Towards Understanding Technical Responses to Requirements Changes in Agile Teams

Kashumi Madampe
kashumi.madampe@monash.edu
Monash University
Melbourne, Australia

Rashina Hoda John Grundy
{rashina.hoda,john.grundy}@monash.edu
Monash University
Melbourne, Australia

Paramvir Singh
p.singh@auckland.ac.nz
University of Auckland
Auckland, New Zealand

ABSTRACT

As a part of an extensive study focusing on responding to agile requirements changes, we carried out a pilot study to understand the technical responses shown by agile practitioners to requirements changes. To the best of our knowledge, how agile teams respond technically to such changes has not yet been studied. We used a qualitative approach using Grounded Theory. Analysis of the interview data collected from ten agile practitioners in New Zealand and Australia resulted in identifying three stages where agile teams respond to requirements changes technically – while receiving, developing, and delivering changes. We found that even though agile practices do not recommend comprehensive documentation, *the product owner defining a requirements change in detail* was stated by the participants as the most common technical response. Developers *conducting a technical feasibility study* and *negotiations among product owner and team when receiving a requirements change* were the other most common technical responses.

CCS CONCEPTS

• **Software and its engineering → Agile software development**.

KEYWORDS

agile, requirements engineering, requirements changes, responses, teams

1 INTRODUCTION

Agile software development is carried out in an iterative and incremental manner. It aims to improve the customer's competitive advantage by acknowledging the principle *"welcome changing requirements, even late in development"* [3]. To incorporate this, requirements engineering (RE) practices including requirements elicitation, management, and validation are performed in every iteration. Several agile RE techniques exist to conduct these practices e.g., the technique *writing user stories* is used for the practice *requirement elicitation*. These practices help ensure the preservation of the agile value *"responding to change over following a plan"* [3].

However, we wonder whether these agile RE techniques can be applied to fit every situation with regard to requirements changes? How exactly do the agile practitioners respond to requirements changes in the real world? As a part of an extensive study on exploring the most appropriate techniques to handle requirements changes in agile development, we conducted a pilot study to understand how agile teams currently respond to requirements changes. Our pilot study identified that agile teams perform various practices and techniques – which we call "technical responses" – at three stages with respect to requirements changes. Namely, while *receiving* changes, while *developing* code, and when *delivering* changes. In this paper, we present our findings on how agile teams technically respond when they receive requirements changes.

2 RELATED WORK

The focus of this paper is to better understand the technical responses shown by agile teams to requirements changes on a situational basis. We aim to identify the most appropriate techniques to handle different requirements changes. However, to the best of our knowledge, studies done on agile RE and requirements changes to date do not address the focus of our study. Agile RE practices and techniques research [2, 4, 5, 8, 9, 11, 14, 15, 17] has received a considerable amount of attention, while other aspects on requirements changes [12] have received less research attention. Understanding of how agile teams respond to requirements changes technically is a current gap in the area.

Comparison of traditional and agile RE techniques [14] has highlighted the need for documentation in RE and the need for research on requirements stability. Similarly, a comparative survey on traditional and agile RE [9] showed that RE practices including requirements validation does not show any significant difference in traditional and agile software development. Research on whether agile RE is given as much importance as thought [17] found that user stories and use cases are mostly used for requirements representation. It also found that intensive communication with customers is used to capture requirements changes. A study on agile RE [4] stated that rapidly changing competitive threats, stakeholder preferences, software development technology, and time-to-market pressures are key reasons for rendering pre-specified requirements inappropriate.

Hoff et al. [8] conducted a survey on agile requirement prioritization factors. They found that fixing errors and cost benefit trade-offs are the key decision factors when prioritizing the requirements for implementation. They identified that better requirements management and having an increased level of stakeholder commitment decreases project failure. Significantly, Baruah and Nomi's study [2] has raised the fact that there is no structured approach to manage requirements changes in agile.

In terms of proposed solutions, Maiden and Jones[11] introduced a novel technique where an interactive table is used to sketch and write requirements. Likewise, Ernst et al. [5] introduced the REKOMBINE framework based on a propositional language *Techne* to be used in requirements modeling. Moreover, Reed et al. conducted a case study [15] on a technique using generalizations to manage uncertain requirements in agile. They found that developing an evolvable design may bring agility to traditional development as

well. However, work climate, continuous customer integration, and the iterative development nature in agile have not mitigated the negative effect of requirements changes in project success of system engineering projects [12].

