automated?—Lessons Learned from the Successful Overhauling of an Industrial Automation System" C. Doppler Laboratory Embedded Software Systems, Univ. Salzburg.

[11] Christopher Lee, Luigi Guadagno, Xiaoping Jia, "An Agile Approach to Capturing Requirements and Traceability" School of Computer Science, Telecommunications, and Information Sciences DePaul University, Chicago.

[12] Scott Ambler, Agile Requirements Modeling, http://agilemodeling.com/essays/agile-Requirements.htm

[13] "Agile Requirements Engineering Practices: An Empirical Study" by Lan Cao, Old Dominion University and Balasubramaniam Ramesh, Georgia State University. Published by the IEEE Computer Society., 2009.

[14] Armin Eberlein, "Agile Requirements Definition: A View from Requirements Engineering" Department of Elec.&Comp. Engineering University of Calgary, Calgary.

[15] Scott W. Ambler: Agile Modeling, John Wiley & Sons., 2001.
[16] Y. Dubinsky, O. Hazzan, D. Talby and A. Keren “SYSTEM ANALYSIS AND DESIGN IN A LARGE-SCALE SOFTWARE PROJECT: THE CASE OF TRANSITION TO AGILE DEVELOPMENT ”
[17] Agile Alliance http://www.agilealliance.org/the-alliance/the-agile-manifesto/the-twelve-principles-of-agile-software
[18] Richard Duncan “The Quality of Requirements in Extreme Programming” ,Mississippi State University, USA.
[19] Business Analysis Body of Knowledge, Version 2 by International Institute of Business Analysis.