Issues in Informing Science + Information Technology

An Official Publication of the Informing Science Institute
InformingScience.org

IISIT.org

Volume 16, 2019

AGILE REQUIREMENTS ENGINEERING: AN EMPIRICAL ANALYSIS AND EVIDENCE FROM A TERTIARY EDUCATION CONTEXT

Meetu Thomas — Department of IT and Software Engineering, Auckland University of Technology, Auckland, New Zealand — meetu.thomas@aut.ac.nz

Mali Senapathi* — Department of IT and Software Engineering, Auckland University of Technology, Auckland, New Zealand — mali.senapathi@aut.ac.nz

* Corresponding author

ABSTRACT

Aim/Purpose: The study describes empirical research into agile Requirements Engineering (RE) practices based on an analysis of data collected in a large higher education organization.

Background: Requirements Engineering (RE) in agile development contexts is considerably different than in traditional software development. The field of agile RE is still nascent where there is a need to evaluate its impact in real-world settings.

Methodology: Using a case study methodology, the study involved interviewing nine experienced software practitioners who reflected on the use and implementation of various agile RE practices in two software development projects of a student management system.

Contribution: The primary contribution of the paper is the evaluation of agile RE practices in a large tertiary educational organization. Based on the analysis of the data, it provides valuable insights into the practice of agile RE in a specific context (i.e., education), but just as importantly, the ones that were omitted or replaced with others and why.

Findings: While the evolutionary and iterative approach to defining requirements was followed in general, not all agile practices could be fully adhered to in the case organization. Although face-to-face communication with the customers has been recognized as one the most important agile RE practices, it was one of the most difficult practices to achieve with a large and diverse customer base. Addressing

Accepting Editor: Eli Cohen | Received: December 11, 2018 | Revised: February 1, February 10, April 1, 2019 | Accepted: April 8, 2019

Cite as: Thomas, M., & Senapathi, M. (2019). Agile requirements engineering: An empirical analysis and evidence from a tertiary education context. *Issues in Informing Science and Information Technology, 16,* 97-112. https://doi.org/10.28945/4286

	people issues (e.g., resistance to change, thinking, and mindset) was found to be a key driver to following the iterative RE process effectively. Contrary to the value-based approach advocated in the literature, the value-based approach was not strictly adhered to in requirements prioritization. Continuous integration was perceived to be a more beneficial practice than prototyping, as it allows frequent integration of code and facilitates delivering working software when necessary.
Recommendations for Practitioners	Our study has important implications for practitioners. Based on our empirical analysis, we provide specific recommendations for effective implementation of agile RE practices. For example, our findings suggest that practitioners could address the challenges associated with limited face-to-face communication challenges by producing flexible, accessible, and electronic documentation to enable communication.
Recommendations for Researchers	Researchers can use the identified agile RE practices and their variants to perform in-depth investigations into agile requirements engineering in other educational contexts.
Impact on Society	There are a number of new technologies that offer exciting new opportunities that can be explored to maximize the benefits of agile and other requirements techniques.
Future Research	Future research could conduct case studies in different contexts and thus contribute to developing bundles or collections of practices to improve software development processes in specific contexts.
Keywords	agile requirements engineering, case study, empirical, agile, Scrum, tertiary education

INTRODUCTION

The important role of requirements engineering (RE) in the effective development of a software product and minimizing project risks is well recognized. The use of agile methods has gained significant momentum and is now generally considered a viable approach in a number of software development settings (Senapathi & Drury-Grogan, 2017). The success of agile methods has led to a substantial change in the way requirements engineering (RE) activities are carried out in software engineering projects (Wagner, Fernández, Felderer, & Kalinowski, 2017). Regardless of the method, there is a fundamental difference in the way agile RE activities are carried out (Leffingwell, 2011). Many agile methods advocate the development of code without a formal requirements analysis and design phase (Ramesh, Cao, & Baskerville, 2010).

However, as observed in recent studies, despite the growing body of knowledge on software development practices, very little is known on the actual practice of RE in general (Cheng & Atlee, 2007) and agile RE in particular (Wagner et al., 2017). Moreover, we could not find any studies that have investigated the use of agile RE in educational settings. Beyond very few empirical studies, the current literature does not provide much insight into the actual practice of agile RE and their effectiveness in supporting software development. In this research, we investigate these issues based on an in-depth exploratory case study conducted in a large tertiary education organization.

LITERATURE REVIEW

Requirements engineering, in general, is defined as a set of activities concerned with identifying and communicating the purpose of a software system and the contexts in which it will be used (Easterbrook, 2004). In agile RE, the basic RE activities of requirements elicitation, analysis, specification, and validation are interleaved with design and implementation. The outputs are decided through con-

stant interaction and negotiation with the customer throughout the development process. Although the common characteristics of agile RE, such as iterative RE, face-to-face communication, and prioritization, have been identified in the literature, the actual definition of agile RE has been described as vague (Heikkilä, Damian, Lassenius, & Paasivaara, 2015). It has been difficult to define agile RE as most agile methods such as Scrum and XP include RE practices in their overall method description without explicitly specifying RE (Heikkilä et al., 2015). Researchers are still trying to develop concepts related to agile (Käpyaho & Kauppinen, 2015). Based on a review of twenty-eight publications, Heikkilä et al. (2015, p. 205) propose the following definition which highlights that close collaboration with the customer and continuous requirements reevaluation are the core aspects of agile RE.

> " *in agile RE, the requirements are elicited, analysed, and specified in an ongoing and close collaboration with a customer or customer representative in order to achieve high reactivity to changes in the requirements and in the environment. Continuous requirements reevaluation is vital to the success of the solution system, and the close collaboration with the customer or customer representative is the essential method of requirements and system validation*".

