Agile requirements engineering practices and challenges

Athina Verroiopoulou

SE 482, DePaul University, 2016

Abstract

The research paper is a result of an overview of existing case studies, reports, and surveys in order to gather information about agile requirements engineering practices. It presents the importance of agile RE and the differences between the traditional RE. It focuses and analyzes Scrum methodology activities. Agile best practices are presented and discusses how they solve traditional RE challenges. These agile practices may result in new challenges during project lifecycle. The challenges and their recommended mitigation is analyzed through the paper.

Keywords: Agile RE, Agile RE challenges

Agile requirements engineering practices and challenges

1. Requirements Engineering in Agile environments

1.1 The importance of Agile RE

The importance of collecting requirements in an agile manner emerged from the rapid changes in the business environment, where requirements tend to change rapidly and end up obsolete even before project completion. Moreover, high competition among companies enhance the need for unexpected requirement changes during the lifecycle of a project. Agile practices came also as a solution to traditional RE challenges, such as: misunderstandings of written requirements, communication problems and late requirement validation. In order to satisfy these demands RE was modified to be adopted in agile methodology.

Agile approaches can be valuable for software producers that don't mind highly uncertain requirements and are open to experimentation with new development technology. Their clients should also be interested in an evolving product which will be focused on their business goals. The key point in RE, regarding collecting requirements, is not whether to do it, but when to do it. That means Agile RE should allow changing requirements even late in the development lifecycle. The main benefits that cannot be achieved with a traditional method are better stakeholder / customer involvement during the whole project process, and high traceability in requirement changes. Unfortunately, agile methodologies bring about some challenges that we didn't have to face with a traditional approach. The major ones are the non-well defined, Non-functional requirements that are repeated in each iteration, as well as, the lack of requirement documentation (Helmy & Kamel & Hegazy, 2012).

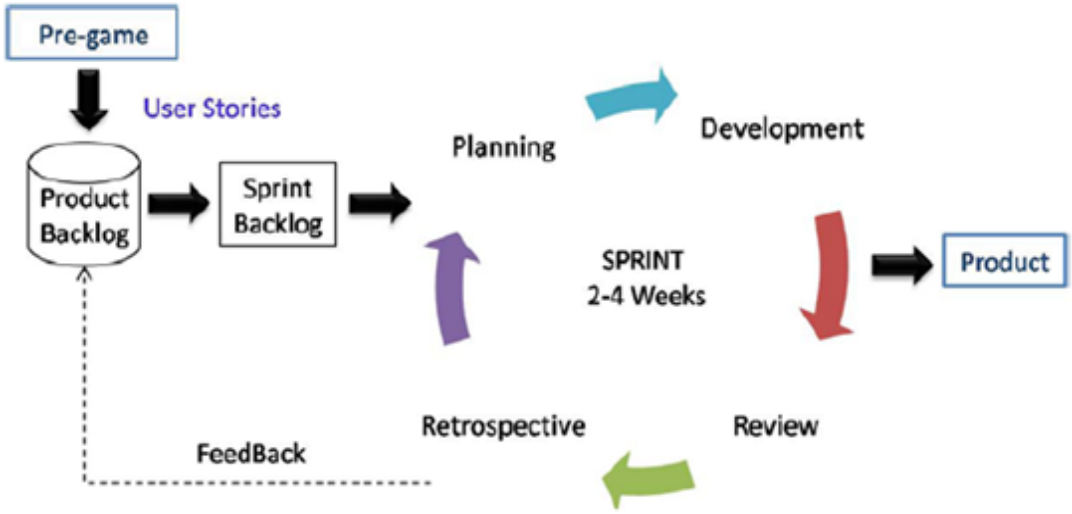
1.2 Activities and tools in agile RE

Agile RE follows the principles of agile manifesto to its processes and is concerned with discovering, analyzing, specifying, and documenting the requirements of the system. The processes used for agile RE are diverged regarding the application domain, the people involved, and the organization developing the requirements. Various agile methodologies exist and are based on a frequent incremental build, which encourage changing requirements. Some of the most important methodologies are (Lucia & Qusef,2010):

- Agile Modeling (AM) for performing modeling activities, which supports brainstorming techniques.
- Feature-Driven Development (FDD) which consists of a five-step process that focuses on building, design phases, and features that are client-valued functions.
- Extreme Programming (XP) which is based on values of simplicity, communication, feedback, and courage.
- Scrum which is based on flexibility, adaptability and productivity.