More research is required to understand and identify the most appropriate techniques to handle changing requirements in agile projects. A pilot study on the emotional responses to requirement changes in agile projects identified a range of emotions that practitioners undergo when dealing with requirement changes [10]. To the best of our knowledge, no studies explore the technical responses to requirements changes in the agile development lifecycle. Our aim is to better understand technical responses to requirements changes in agile settings.

3 RESEARCH METHOD AND APPROACH

Grounded Theory (GT) is used in studies where social interactions and human behaviors are studied [6]. Our focus was to understand how agile teams respond to requirements changes, and since several empirical software engineering studies have used GT as their research method [1, 7, 13], we determined that GT was appropriate to conduct our research. Among the available versions of GT, we used Strauss and Corbin's version [16] to conduct this research because of its structured approach.

3.1 Research Question Determination

Conducting GT research begins with selecting an area of interest followed by a brief literature review, and determining broad research questions. This paper addresses the research question, **How do agile teams technically respond to requirements changes?** within our interest area, *agile requirements engineering*.

3.2 Data Collection

After determining the research questions, GT follows data collection using interviews, surveys, observations, and focus groups. In our pilot research study we used interviews to collect data. We advertised on Global, New Zealand, and Australian chapters of the professional network channel *Agile Alliance* in social media platforms calling for agile practitioners to participate in our research. From the 16 respondents, 10 (n=8 Nz; n=2 Au) were chosen as participants by considering their accessibility. A global head of projects, managers, business analysts, scrum masters, a senior solutions architect/ principle consultant, and testers participated in the study. They had experience in agile software development from 2 to 18 years, and all were experienced in using Scrum. Participants' information of the mentioned demographics *total years of agile experience*, *agile method experienced*, and other demographics, *age, gender,* and *total years of experience in software industry* are summarized in Table 1.

Participants' project and team information related to the project about which they chose to share their experiences with us were also collected through pre-interview questionnaires. The captured information included domain of the project, role of the participant in the respective project, type of the project, agile method used, iteration length, and the size of the team. The project domains of the participants were IT, Transport, Finance and Banking, Business Services, Facilities, Media and Communication, Telecommunication, and Manufacturing. Most of the project costing was time and materials, one fixed price, one internally funded, and one internal product development. XP, Kanban, Scrum, Scrum and XP Combo, and DSDM were the agile methods used in the projects. Two projects had iteration length of 1 week while the rest used 2 week time-boxes. Team size of the projects varied from 4 to 50.

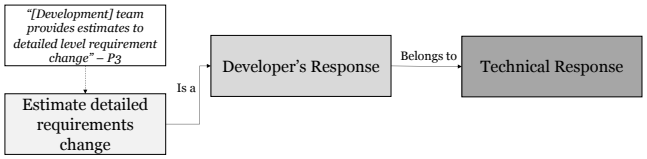

Figure 1: Emergence of the Category *Technical Response*

Each interview consisted of two parts. The first was a pre-interview questionnaire (approximately 10 minutes to fill out) using a Google form, from which we captured participant's demographic, team, and project information. The pre-interview questionnaire was used to save time for the interview (50-60 minutes long). Face-to-face (n=7) and online (n=3) modes were used to conduct the interviews. Interviews were composed of semi-structured questions such as *"Thinking of a recent project you were a part of, how did a requirements change travel through a project?"*. Here participants shared key milestones of requirements changes. Then we followed up asking questions such as *"What happens when the requirements change is at <participant defined milestone>?"* and *"What did you do at <participant defined milestone>?"*. Interviews were recorded and transcribed for analysis.