Research on agile RE can be classified into two main streams: (i) reviews and mapping studies and (ii) a small number of empirical studies on actual practices used in agile software development. Most of the reviews and mapping studies have focused on agile practices in general (Dybå & Dingsøyr, 2008; Hasnain, 2010; Hossain, Babar, & Paik, 2009) where agile RE has been a tangential subject (Heikkilä et al., 2015). More recently, the number of review studies on agile RE is also increasing (Heikkilä et al., 2015; Inayat, Salim, Marczak, Daneva, & Shamshirband, 2015; Schön, Thomaschewski, & Escalona, 2017). In the next section, we discuss the empirical studies that have been conducted in agile RE.

Empirical Studies:

In an empirical study based on data from sixteen US software development organizations, Ramesh et al. (2010) identified six agile RE practices: face-to-face communication, iterative RE, managing requirements change management through constant planning, extreme requirements prioritization, prototyping, and review meetings and tests. Their findings highlight that intensive communication between the developers and customers is the most important practice in terms of its influence on the core RE activities (elicitation, analysis, documentation, and validation).

In an empirical study of agile RE prioritization in eight software organizations, Racheva, Daneva, Sikkel, Herrmann, and Wieringa (2010) found that the prioritization practices that were used differed from those postulated by the underlying agile methods such as Scrum. While the role of client's participation in prioritizing requirements was recognized, the study's findings highlight that the actual changes to requirements can be effectively implemented only when the interests of both the client and the developers align well in order to achieve sustainable business value. Daneva et al. (2013) identify a set of key agile RE practices introduced in the out-sourced project development department of a large organization. Their findings show the need for adapting the agile practices in order to fit the context of large projects. Kapyaho and Kauppinen (2015) conducted a case study to investigate how prototyping could provide support to solve agile RE challenges, such as lack of documentation. They found that in order to achieve the full potential benefits of prototyping, it needs to be complemented with other practices such as tracking quality requirements and acceptance test-driven development.

Hotomski, Charrada, and Glinz (2016) conducted a qualitative study involving twenty software practitioners from fifteen organizations (twelve of these used agile methods) to investigate two software documentation practices: managing requirements and acceptance tests. They found that performing the requirements and acceptance tests as two different tasks led to incomplete specifications and underestimating the complexity of requirements. In the case of agile RE, user stories were mainly used for requirements, but vision documents to provide long-term perspective were missing. In agreement

with findings from other related research, the study emphasizes the need for reliance on face-to-face communication.

In summary, empirical research in agile RE has mainly focused on investigating the use of specific practices such as prioritization, documentation/user stories, and prototyping. Except for very few empirical studies (Ramesh et al., 2010), little is known about how agile RE is actually implemented in real software development practice. Based on a review of twenty-one published studies on agile RE, Inayat et al. (2015) highlight the need for further empirical research on agile RE practices in real case settings. Therefore, it is especially pertinent to understand its use in a real software development context, where the development team adopted a customized approach to implementing agile RE practices. We believe that the lessons learned from real software development contexts are invaluable, as few such studies have been published.

Given the above, we used the agile RE practices identified in Ramesh et al. (2010) as a guiding framework to investigate the agile RE practices and challenges experienced by student management system development teams in a large higher education organization. Specifically, our research objective was to investigate the use and the extent to which the practices identified in the literature are implemented in the case organization and explore the associated challenges.

In the next section, we present the conceptualization of agile RE practices, which is followed by a description of the research method adopted in this study. Next, we present the findings of the study which is followed by the discussion section. Finally, the implications of the study and conclusions are presented.

AGILE RE PRACTICES

The research framework described in Ramesh et al. (2010) identifies three main factors that necessitate the agile RE approach: (i) evolving requirements, (ii) rapidly changing technology, and (iii) strict time constraints. Firstly, it is impossible to develop a clear, consistent, and complete specification of requirements that continually evolve during the development of today's innovative products and services. In recent years, software requirements, development, and implementation technologies are rapidly changing at a steadily increasing rate. This implies that the requirements of these emerging applications might not be fully understood due to the lack of domain experts in these application areas. Lastly, emerging trends such as continuous delivery and continuous deployment demand fast and short delivery cycles for delivering software.

The above factors cause the adoption of agile RE, whose main purpose is to employ continuous requirements evaluation and prioritization and close collaboration with the customer to support an agile software development lifecycle. The six practices identified by Ramesh et al. (2010) are used to investigate the agile RE practices and challenges that were observed in the case organization to guide our research: (i)*face-to-face communication*, (ii)*iterative RE*, (iii)*requirements prioritization*, (iv) *requirements change management*, (v)*prototyping*, and (vi)*user review meetings and acceptance tests*. We found that these six practices encapsulated many of the fundamental practices in agile RE and provided a useful structure to describe and analyze RE practices and challenges in the case organization.

BACKGROUND TO THE CASE

The case organization is a large higher education organization in the Asia-Pacific region that delivers education services for a large number of domestic and international students. The organization has been using the Scrum agile method for the past four years, and so it can be considered in the intermediate phase of adopting agile RE. The Student Management Systems (SMS) has been used by the institution for more than sixteen years, during which period it has grown significantly in student numbers and undergone major changes in the types of qualifications offered. This change has led to increased demands on the SMS, especially in terms of enrolment, reporting, and tracking progress from enrolment to graduation.

The practitioners shared their experience with agile RE in the context of specifically two projects (we call these project A and B) of SMS. Project A is a complex system related to the admission, enrolment process, results, and graduation. Project A is to replace the current system in place. The project is aimed at enabling users to maintain a set of academic components (e.g., paper lists, course of study, completion requirements) used to construct the courses of study for the present academic calendar year as well as for future academic calendar year. It includes the ability to have a current year course of study and the ability to setup components and course of study for a future academic calendar to support early enrolment into future courses of study. Project A is an ongoing piece of work, which involves a number of submodules. The ultimate goal of the project is to empower students to maintain their study plans and enable them to see a complete view of the trajectory of their current desired outcome in terms of the papers that they have already completed and those they are yet to complete along with the available options to achieve their qualification. Project B deals with sending specific student enrolment and qualification details from SMS to an externally hosted service provided by a third party for the Asia-Pacific including Australian and New Zealand educational institutions to securely publish academic record documents such as transcripts and parchments, which can be accessed by students and trusted third parties.