Scrum methodology is the most commonly used according to a 2014 survey by Version, approximately 94% of all organizations surveyed practice agile with approximately 56% of them using Scrum methods (Versionone, 2015). During Scrum methodology, requirements are developed and written as users' stories, usually followed by agile type use cases. They are stored into Product Backlog and prioritized by Product Owner. User stories are informal with a brief description of functionality as viewed by user perspective and user goals. Daily 15-minute team meetings are conducted for coordination and integration. A Sprint cycle lasts 30 days and during that period a specified set of Backlog features has to be delivered. Below is an illustration of the

Agile development model according to Scrum methodology: (Bomarius & Oivo & Jaring & Abrahamsson,2009)



As far as the RE activities and their implementation with Scrum methodology, below we have a table that outlines the key implementation points. (Vithana,2015):

RE Activity	Scrum Implementation
Requirements Elicitation	A Product Backlog created by Product Owner and stakeholders' participation
Requirements Analysis And Negotiation	Backlog Refinement Meeting A prioritized Product Backlog by Product Owner Requirements are analyzed for their feasibility by the Product Owner
Requirements Documentation	Face-to-face communication
Requirements Validation	Review meetings
Requirements Management	Sprint Planning Meeting Track items in Product Backlog Change requirements as needed

Agile RE activities are divided into Feasibility Study, Elicitation, Analysis and Negotiation, Documentation, Validation, and Management activities. During the Feasibility Study activity, the system overview and its worth is defined. As input is an outline description of the system and its

place within an organization. The output is a short report, determining if it's worth or not carrying on with the RE process. During the Requirements Elicitation activity, requirements are traced and system boundaries are identified by consulting stakeholders. This can be conducted by open or closed interviews, use case scenarios, observation and social analysis, focus groups, brainstorming and prototyping. Requirements are developed and written as users' stories and stored into Product Backlog, which is created by product owner and stakeholders' participation (Lucia & Qusef,2010).

During Analysis and Negotiation activity, we analyze requirements for necessity, consistency, completeness, and feasibility. Requirements are prioritized with negotiation of stakeholders and solutions to potential problems are identified. The main techniques used for this activity are JAD sessions, Prioritization, and Modeling. In the end of this activity we have a prioritized product backlog by the product Owner. Regarding Requirements Documentation activity, we link requirements between stakeholders and developers and is conducted with face to face communications. Moreover, it is the baseline for evaluating subsequent products and processes in addition to change control. Requirements validation activity certifies that the requirements are in acceptable description with the implementing system needs. Techniques used for requirements validation are review meeting, unit testing, evolutionary prototyping and acceptance testing. During Requirements management activity, sprint planning meetings are conducted, items are monitored and necessary requirement changes are implemented (Lucia & Qusef,2010). We ensure that all the information is being captured, stored, disseminated, and managed (Paetsch & Eberlein & Maurer).

As far as the tools used to capture the requirements, they usually are gathered in a postcard form by brainstorming methods and hang out on a pin board for reviewing. However, computer-based tools can be used in some cases. The most known are UML Modeling, for achieve a high

level description and manages to reverse engineer code to create documentation. Moreover, Requirements Negotiation tools can be used in order to identify, prioritize, and manage requirements. Instant Messaging tools can be used to maintain communication with the customer and Project Management tools can be used to store and retrieve requirements documents in an electronic format (Sillitti & Succi 2005).

2. Requirements engineering practices for agile methods.

A common best practice approach would be a high-level requirements envisioning during the initial stage of the project in order to have an understanding about the project scope and needs. Accomplishing good communication with customers, the team aims to identify the business goals, share a common vision, and trace the initial requirements for the system at a high-level. The documentation should be kept informal and the use of Modeling and Test Driven development would be valuable. Below we analyze the best practices we can implement in agile RE and their benefits.