3.3 Data Analysis

In GT, data collection and analysis are simultaneous and interleaved activities, assisting the researcher to modify interview questions accordingly to focus the research direction. Data analysis in Strauss-Corbin GT starts with *open coding* [16] where fragments of data are meaningfully labeled as *concepts*. Concepts are *constantly compared* to identify the similarities, and *categorized* accordingly. These categories are then linked by identifying the relationships among them through *axial coding* [16]. Figure 1 shows the emergence of

Table 1: Participant Demographics (P#: Participant number; XT: Total experience in the software industry (years); XTA: Total agile experience (years))

P#	Age	Gender	XT	XTA	Agile Method Experienced	Job Title
P1	46-50	Male	20	12	Scrum, XP, Scrum and XP combo, Kanban, Feature Driven Development, Scrumban	Tester/ Scrum Master
P2	Above 50	Male	56	16	Scrum, XP, Scrum and XP combo, Kanban	Manager
P3	41-45	Female	24	4	Scrum, Kanban	Business Analyst
P4	41-45	Male	25	8	Scrum, XP, Kanban, Feature Driven Development, Dynamic Systems Development Method	Global Head of Projects
P5	36-40	Male	15	4.6	Scrum, XP, Scrum and XP combo	Scrum Master
P6	41-45	Male	26	1	Scrum	Scrum Master
P7	36-40	Female	3	2	Scrum, Kanban	Scrum Master
P8	36-40	Male	19.5	18	Scrum, XP, Scrum and XP combo, Kanban, Spotify, Company methods	Senior Solutions Architect/ Principle Consultant
P9	26-30	Female	7	5	Scrum, Kanban, Feature Driven Development	Tester
P10	31-35	Male	2	2	Scrum, Kanban, looking at adapting a Spotify like model in the near future	Scrum Master/ Business Analyst/ Manager

the category *Technical Response*. The concept *Estimate detailed requirements change* belongs to the sub category *Developer's Response*, which belongs to the category *Technical Response*.

The category which has most relationships to other categories is called the *core category* [16] and identifying the core category in Strauss and Corbin's GT version is called *selective coding* [16]. Once the researcher reaches a point where he/she does not find any new concepts or insights, *theoretical saturation* [16] is reached. Then the researcher writes the theory on the core category, which is the final step in GT method. However, we have not reached this stage yet. In this paper, we present emergent category *Technical Response* from our pilot study.

4 RESULTS AND DISCUSSION

Through data analysis, we found that agile teams display technical responses to requirements changes differently at different *stages*, i.e., while receiving changes, when developing code, and when delivering changes. Moreover, we found that these technical responses vary according to the *role*. Therefore, we categorized the technical responses according to the the stage and role. In this paper, we focus on cover the stage *when received*.

Table 2 shows the technical responses shown when a requirements change is received, the involved roles (product owner and development team), and the number of times the technical responses were mentioned by the participants. Out of the 16 different technical responses as given in Table 2, ten were found to be displayed by the product owner, 4 by development team, and 2 type of responses (e.g. negotiating and discussing) were displayed by both.

4.1 Technical Responses by Product Owner

The most commonly mentioned (n=3/10 participants) technical response indicated by a product owner to requirements changes is *specify requirements change in detail*. Even though agile does not recommend comprehensive documentation [3], this finding suggests that defining the requirements change in detail is required as an approach to better understand the requirements change. But this leaves the finding in contradiction with agile as this is not a best practice to adhere with agile values and principles.

Adding the requirements change to product backlog, where no further action is taken, and making the development team aware of the requirements change were the second most commonly mentioned technical response displayed by the product owners (n=2/10 each).

The other technical responses, *refining the product backlog* based on the priority, *adding the requirements change to corresponding source backlog* where different backlogs exist according to the source, *analysing the impact on other requirements, conducting feasibility study, analysing the impact on other products* where dependencies among product exist, *negotiating with customers,* and *forcing the team to implement the requirements change* were stated by different participants (n=1/10 each).

An interesting finding is the existence of micro-management (n=1/10), i.e., *forcing the team to implement the requirements change* where agile teams are supposed to be self-organizing [7]. This implies that teams may deviate from agile principles in some given different situations.