The software development process is based on Scrum, which includes agile RE practices and roles in general. The teams have two-week sprints that include daily stand-up meetings, sprint planning and sprint review meetings, and sprint retrospectives. Prioritized requirements reside in a product backlog as a list of PBI (Product backlog items). The product owner and the development team together conduct requirements analysis, prioritization, and validation by involving the customer representatives as often as required.

RESEARCH METHOD

Since the main objective of the research was to develop an in-depth understanding of the use of agile RE, an exploratory case study methodology was deemed appropriate (Yin, 2013). It is contended that the blend of technical and human-behavioural aspects in software development lends itself to qualitative methods, which let the researcher delve into a problem's complexity and develop rich, informative conclusions (Ramesh et al., 2010; Seaman, 1999). Agile requirements engineering is a socio-technical phenomenon which can only be understood by examining it in its real settings (Ramesh et al., 2010). Therefore, the practice of agile RE was examined in a real specific context (tertiary education), where experienced software practitioners adopted a gradual and customised approach to its implementation. We used the conceptual framework developed by Ramesh et al. (2010) as a guiding framework to investigate the implementation of agile RE in the case organization.

Data collection involved a series of nine in-depth semi-structured one-on-one interviews, conducted over a six-month period and later transcribed in detail. The interview data were supplemented with secondary data sources such as project documentation and publicly available information such as websites. The spectrum of the key roles involved in the two projects related to the SMS, namely, Business Analyst (BA) - 1, Developers (Dev1, Dev2, Dev3, Dev4) - 4, Product Owner (PO) - 1, Scrum Master (SM) - 1, Quality Analyst (QA) - 1, and Application Life Cycle Manager (ALM) - 1. Ethical guidelines were followed and informed consent was obtained from all participants. Interviews were generally of 1-1.5 hour duration and were followed up by some informal sessions to clarify and refine issues as they emerged. An interview protocol based on Ramesh et al.'s (2010) framework was used to guide data collection. Interviewees were encouraged to reflect on their experiences related to the importance, benefits, and challenges associated with agile RE. The responses of the interviewees included information on both the projects. All interviews were digitally recorded with the permission of the participants.

Individual interview transcripts were analyzed and grouped into three main themes based on Ramesh et al's (2010) framework (agile RE practices, benefits, and challenges). As participants related their input to specific themes, they were classified under the respective group and refined as the analysis

evolved. The themes were re-analyzed to ensure that they belonged to the correct theme/category. The analysis continued until the conceptual categorization we developed was well-supported by the data we collected.

FINDINGS

Our findings from the analysis of the interview transcripts are discussed in the following sections. First, we present an overview of agile RE journey from the perspective of the main underlying concepts and definitions associated with agile RE in the case organization. Next, we present our findings under each of the agile RE practice themes that were derived based on Ramesh et al's (2010) framework, We also present the challenges associated with the effective implementation of each agile RE practice.

MEANING AND IMPORTANCE OF AGILE RE

Interviewees offered a number of interesting perspectives on the meaning and conceptualization of agile RE, having experienced its adoption for about four years. At a high level, agile RE was described as permeating through all aspects of development and quality assurance. It was viewed as a flexible approach to the RE process. For example, in the case of requirements elicitation, requirements came from various sources including the manager, ALM, architect, team lead, or applications analyst, contrary to the traditional method where predominantly the business analysts are expected to provide the complete documentation. The business analyst explained that while the requirements were mainly in the form of user stories, "*a lot of times I don't see any problem with taking a document and encapsulating things and breaking things down and to putting onto TFS* [Team Foundation Server]", the lack of manpower, business analysts in particular, and the diverse range of users (students, administrators, and a number of different faculties) meant that the requirements were assigned to the developers merely by discussion without any documentation. This finding supports the fundamental assumptions of agile RE that it is not possible to have a complete set of verifiable requirements before the development starts (Ramesh et al., 2010). The meaning of agile RE can be summed up in the product owner's statement,

> *"..agile RE requirements engineering for me...I think if we talk about the concept of doing things in small chunks, keeping the business involved, the requirements to sort of develop the requirements iteratively while you're developing and testing"* .

AGILE RE PRACTICES

Face-to-face Communication

According to the participants, face-to-face direct communication with the customers was seen as one of the main practices of agile RE rather than creating extensive documentation. However, for both the SMS projects, face-to-face communication is mainly through a middle person, such a product owner or Student Support Systems, who in turn communicates with the customer. For the case institution with a large and diverse user base, direct communication with the customer is not always feasible as the product owner explained,

> "*It's not like you can just speak to one or two people. So, you always have to speak to a group of people or try to get consensus on something…but especially when it's something brand new, like the academic engine, that was very difficult to actually make sure that what we build is what the organization actually needs and wants. So that is the most difficult part."*

Most participants had a strong awareness of the significance of direct communication with the customer:

> *"without direct communication, sometimes it takes a while to clarify things because the person in the team who acts as the middle person between the developers and customer could have missed some critical information, and there are times that a decision relies on the customer specifically for a critical functionality. The turn-around time for this could be days to a week.*" (Dev2)

Though there is an attempt to include customers more directly, factors such as hierarchical views in the institution are hindering such efforts and changes:

> "*we may run into issues where they don't* [top management] *perhaps respond as quickly as they might, we might ruffle a few feathers politically, about talking straight to the customer because historically we've had people in the way, either a product support person or the solutions architect, and it took a week - where I could have been designing - to get the answer.*" (ALM)

Challenges: The inclusion of a middleman (most often the product owner) impacts on the requirements elicitation process (often leading to either inadequate or incorrect set of requirements) because something is lost in translation when there is no face-to-face communication between the development team and the different customer groups. This was found to be especially challenging for the developers as they had to spend additional time and effort to achieve a consensus of requirements in constant discussion/negotiation with the PO, business analysts and QA.