Customer involvement practice is the key practice in every agile approach, as we aim the accessibility of the customer through the lifecycle of the project. We try to transfer ideas from the customer representative to the development team, instead of creating extensive requirements documents. The effectiveness of face-to-face communication depends on the intensive interaction between customer representatives and developers. Interviews with the customers are the best method for gathering all the necessary information for development without misunderstandings. Another valuable method is a short daily meeting between the team and the customer to enhance communication. Some benefits of this practice are that requirements can adapt easily to changes in the environment according to customer's directions, and there is no need for time-consuming documentation and approval processes (Ramesh & Cao, 2008).

Iterative practices are needed as only high-level requirement analysis is carried out at the beginning of the project. Requirements are just a starting point and they do not intend to be complete or cover all the features of the system. Keeping requirement analysis brief in initial stages is vital because the requirements volatility is high and re-evaluation of requirements is required even if the business domain is stable, as the technical detail are unknown and will affect implementation. Moreover, the customer can only clearly define the requirements when they see them, which means the more we know about the product, the more features or user stories will be added or discarded. Some benefits of this practice are: a more satisfactory relationship with the customers, and clearer and more understandable requirements due to customer accessibility and involvement. (Ramesh & Cao, 2008).

Practices that are used during Elicitation activity can be open or closed interviews, use case scenarios, observation and social analysis, focus groups, brainstorming and prototyping. Observation Social Analysis and Brainstorming practices, can be used to improve requirements elicitation where requirements are not pre-defined. Brainstorming is a group technique for generating new, valuable ideas, and promoting creative thinking. Interviews are a method for revealing what potential stakeholders think about the system under development. It can be in closed form where a predefined set of questions can be answered, or in open form, where spontaneous questions are answered to identify issues. Use Case analysis is a scenario based technique which identifies the actors involved in an interaction and describes the interaction itself. Each use case illustrates a user oriented view of one or more functional requirements of the system. Prototyping is used to communicate with their customers and based on their feedback on the prototypes developed for demonstration and experimentation, requirements are validated and

refined. The benefits of this practice is quick customer feedback on requirements and an alternative way to create formal requirements documents (Ramesh & Cao, 2008).

Practices that are used during Analysis and Negotiation activity can be JAD sessions, Prioritization, and Modeling. Joint Application Development (JAD) is a workshop with used to collect business requirements while developing a system. JAD aims to enhance user participation, expedite development, and improve the quality of specifications. In cases of conflicts between stakeholders' requirements, it can help promote the use of a professional facilitator who can help to resolve conflicts. Modeling can link the analysis and the design process. They are divided into models to be implemented, under implementation, and completed. (Lucia & Qusef, 2010).

Prioritization practice aims to prioritize features properly with respect of business value as defined by the customer though development process. Early stage prioritization in a project allow high value feature delivery which is desirable for every project and essential for projects with limited resources regarding budget, staff and schedule. During each development cycle we can conduct requirements prioritization simultaneously with other various development tasks such as making changes to existing functionality, bug fixings and refactoring, and also re-prioritize as a part of the planning activity. Requirements can be divided into essential requirements, useful capabilities, and desirable capabilities according to a standpoint of importance. Some benefits of this practice are that customer needs are met better by the development team and numerous opportunities for reprioritization exist (Ramesh & Cao & Baskerville,2010).

Practices that are used during documentation activity are informal and small and are used for sharing knowledge between team members. The features and the requirements are recorded on story boards, index cards, and paper prototypes like use cases and data flow diagrams Modeling practice is used in different levels of abstraction and is a part of the system documentation that

needs to be kept up to date. Practices that are used during requirements validation activity are review meeting, unit testing, evolutionary prototyping and acceptance testing. Some benefits of these practices are that we can provide progress reports to the customer and other stakeholders in the organization. Additionally, plenty of opportunities exist to ensure that the project is on target, enhance customer trust and confidence in the team, and identify problems early during development (Ramesh & Cao & Baskerville,2010).