Table 2: Technical Responses Shown When a requirements change is Received (Mentioned#: Number of Times Mentioned (n=10); PO: Product Owner; RC: Requirements Change; Dev: Development Team)

Technical Response	Involved Role	Mentioned#
Specify RC in detail	**PO**	**3**
Add RC to product backlog	PO	2
Make the team aware of the RC	PO	2
Refine the product backlog	PO	1
Add RC to corresponding source backlog	PO	1
Impact analysis on other requirements	PO	1
Conduct feasibility study	PO	1
Impact analysis on other products	PO	1
Negotiate with customer	PO	1
Force team to implement RC	PO	1
Conduct technical feasibility study	**Dev**	**3**
Question the PO	Dev	1
Estimate detailed RC	Dev	1
Implement RC	Dev	1
Negotiate RC	**PO, Dev**	**2**
Discuss RC	PO, Dev	1

4.2 Technical Responses by Development Team

Conducting technical feasibility study as the requirements changes are received was mentioned most commonly (n=3/10) as the technical response shown by the development team. This indicates that prior to taking responsibility for implementing a received requirements change, performing a technical feasibility study is thought to be required. Hence *conducting technical feasibility study* can be said to be a decision factor for accepting requirements changes. This can be further studied in relation to Hoff et al.'s study on agile requirement prioritization decision factors [8]. Other technical responses: *questioning the product owner, estimating the detailed requirements change*, and *implement the requirements change as soon as it is received* were stated once by the participants.

Some responses contradicted each other, e.g. *implementing a requirements change as soon as it is received* is in contradiction with the technical response *conducting technical feasibility study* which is executed before implementing the requirements change. Also, implementing the requirements change as soon as it is received implies that it is implemented in the current sprint itself. The product owner has to add the requirements change to the iteration backlog, which was not mentioned as a technical response shown by the product owner. These leave us with the questions, (1) Why are some agile teams empowered to lead in decision making when accepting requirements changes and whereas the others are not?, (2) Are the requirements changes implemented without adding them to the sprint backlog?, (3) Are the teams using the same definition for "requirements change"?, (4) Do different kinds of requirements changes (e.g., minor vs. major change) result in different technical responses?

4.3 Technical Responses by Product Owner and Development Team

Negotiating the requirements change was a most commonly mentioned response involving both product owner and development team (n=2/10). Arguably these internal negotiations favoured the most empowered party. Depending on the perceived value of the facts/arguments the team and the product owner present when negotiating, the decision whether the requirements change is accepted straight away for implementation or any other necessary action is required before implementation is taken.

The other technical response shared by both product owner and development team is *discussing the requirements change*. Discussions can either be clarifications or may steer to internal negotiations. Conclusively, the shared technical responses by product owner and technical team clue that interconnections among these technical responses may exist. In order to assure this verdict, further investigation is required.

5 LIMITATIONS AND FUTURE WORK

Since this was a pilot study, the number of participants were limited to 10. We will be recruiting more participants as the study progresses to improve the quality of the findings and aim for theoretical saturation.

Developers are the ones who implement requirements changes but none of the agile practitioners responding to the pilot study advertisement were developers. Therefore, to mitigate this limitation, we will be modifying the advertisement to recruit developers as participants in the forthcoming round of data collection. Also, 8/10 of the participants in the conducted study were from New Zealand and 2/10 of the participants were from Australia. To include more participants from diverse territories, we will be conducting a worldwide survey.

Furthermore, we will investigate more on interconnections among the technical responses and different kinds of requirements changes. Technical responses at the other two stages, i.e., *while developing code to implement a requirements change*, and *when the requirements change is delivered*, will be explored, as our pilot interview data was inadequate to use for these. Social responses, emotional and behavioral in particular, will also be studied.

6 CONCLUSION

Through this interview–based pilot study with the participation of 10 agile practitioners from New Zealand and Australia, we found that agile teams show different technical responses to requirements changes when they are received. Grounded Theory analysis shows that product owners mostly seem to specify requirements change in detail when received. Also, developers mostly perform technical feasibility studies when requirements changes are received. In addition, product owners and the development team have negotiations among themselves, which is the highest mentioned technical response where both parties are involved. We will expand our study to investigate how agile teams respond technically and socially to requirements changes at all three stages with a wider coverage of agile practitioners.