Iterative RE

Iterative RE is followed in both projects A and B, where normally high-level RE occurs in the first few iterations followed by later iterations that focus on the more specific and finer details. The product owner recalled from one of his recent experiences with project B,

> "*we did the big thing; the big flows and then we refined deeper and deeper and deeper. You don't focus on the finest detail on Day One. You make sure you have the big flows in place and then you iterate, iterate, iterate.*"

Iterative RE allows the bigger chunk of work to be broken into smaller, finer details where the estimated approximation of total man hours could be determined. It allows everyone in the team to work continuously and autonomously without many dependencies on other parts of the software. Iterative RE also gives the flexibility and ability to refine the requirements continuously. The business analyst explained in comparison with traditional RE,

> "*.. 'do exactly what the spec is' and you'd look at the spec and you'd go, 'really? there's something missing here' or 'it's just not right.' People in the old days would say to you, 'Program to the spec and do not deviate'… it's just that sometimes when you build something, once you've built it you're only starting. You've done this. What does this mean? Does it relate to what we thought about it? Iterative RE gives us the ability to get it right."*

Two challenges related to iterative RE were reported. Firstly, as the projects do not follow a formal RE process, the effort and size estimation is based on known tasks or user stories. Therefore, it is not possible to create accurate estimates for the projects. However, estimation is merely seen as a reporting statistic and therefore did not seem to hinder the benefits of iterative RE as the ALM explained,

> *"if you are worried about sizing effort, you might have to get used to not being about to do that very accurately to start with, but I don't care about sizing effort; for the effort is what the effort is. That's a reporting question; it comes after the fact.*"

Secondly, there was some resistance to change from some team members due to continual changes to requirements after every iteration.

Requirements Prioritization

A common practice in agile RE is to implement the features with the highest priority first so that the customers can realize most business value (Ramesh et al., 2010). Prioritization is one of the central

concepts of agile RE. In SMS projects, the prioritization is done at a high level in an informal manner within each release after which work items are prioritized within each iteration.

The product owner normally prioritizes based on factors such as team members' availability, current tasks, assigning high priority to tasks related to core functionality and features, and low priority to tasks such as administrative interface and generic maintenance. Within a release, although the customer prioritizes in terms of their assessment of how important a change request is or how critical a bug fix is, the development team does not necessarily always agree. As the ALM explained,

> "*we don't take their word for it, and we can change; we can put a completely different priority on the work item. So, they, for example, they'll do critical bug. A couple of weeks ago we went and had a look, and it can't be critical because it works exactly the same way in gen-two, as designed. …*".

The clients are not involved in the prioritization process, as they do not always fully comprehend how the different aspects of the system function together such that specific operations can be completed with the highest priority. Normally, the ALM and the product owner together evaluate and finalize the prioritization of tasks,

> *"When we started, we put all the administrative interface and generic maintenance screens as low priority and logged items in the backlog and the core functionality like the generation of parchments and transcripts had the highest priority. For each iteration we just view the backlog and decide, we're going to do it now or later"* (PO)

As a result, at a meta level, prioritization was not perceived as formal or "pure agile" as the business analyst explained,

> *"I mean pure agile would be that I'd have all my user stories written on cards and then have my customers order them on a big page. I mean, that would be and people do that but clearly, we didn't do that. Things were kind of defined for us."*

Within each iteration, the developers worked on tasks that are prioritized based on the severity ranking assigned on a user story,

> *"we always do requirement prioritization based on the severity ranking assigned on a user story. This is decided by the BA/Manager or ALM who acts as the PO after consultation with the customer. A great benefit of this is it lets the developer know which one will be delivered (or expected to be delivered) first, so getting that critical one out of the window will help in reducing the pressure of deadline"(*Dev1).

Challenges: as business value is not the primary criterion for requirements prioritization, no major problems or challenges exist except for the need for a longer sprint planning meeting for confirming the requirements. Another associated challenge would be in determining what criteria the project team would use to assign to the prioritization of requirement deliverables.

Managing requirements change

Changes to requirements are one of the core agile RE practices followed by the institution. The type and number of changes depend on the project, but most changes are handled upfront as they come throughout development, as one of the developers explained,

> *"for project A, changes are a constant thing from day 1. Because the project is conceptualized and handled by different outgoing project managers and team leads, changes have been coming from just about anywhere, from the business down to the team lead, during the development life cycle. Most of the time, the tests, if we are talking about unit tests, will be aligned to the changes made at the end of the sprint when all functionalities have been delivered, thereby ensuring the changed requirement works."* (Dev4).

According to the participants, requirement changes during development are accommodated in a very flexible manner, as there is no formal process for managing and controlling change. Most changes come through when the developers find out that some of the requirements are not what the users

actually meant or required which may result in requirements changes. Most changes come *"probably in the early stages to the middle with the occasional straggler at the end"* (Dev2).

A Product Backlog Item (PBI) is created for individual tasks or bugs in the current branch, which is normally the main branch, and then those tasks are pulled out of the backlog and linked to that PBI as the parent. PBI is the parent of the backlog tasks or bugs. The PBI does not exist in the backlog but is created as a parent work item for a suitable collection of backlog items. So, in general, there are a number of PBIs with associated tasks and bugs and then during sprint planning, a selection of PBIs would be chosen to go into that sprint and then they are added to that iteration along with the sub-tasks. If there is a change in requirement for a particular PBI, the PBI itself would be updated, regardless of where it is, even if it is part of a current iteration and everyone adapts to it.