Review meeting practice is used to present the new software in a formal way. We validate the requirements, review the developed features and get customer feedback. Moreover, customers are informed about strengths and weaknesses of the design and where advantages and limitations exists. Acceptance tests are used as a part of requirements specification to validate if the system reacts properly or clarify potential issues. They are similar to automated system tests but they are performed by the customer. The customers set the acceptance criteria for the requirements and test the requirements against these criteria. Unit testing practice is used for requirements validation and can be considered as an ongoing and up-to-date documentation. It can be seen as a repository for developers that try to understand the system, since unit tests show how parts of a system are executed. Evolutionary prototyping aims to deliver a working system to customers by focusing on customer interaction. The most important, high priority and well defined customer requirements are implemented in the initial system phase. Customers can experiment to see how the system supports their work and reveal errors and omissions in the requirements which have been proposed (Lucia & Qusef, 2010).

Practices that are used during management activity aim to make the system better according to customer needs and their feedback. We track changes made to requirements, design, or documentation in order to detect the reasons behind those changes. We can add or drop features

and conduct changes to features that are already implemented. Some benefits of these practices are that the need for major changes is minimized, as we have early and constant validation of requirements and the cost of addressing a change request is very low *(*Ramesh & Cao & Baskerville,2010). During Test-driven development (TDD) practice developers create tests before writing new functional code. Writing tests are treated as part of a requirements/ design activity, in which a test specifies the code's behavior. It can be considered as a source of free traceability information that may improve the efficiency with which tests are produced and code is written for each iteration. The main benefit is that Changes are incorporated easily. Refactoring practice is a part of TDD practice and is used to change the internal structure of the software. It aims to make a software structure that will be more understandable and cheaper to modify, without changing its observable behavior (Lucia & Qusef, 2010).

3. Challenges in RE

3.1 Challenges in traditional RE that mitigated with agile RE

As we addressed before agile requirements aims to solve challenges that occur during traditional RE. The table below illustrates a summary with the most important challenges in traditional RE that are mitigated by agile practices (Inayat & Salim & Marczak & Daneva & Shamshirband, 2015):

RE Challenges	Agile Practice
Communication Issues **Rare Customer Involvement**	Customer Involvement Requirement prioritization by the customer
Overscoping	Iterative RE
Requirements Validation	Review meeting Acceptance testing
Requirements Documentation	Informal Documentation

Communication issues and rare customer involvement between customers and development teams are mitigated by customer involvement and requirements prioritization by customers. In order to achieve that we conduct frequent face-to-face meetings to enhance the interaction with customers and among teams. Collocated teams are formed for better communication and collaboration and onsite customer or alternative customer representations are conducted. Customer and development teams should be located in the same place and if a direct customer is not available an alternative solution such as proxy customers can serve the same purpose. Cross-functional agile teams can be useful in the clarification and understanding of requirements. Moreover, prioritization of the requirements occurs in each iteration and ensures, that the customer goals will be met. Overscoping issues are mitigated by iterative RE and as a result the need to repeat allocation in projects is reduced. This happens as we have one continuous scope flow that allows for developers to receive a list of features that are constantly prioritized by the customer. Moreover, Gradual detailing of requirements contributes to a feasible scope and with the use of cross functional agile teams we can gather the important features.

Issues that may occur during requirements validation activity are mitigated by review meetings and acceptance testing with the use of prototyping and requirements prioritization.

Prototyping technique gives a blueprint of the product, and as a result it helps in validating the requirements. Customers continue with prioritization requirements as an ongoing activity which ensures that less important requirements remain on hold. Requirements Documentation issues are mitigated by Informal Documentation that aims to minimize the need for maintaining long documents. In order to achieve the desirable documentation, we use user stories that provide only the needed user demands and face to face communications that reduces ambiguities (Inayat & Shamshirband & Salim & Marczak & Daneva, 2015).