REFERENCES

[1] Steve Adolph, Philippe Kruchten, and Wendy Hall. 2012. Reconciling perspectives: A grounded theory of how people manage the process of software development. *Journal of Systems and Software* (2012). https://doi.org/10.1016/j.jss.2012.01.059

[2] Nomi Baruah. 2015. Requirement management in agile software environment. In *Procedia Computer Science*, Vol. 62. Elsevier B.V., 81–83. https://doi.org/10.1016/j.procs.2015.08.414

[3] K Beck, M Beedle, A Van Bennekum, A Cockburn, Ward Cunningham, Martin Fowler, James Grenning, Jim Highsmith, Andrew Hunt, Ron Jeffries, Jon Kern, Brian Marick, Robert C Martin, Steve Mellor, Ken Schwaber, Jeff Sutherland, and Dave Thomas. 2001. Manifesto for Agile Software Development. https://agilemanifesto.org/

[4] Lan Cao and Balasubramaniam Ramesh. 2008. Agile requirements engineering practices: An empirical study. *IEEE Software* 25, 1 (1 2008), 60–67. https://doi.org/10.1109/MS.2008.1

[5] Neil A. Ernst, Alexander Borgida, Ivan J. Jureta, and John Mylopoulos. 2014. Agile requirements engineering via paraconsistent reasoning. *Information Systems* (2014). https://doi.org/10.1016/j.is.2013.05.008

[6] Barney Glaser and Anslem Strauss. 1967. *Grounded Theory: The Discovery of Grounded Theory.*

[7] Rashina Hoda, James Noble, and Stuart Marshall. 2012. Developing a grounded theory to explain the practices of self-organizing Agile teams. *Empirical Software Engineering* (2012). https://doi.org/10.1007/s10664-011-9161-0

[8] Greg Hoff, Ann Fruhling, and Kerry Ward. 2008. *Requirement Prioritization Decision Factors for Agile Development Environments.* Technical Report. http://aisel.aisnet.org/amcis2008/66

[9] Mohamad Kassab. 2014. An empirical study on the requirements engineering practices for agile software development. In *Proceedings - 40th Euromicro Conference Series on Software Engineering and Advanced Applications, SEAA 2014.* Institute of Electrical and Electronics Engineers Inc., 254–261. https://doi.org/10.1109/SEAA.2014.77

[10] Kashumi Madampe, Rashina Hoda, and Paramvir Singh. 2020. Towards Understanding Emotional Response to Requirements Changes in Agile Teams. In *Accepted to New Ideas and Emerging Results track of the 42nd IEEE/ACM International Conference on Software Engineering, ICSE2020.*

[11] Neil Maiden and Sara Jones. 2010. Agile Requirements Can We Have Our Cake and Eat It Too? *IEEE Software* 27, 3 (5 2010), 87–88. https://doi.org/10.1109/MS.2010.67

[12] Sabine Maierhofer, Ernst Stelzmann, Markus Kohlbacher, and Björn Fellner. 2010. Requirement Changes and Project Success: The Moderating Effects of Agile Approaches in System Engineering Projects. In *Communications in Computer and Information Science.* Vol. 99. 60–70.

[13] Zainab Masood. 2017. Self-assignment: Task allocation practice in agile software development. In *Lecture Notes in Business Information Processing.* https://doi.org/10.1007/978-3-319-57633-6{_}22

[14] F. Paetsch, A. Eberlein, and F. Maurer. 2003. Requirements engineering and agile software development. In *Proceedings of the Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, WETICE*, Vol. 2003-Janua. IEEE Computer Society, 308–313. https://doi.org/10.1109/ENABL.2003.1231428

[15] Karl Reed, Ernesto Damiani, Gabriele Gianini, and Alberto Colombo. 2005. *Agile Management of Uncertain Requirements via Generalizations: A Case Study.* Technical Report.

[16] A Strauss and J Corbin. 1998. *Basics of qualitative research techniques.*

[17] Xinyu Wang, Liping Zhao, Ye Wang, and Jie Sun. 2014. The Role of Requirements Engineering Practices in Agile Development: An Empirical Study. In *Communications in Computer and Information Science*, Vol. 432 CCIS. Springer Verlag, 195–205. https://doi.org/10.1007/978-3-662-43610-3{_}15