Challenges: Occasionally, if a requirement does not fit into a sprint anymore, then it is pulled out, postponed into the next sprint. While the changes are normally noted either in an MS Word or solutions document using Team foundation server, it was not possible to track/ maintain a history of changes. This does not normally well align with testing and quality assurance and is prone to a number of errors,

> *"you'd have to go back to a previous version of that document to find it and unless you're looking at change tracking within Word, for example, there's often no way to know that something has changed, which is a little bit annoying from a QA point of view. As it relies heavily on someone telling you that they've changed because there's often no indicator in the system which flags it or notifies the tester that this has changed, so it's quite error prone.."*(QA)

Prototyping

Prototyping is not normally followed by the organization except under special circumstances. According to the developer,

> *"we sometimes provide a working demonstration of the software if the requirements are a bit complex and the delivery of the complete working product might take some time. This normally happens if the product is web-related and the user might not have a concrete idea of what is really needed as an outcome*."

Balsamiq Mockups (a graphical user interface mockup and website wireframe builder application which allows the designer to arrange pre-built widgets using a drag-and-drop editor; Balsamig, n.d.) is used instead of prototyping which enables the developers to have an in-depth conversation with the business analysts and assists in gaining a better understanding of the customer needs. In most cases, continuous integration is seen to be a more beneficial and useful agile RE practice than prototyping,

> *"we build all our source code every time there's a check-in. Because of that, the dev's have to check-in working code; you can't do halfway and put it in because you'll break the build. So, that is our version of prototyping. Functionally, it doesn't necessarily mean that you get working software, because you can have something that passes the build and breaks, the client, but's that fine. That means nothing ever half finished, from the point of view of source control. That together with the understanding that all the devs have, very well actually, that you don't hold onto your code. You don't wait a week to finish your stuff and then check it in; we check it in as often as you can."* (ALM).

Challenges: The organisation uses prototyping to some degree, i.e., it appears to follow 'system' prototyping, rather than 'throwaway' prototyping once development begins. Most participants perceived prototyping to be not only time-consuming but also creating a lot of confusion and unrealistic expectations from the diverse range of customers in the organization.

Review meetings and acceptance tests

Agile approaches normally use review meetings which consist of two parts: a sprint customer review meeting and a team retrospective meeting. The main purpose of the sprint review meeting is for the customers to review the completed work and any key decisions made during the sprint. The purpose

of the retrospective meeting is to review the team's progress, identify what went well, and reflect on areas of improvement. Both these meetings occur on the last day of the sprint where the sprint customer review meeting is followed by the team retrospective meeting. In the case organization, sprint retrospective meetings are held at the end of the two-week sprint, but review meetings with the customer seldom occur due to the large and diverse customer base,

> "*we do review meetings at the end of each sprint to discuss the challenges we faced the prior weeks and how are we going to make improvements, moving forward. This helps tremendously with identifying key issues during development and by addressing these issues, the future deliverables will benefit."* (Dev)

Every PBI created has acceptance tests which the QA uses to create test cases so that they can be tested effectively.

The large customer base did not permit the teams to have a regular review meeting with the customer, and the PO explained,

> "*we don't directly obtain customer acceptance. Our acceptance is really QA acceptance for release. And yes, before a major release we do have daily build and release meetings where we go through the outstanding work and every single work item is individually tested and accepted."*

Challenges: The customers not being directly involved in creating acceptance tests is seen as one of the main challenges for the case organization. Most participants agreed that such meetings would be very beneficial if at least the real customers could be brought in at regular intervals of time particularly when there is a big refresh module such as graduation.

A summary of the main findings in comparison with the agile RE approach and the agile RE practices as identified in the literature is presented in Table 1.

Table 1. Comparison of Agile RE practices between the case organization and the literature (Ramesh et al., 2010)

RE activities	Agile RE approach	Agile RE practices (Ramesh et al., 2010)	Agile RE in the case organization (SMS system)	Agile practices used to support the RE activities in the case organization (SMS)
Requirements elicitation	Iterative: requirements evolve over time and discovered throughout the development process.	-Iterative RE -Face-to-face communication	Preliminary requirements start off with a solution document and evolve over time	-Iterative RE -Face-to-face communication with the PO
Requirements analysis and negotiation	Focus on refining, changing and prioritizing requirements iteratively	-Iterative RE -Face-to-face communication -Constant planning -Requirement prioritization	Focus on refining, changing and prioritizing requirements iteratively	-Iterative RE -Face-to-face communication with the PO and business analysts -Requirements prioritization -Balsamiq Mockups
Requirements documentation	No formal documentation	Face-to-face communication	Solution documents	-Iterative RE -Solution documents, User stories & sub tasks
Requirements validation	Focus on ascertaining whether the requirements reflect current user needs	-Review meetings -Face-to-face communication	Continuous integration provides the working model of the product & unit tests ascertain whether the functionality has changed	-Face-to-face communication with the PO -Acceptance tests

DISCUSSION

We analyzed the agile RE practices that were implemented in the organization in comparison with those reported in the extant literature.

FACE-TO-FACE COMMUNICATION

Although face-to-face communication and intensive interaction between the developers and the customers has been recognized as one the most important agile RE practices, it is one of the most difficult practices to achieve (Racheva et al., 2010; Ramesh et al., 2010). This problem is more common in large projects with many customers, which requires a customer representative such as a product owner to divide their attention between the customer and the developers (Ktata & Lévesque, 2009). This problem is well evidenced in the findings of this study in the case of project A which is a large project with a large and diverse customer base. Direct communication with the customer is not always feasible in the case organization. The developers spend additional time and effort in discussing and negotiating the requirements with the PO, business analysts, and QA. The presence of a middleman such as PO results in either losing critical information or causing long delays in finalizing a decision and achieving consensus. Convertino and Frishberg (2017) argue that working with in-house representatives leads to two common types of misrepresentation: (i) they are unrepresentative as end users because they have prejudicial advantages such as knowing the software well including the workarounds, and do not use the product within the environment of the end user's real setting, and (ii) the evidence they bring to the team is also biased because they are more likely to focus on problems or pain points of the customers and less on what works well, or may ignore new requirements that are not yet addressed. In order to address this problem, organizations may have to look at alternate options so that the quality of RE is not compromised. For example, domain experts and business analysts may be employed to improve the quality and correctness of requirements (Heikkilä et al., 2015). The ethnographic process can be used to adapt to changing and unpredictable user requirements and to incorporate customers into the software development process (Surendra, 2008). Agile methods such as Scrum could be supplemented with goal-oriented requirements engineering to mitigate the issues with the single PO model in large agile projects (Ktata & Lévesque, 2009).