3.2 Challenges in agile RE

Each tradition RE risk is aimed to be mitigated by agile RE, however, this cannot be succeeding as agile RE has its own challenges. So, agile RE can either mitigate the risks or create challenges that will exacerbate them. The risks in agile RE can be characterized as either tractable or intractable. Tractable risks are those that are relatively easy to handle or manage in agile development. Intractable risks are those that are difficult to handle or manage when using agile RE. Below are presented the potential challenges that may occur during agile RE (Ramesh & Cao, 2008) & (Inayat & Shamshirband & Salim & Marczak & Daneva, 2015):

- Customer Involvement challenges: it's hard to find a direct customer representative for most software projects. Another issue can occur when various customer groups are involved and with each concerned about different aspects of the system. In these situations, achieving compromise in the short development lifecycle can be challenging. Moreover, establishing trust between the customer and developer, which is essential for agile RE, can be challenging. Unfortunately, for projects that can't achieve high-quality interaction, requirements can end up being ineffectively developed or totally wrong.

- Prioritization, challenges: they may occur as we use business value as the only or primary criterion for requirements prioritization. Also, continuous reprioritization, when not practiced with caution, leads to instability.
- Requirements validation challenges: During this activity, risks may arise as no formal modeling of detailed requirements exists. This means no formal aspects of verification exists and consistency checking or formal inspections seldom occur. Implementing acceptance tests may be difficult in case of no customer accessibility as they help to develop these tests. Maintaining or evolving prototypes can also be challenging and have an impact on features such as scalability, security, and robustness. Moreover, deploying prototypes too fast in early stages can create unrealistic expectations among customers.
- Management challenges: During this activity, risks may arise as requirement changes can lead to inefficient architecture. Redesign of the architecture should be implemented and that would affect the cost of the project and will increase significantly. Refactoring may be applied as an ongoing activity to improve the design and help to minimize architecture changes. However, it doesn't completely address the problem of inefficient architecture or the refactoring of the software may not be well managed if developers are inexpedient or there is scheduling pressure. If Refactoring doesn't work the only solution is to rewrite large application modules.
- Test-driven development practice challenges: It can be a challenge as developers are not used to writing tests before coding and it demands a lot of discipline. As it involves refining low-level specifications iteratively, requirements have to be fully understood and extensive collaboration between the developer and the customer has to be conducted. Refactoring is an important aspect of TDD, and can lead to serious challenges to traceability. Refactoring

of code may add new traceability links or discard the old ones between tests/requirements and code. Moreover, it may lead to temporary code degradation, when some of the existing tests fail to pass.

- Minimal documentation challenges: If the necessary documentation is not provided, this can lead to communication breakdown and misunderstandings. As there are no documented RE activities there is no formal way to follow up in order to properly obtain user requirements and this may lead to lack of Requirement Activities.
- Cost and schedule estimation challenges. The original estimation has to adjusted during development lifecycle, as the initial estimation of project size typically is based on the known user stories. Many of them might be discarded or added during the development lifecycle. This results in unknown costs estimations until the last stage of development and potential unrealistic expectations.

3.3 Architecture challenges during Agile RE

Several architecture-related issues may occur and can lead to a negative impact on architectural practices, artifacts, or design decisions. During Requirements Elicitation activity, as requirements are added or dropped as an ongoing activity, software architecture development can be challenging. The development team has to choose architecture during the early cycles, that may become inefficient in later stages as requirements can change. Without the knowledge of the entire requirements in initial stages the interface between requirements will be missing. This will have an impact on the next iterations and can lead to re-work as the requirements interfaces were not addressed. Another major problem is the incorrect prioritization of user stories, as they may be prioritized without the appropriate technical considerations. Re-factoring can be applied in cases

of identified critical interdependencies among User Stories during late stages of the project. If significant refactoring is required, then the whole structure of the software would have a huge impact.

The neglect of nonfunctional requirements can also be a major issue in architecture. It is important for non-functional requirements to be known in development as they affect architecture choices such as: database, programming language or operating system. They need to be traced and analyzed before implementation. Unfortunately, there is no widely accepted technique for eliciting and managing non-functional requirements and no formal acceptance tests exists for them. This is a huge risk for agile methods because non-functional requirements may deeply affect the final release of the application (Helmy & Kamel & Hegazy, 2012).