ITERATIVE RE

Most participants agreed that Iterative RE is followed reasonably well in both projects A and B. The PO explained that iterative RE places less emphasis on planning far ahead because the big difference is:

> "*you make sure that you focus on the biggest risk issues, sort of the big picture requirements, address them as you go along and then you refine and iterate. You don't plan in three months' time you're going to do this because you don't know*."

Though the customers are not directly involved, requirements are managed iteratively through the active involvement and participation of the PO and business analysts. Neglecting non-functional requirements is reported as a major concern with iterative RE in agile development (Käpyaho & Kauppinen, 2015; Ramesh et al., 2010). However, this is not considered a major issue in the case organization as non-functional requirements such as safety, usability, and performance are considered equally important to functional requirements. Both functional and non-functional requirements are clearly discussed and tasked for the teams to work. Similar to the findings in the literature (Ramesh et al., 2010), the two projects in the case study do not follow a formal RE process, and as such the effort and size estimation is based on known tasks or user stories. Though it is not possible to create accurate estimates, it is not seen as a major concern as estimation is mainly used for reporting purposes. For the case organization, the major concern with iterative RE is managing people issues and coping with people's resistance to change. A number of people related issues were reported. From the customer viewpoint, the PO explained the challenges involved,

> "*some types of requirements are quite difficult to involve the customer with because they also don't really know what they need or want. It's very difficult to go and speak to a faculty registrar or these kind of people, they are so used to the old way of doing things, they cannot visualize something new."*

And from a QA point of view, it is difficult for testers to comprehend that they will not be seeing the complete set of requirements in the first iteration but it is going to be changing iteration after iteration,

> "*that makes it a bit difficult to plan testing; because you might think you're done but then a few weeks later you realize the whole thing has changed round here and you have to retest it again and again." So, knowing that things are constantly changing is a concern for the testers.*

A need for changing people's thinking, attitude, and mindset is recognized as an important issue to be addressed, as the Scrum Master explained,

> *"Yes, it does take longer to get to the final product. It's faster to get some earlier version of it and I think some people struggle with the fact that it's not finished when they first see it. That's more just managing their expectations that this is version one of what could be version hundreds. People have to change their thinking to think that it's an ongoing evolution of that product. It's not a finished product up front and it's not meant to be because their business cases may change over time or their business processes will change…*"

Requirements Prioritization

RE prioritization in agile is different from traditional RE in at least two ways: (i) prioritization happens at inter-iteration time, and (ii) highest priority requirements are implemented first to realize business value (Racheva et al., 2010). A value-based approach to prioritizing requirements is advocated where the customer requirements, business requirements, and technological opportunities align well to achieve maximum value to the customer (Aurum & Wohlin, 2007). Although business value as defined by the customer is predominantly reported as the key criterion used in agile RE prioritization (Racheva et al., 2010; Ramesh et al., 2010), there is very limited understanding of value creation through prioritization (Petersen & Wohlin, 2009). Contrary to the value-based approach advocated in the literature, value-based approach is not strictly adhered to in the findings of this study. RE prioritization is not perceived as "pure agile" because the real customers do not always participate in the process. Developers work on tasks that are prioritized based on the severity ranking assigned to a user story within each iteration where the PO prioritizes by assigning a low or high ranking to specific tasks. Although the customer is sometimes involved in prioritization, their assessment is not always taken into consideration by the development team. This could be partly due to the fact that the developers' ultimate goals may be related to other issues such as reuse, concurrently running projects, and distribution of resources that focus on achieving maximum value to the organization (Racheva et al., 2010). This calls for consideration of the different ways that development teams balance achieving value for both the customer and the organization (Racheva et al., 2010).

Managing Requirements Changes

Accommodating changes to requirements ensure better alignment with the customer needs as they are easier to implement and cost less in agile development (Ramesh et al., 2010). Organizations that practice frequent and intense interaction with the customer during development receive constant feedback which precludes the need for major post developmental changes (Ramesh et al., 2010). It is surprising to note that in our study, although direct interaction with the customer is limited, a number of requirements changes are managed and controlled iteratively as they come through. This could be partly due to the constant interaction of the PO with both the development team and the business analysts. However, due to lack of formal process and version management in handling requirement changes, the changes are sometimes error-prone as they do not align well the tests. Other challenges reported in the literature include the changes to the architecture during the later cycles of develop-

ment and the need for refactoring (Käpyaho & Kauppinen, 2015; Ramesh et al., 2010). However, none of these challenges was reported by the study's participants.

Continuous Integration

Based on the findings of an agile RE study with prototyping, Käpyaho and Kauppinen (2015) recommend that the benefits of prototyping can be achieved only when it is used in conjunction with other practices. While prototyping is a recognized as a useful practice for getting fast and timely feedback from customers, it is also associated with a number of challenges (Käpyaho & Kauppinen, 2015; Ramesh et al., 2010). Maintaining or evolving prototypes can be difficult and lead to problems with scalability, security, and robustness (Ramesh et al., 2010). Quick development of prototypes can create unrealistic expectations among customers. Customers may be unwilling to accept longer development cycles that are necessary to develop more scalable and robust product implementations (Ramesh et al., 2010). Most participants in our study perceived prototyping to be time-consuming, and prone to creating unrealistic expectations from the large and diverse range of customers in the organization. Instead, continuous integration (CI) is deemed a more beneficial practice and participants reported positive experiences with its use. With CI developers integrate their code as often as on a daily basis and, when combined with small releases, guarantee the constant availability of a software product. For the case organization, this is equivalent to delivering working software when necessary. CI is an integral part of developing software that it has been characterized as being one of the best practices in agile software development (Claps, Svensson, & Aurum, 2015). CI is also fundamental to the success of Continuous Deployment where software is deployed more frequently to production (Olsson, Alahyari, & Bosch, 2012).

Acceptance Tests

Although agile approaches recommend customer review meetings for requirements validation, they are not commonly practised in many organizations. For example, in an empirical study of 16 organizations, it was found that review meetings primarily provided progress reports to the customer and other stakeholders rather than its intended purpose of reviewing and providing feedback on the developing features (Ramesh et al., 2010). This is also supported by our study's findings: while sprint retrospective meetings are regularly held at the end of the two-week sprint, review meetings with the customer are seldom held. And, due to the lack of direct access to customers, QA acceptance is used instead of direct customer acceptance. Given that most organizations use sprint retrospective meetings as part of their agile approach, and given the lack of evidence from both the literature and the current study's findings that customer review meetings serve as a useful agile RE practice, it is questionable whether review meetings should continue to be recommended as an agile RE practice.

In summary, the agile RE practices adopted in the case organization are Face-to-face communication, terative RE, Requirements prioritization, Requirements change management, Continuous integration, and Acceptance tests.

Limitations and Evaluation of Threats to Validity

We evaluated the possible threats to the validity by using the checklist for case study researchers recommended by Runeson and Höst (2009). When we planned the exploratory qualitative case study, the key question used to address the evaluation of threats to the validity of its results was its lack of generalizability, i.e., external validity, as the data collected is specific to the particular context of the case organization (Yin, 2013). Here we sought to gain an in-depth understanding of the practice of agile RE in one real software development context in an educational setting. The findings are presented in a way that allows the assessment of their potential applicability to similar settings. We believe our conclusions will be relevant and useful for other organizations in similar contexts to ours (purpose, project size, intermediate level of agile adoption (4 – 5 years)) and large customer base.

To minimize the potential bias of the researcher, we also considered the construct validity of our study (Yin, 2013). Yin recommends the following for achieving construct validity in case study research: use multiple sources of evidence (called triangulation), establish a chain of evidence (e.g., use a research protocol), and have key informants review draft case study reports. In this study, triangulation was achieved by involving participants with diverse roles in the two projects. A 'chain of evidence' was achieved by following an interview protocol. Transcripts were reviewed by key participants.

Reliability is concerned with ensuring a study could be repeated by another independent researcher so that similar conclusions can be drawn. Yin (2013) recommends using a case study protocol and a data repository to ensure reliability. This study maintained an interview protocol for data collection and data analysis. An electronic project database was maintained throughout the research process and most paper artefacts were converted to electronic form.

We also acknowledge the inherent weakness of the interview techniques, as they were driven by the first researcher, which implies that there is always a residual threat to the accuracy of how the interview is conducted and how the interviewees respond, i.e., an interviewee may not have correctly understood the original intent of a question or may not be honest in his/her answer. However, we believe that this threat has been reduced in our study, because the interviewer used techniques such as asking follow-up questions and open-ended questions encouraging the participants to respond in a number of different ways. Moreover, all interviews were completely transcribed and made available to the interviewees for review and feedback.

We acknowledge that this study explores the perceived benefits and challenges of using agile RE practices for one organization in its intermediate stages of adopting agile RE. Further research needs to include several organizations to produce evidence to evaluate the extent to which our findings are observable in similar contexts.

SUMMARY AND CONCLUSIONS

While the use of agile methods has gained significant momentum and is now generally considered a viable approach in a number of software development settings, the field of agile RE is still considered nascent where there is a need to evaluate its impact in real-world settings. We used the agile RE practices identified in (Ramesh et al., 2010) to guide our investigation on the agile RE practices and challenges experienced by student management system development teams in a large higher education organization.

We used an exploratory case study methodology involving nine experienced software engineers who reflected on the use and implementation of various agile RE practices. While the evolutionary and iterative approach to defining requirements is followed in general, not all agile practices could be fully adhered due to the hierarchical structure of the organization. Although face-to-face communication with the customers has been recognized as one the most important agile RE practices, it is found to be one of the most difficult practices to achieve in large projects with a large and diverse customer base. Addressing people issues (e.g., resistance to change, thinking, and mindset) is critical to following the iterative RE process effectively. Contrary to the value-based approach advocated in the literature, the value-based approach is not strictly adhered to in requirements prioritization. Continuous integration is perceived to be a more beneficial practice than prototyping, as it allows frequent integration of code and facilitates delivering working software when necessary.

A key contribution of the paper is its description of the agile RE practices that were followed in the case organization and how they deviated from those identified in the literature, but just as importantly, the ones that were omitted or replaced with others and why. Secondly, the study provides a list of agile RE practices derived from the case study that can be used by other researchers to further evaluate the appropriateness of specific agile RE practices. The findings contribute to developing bundles

or collections of practices to improve software development processes in specific contexts (Ramesh et al., 2010).

REFERENCES

Aurum, A., & Wohlin, C. (2007). A value-based approach in requirements engineering: Explaining some of the fundamental concepts. In *Proceedings of the International Working Conference on Requirements Engineering: Foundation for Software Quality*. Springer. https://doi.org/10.1007/978-3-540-73031-6_8

Balsamiq. (n.d.). In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/Balsamiq

Cheng, B. H., & Atlee, J. M. (2007). Research directions in requirements engineering. In *Proceedings of the 2007 Future of Software Engineering*, IEEE Computer Society.