4. Suggested mitigation to Agile challenges

In order to solve the above agile RE challenges, we should implement some practices and techniques to mitigate the potential risks. The major suggestion for avoiding potential risks is to ensure that good communication between customer and team will occur.

During customer involvement issues, if a real customer is not available, surrogates take his place. Usually product managers or business analysts, who have direct contacts with customers, play that role and meet with the development team frequently. They are responsible for changes in requirements and decisions on development options. Budget and time estimation challenges can be mitigated by enhancing the communication between team members and customers. This will lead to less modifications made to the projects, and in turn, less cost and time changes. Prioritization challenges can be mitigated by the use of Joint Application Development (JAD) sessions practices. They are used to increase customer involvement and have a constant feedback during the development process.

During Documentation practices, agile teams should judge the amount of needed documentation regarding the customer's accessibility. They can be collocated or onsite, but can also be customers from distributed geographical locations. In this case, documentation is required to enhance collaboration and avoid misunderstandings. This can be achieved with a combination of use case diagrams and user stories or the use of project management tools. Moreover, user stories can be complemented with delivery stories in order to ensure that developers will conduct the right implementation choices in the coding stage of a sprint.

During requirements validation activity it should be suggested to include verification simultaneously with validation. In order to improve the quality of agile processes, that rely only on validation, it is suggested to use checking tools. Early stage requirements descriptions that are associated with effective management practices should be inspected. During requirements management activity it is recommended to avoid relying on only configuration management practices, aimed at the code level, but also find ways to provide traceability from customers' intentions to actual implementation.

As inappropriate architecture issues may occur, it is suggested to conduct proper code refactoring. It will be used as an ongoing activity to improve the design and modify the code as needed according to requirement changes. This would help redesign the architecture more easily and avoid bigger expenses. Without it, changes occurring during later stages would add to the cost and become cumbersome to deal with. Regarding Non Functional Requirements, we should get them into consideration at the same time that User Stories are scoped with customers. The use of NFR graph is suggested in order to refine non-functional goals into operations at the functional level and avoid conflicts between non-functional requirements (Inayat & Shamshirband & Salim & Marczak & Daneva, 2015).

Bibliography:

Reference 1:

Waleed Helmy W. & Kamel A. & Hegazy O. (2012), Requirements Engineering Methodology in Agile Environment, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 3

Reference 2:

Lucia, A. D., & Qusef, A. (2010). Requirements Engineering in Agile Software Development. Journal of Emerging Technologies in Web Intelligence JETWI, 2(3)

Reference 3:

Versionone (2015), 9th Annual state of Agile survey

Reference 4:

Bomarius F & Oivo M. & Jaring P. & Abrahamsson P. (2009) (Eds.) Product-Focused Software Process Improvement 10th International Conference, PROFES 2009 Oulu, Finland, June 15-17, 2009 Proceedings

Reference 5:

N.vithana, V. (2015). Scrum Requirements Engineering Practices and Challenges in Offshore Software Development. International Journal of Computer Applications IJCA, 116(22), 43-49.

Reference 6:

Paetsch F. & Eberlein A.& Maurer F., Requirements engineering and agile software development

Reference 7:

Sillitti A. & Succi G. (2005), 14 Requirements Engineering for Agile Methods

Reference 8:

Ramesh, B., Cao, L., & Baskerville, R. (2007). Agile requirements engineering practices and challenges: An empirical study. Information Systems Journal, 20(5), 449-480.

Reference 9:

Ramesh, B., Cao, L (2008), Agile requirements engineering practices and challenges: An empirical study. Published by IEEE Computer Society.

Reference 10:

Inayat, I., Salim, S. S., Marczak, S., Daneva, M., & Shamshirband, S. (2015). A systematic literature review on agile requirements engineering practices and challenges. Computers in Human Behavior, 51, 915-929.

Reference 11:

Bjarnason, E., Wnuk, K., & Regnell, B. (2011). A case study on benefits and side-effects of agile practices in large-scale requirements engineering. Proceedings of the 1st Workshop on Agile Requirements Engineering - AREW '11.