Claps, G. G., Svensson, R. B., & Aurum, A. (2015). On the journey to continuous deployment: Technical and social challenges along the way. *Information and Software Technology, 57*, 21-31. https://doi.org/10.1016/j.infsof.2014.07.009

Convertino, G., & Frishberg, N. (2017). Why agile teams fail without UX research. *Communications of the ACM, 60*(9), 35-37. https://doi.org/10.1145/3126156

Daneva, M., Van Der Veen, E., Amrit, C., Ghaisas, S., Sikkel, K., Kumar, R., . . . Wieringa, R. (2013). Agile requirements prioritization in large-scale outsourced system projects: An empirical study. *Journal of Systems and Software, 86*(5), 1333-1353. https://doi.org/10.1016/j.jss.2012.12.046

Dybå, T., & Dingsøyr, T. (2008). Empirical studies of agile software development: A systematic review. *Information and Software Technology, 50*(9-10), 833-859. https://doi.org/10.1016/j.infsof.2008.01.006

Easterbrook, S. (2004). *Fundamentals of requirements engineering*. Retrieved from http://www.cs.toronto.edu/~sme/CSC340F/readings/FoRE-chapter01-v7.pdf

Hasnain, E. (2010). An overview of published agile studies: A systematic literature review. In *Proceedings of the 2010 National Software Engineering Conference*, ACM. https://doi.org/10.1145/1890810.1890813

Heikkilä, V. T., Damian, D., Lassenius, C., & Paasivaara, M. (2015). A mapping study on requirements engineering in agile software development. In *Proceedings of the Software Engineering and Advanced Applications (SEAA), 2015 41st Euromicro Conference on*, IEEE. https://doi.org/10.1109/seaa.2015.70

Hossain, E., Babar, M. A., & Paik, H.-Y. (2009). Using Scrum in global software development: A systematic literature review. *SBIE - Simpósio Brasileiro de Informática na Educação,* 175-184. https://doi.org/10.1109/icgse.2009.25

Hotomski, S., Charrada, E. B., & Glinz, M. (2016). An exploratory study on handling requirements and acceptance test documentation in industry. In *Proceedings of the Requirements Engineering Conference (RE), 2016 IEEE 24th International*, IEEE. https://doi.org/10.1109/re.2016.37

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. https://doi.org/10.1016/j.chb.2014.10.046

Käpyaho, M., & Kauppinen, M. (2015). Agile requirements engineering with prototyping: A case study. In *Proceedings of the Requirements Engineering Conference (RE), 2015 IEEE 23rd International*, IEEE. https://doi.org/10.1109/re.2015.7320450

Ktata, O., & Lévesque, G. (2009). Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects. In *Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering*, ACM. https://doi.org/10.1145/1557626.1557636

Leffingwell, D. (2011). *Agile software requirements: Lean requirements practices for teams, programs, and the enterprise*. Addison-Wesley Professional.

Olsson, H. H., Alahyari, H., & Bosch, J. (2012). Climbing the "Stairway to Heaven:" A multiple-case study exploring barriers in the transition from agile development towards continuous deployment of software. In *Proceedings of the 2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, IEEE. https://doi.org/10.1109/seaa.2012.54

Petersen, K., & Wohlin, C. (2009). A comparison of issues and advantages in agile and incremental development between state of the art and an industrial case. *Journal of Systems and Software, 82*(9), 1479-1490. https://doi.org/10.1016/j.jss.2009.03.036

Racheva, Z., Daneva, M., Sikkel, K., Herrmann, A., & Wieringa, R. (2010). Do we know enough about requirements prioritization in agile projects: Insights from a case study. In *Proceedings of the Requirements Engineering Conference (RE), 2010 18th IEEE International*, IEEE. https://doi.org/10.1109/re.2010.27

Ramesh, B., Cao, L., & Baskerville, R. (2010). Agile requirements engineering practices and challenges: An empirical study. *Information Systems Journal, 20*(5), 449-480. https://doi.org/10.1111/j.1365-2575.2007.00259.x

Runeson, P., & Höst, M. (2009). Guidelines for conducting and reporting case study research in software engineering. *Empirical Software Engineering, 14*(2), 131. https://doi.org/10.1007/s10664-008-9102-8

Schön, E.-M., Thomaschewski, J., & Escalona, M. J. (2017). Agile requirements engineering: A systematic literature review. *Computer Standards & Interfaces, 49*, 79-91. https://doi.org/10.1016/j.csi.2016.08.011

Seaman, C. B. (1999). Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering, 25*(4), 557-572. https://doi.org/10.1109/32.799955

Senapathi, M., & Drury-Grogan, M. L. (2017). Refining a model for sustained usage of agile methodologies. *Journal of Systems and Software, 132*, 298-316. https://doi.org/10.1016/j.jss.2017.07.010

Surendra, N. C. (2008). Using an ethnographic process to conduct requirements analysis for agile systems development. *Information Technology and Management, 9*(1), 55-69. https://doi.org/10.1007/s10799-007-0026-6

Wagner, S., Fernández, D. M., Felderer, M., & Kalinowski, M. (2017). Requirements engineering practice and problems in agile projects: Results from an international survey. In *Proceedings of XX Ibero-American Conference on Software Engineering (CIbSE), 2017* https://doi.org/10.7287/peerj.preprints.2038

Yin, R. K. (2013). *Case study research: Design and methods*: Sage Publications.

BIOGRAPHIES

Meetu Thomas completed her Masters in Computer and Information Sciences at the Auckland University of Technology. Her research interests include Agile Software Development and Requirements Engineering. Meetu currently works as a systems developer in Auckland, New Zealand.

Dr. Mali Senapathi is a Senior Lecturer in the School of Engineering, Computer and Mathematical Sciences at Auckland University of Technology, New Zealand. Mali's research interests include Agile Software Development, Requirements Engineering and Computing Education. She employs a variety of qualitative and quantitative empirical research methods including case studies, interviews, and surveys. Mali's research has been published in reputed journals and conferences such as the Journal of Systems and Software, Agile and XP Conference proceedings, Evaluation and Assessment of Software Engineering Conference (EASE), among others. Mali serves on the program committee of conferences such as ACIS, AMCIS, InSITE, PACIS, and ACE, and regularly reviews for premier journals such as Journal of Systems and Software, MIS Quarterly, and Computer Science Education.