Self-Assignment: Task Allocation Practice in Agile Software Development

Zainab Masood(✉)

Department of Electrical and Computer Engineering, The University of Auckland, Building 903, 386 Khyber Pass, Newmarket Auckland, Auckland 1023, New Zealand
zmas690@aucklanduni.ac.nz

Abstract. Self-assignment is a self-directed way of task allocation commonly practiced by members of agile teams. However, not much is known about different aspects of self-assignment in literature. This research focuses on two objectives with respect to self-assignment. The first objective is to explore what strategies agile practitioners follow to self-assign tasks of different nature (i.e. new feature, enhancement, and bug-fix). The second objective is to identify the challenges associated with self-assignment and investigate how agile practitioners overcome these challenges to achieve project outcomes. Grounded theory is chosen as the research methodology for this study with data collection through interviewing agile practitioners and observing teams practicing self-assignment. Based on the results, we would propose a theory for self-assignment as a task allocation practice and a set of context-driven guidelines. Knowing the proposed theory and guidelines will help the agile practitioners and companies to make self-assignment a valuable practice in their settings.

1 Introduction

Agile development methodology emerged as an alternative to conventional, sequential and phase-based development. It follows an iterative and incremental approach to development and is open to changes throughout the project [1]. In contrast to traditional development processes, agile offers a different approach to managing the software development cycle. Agile software development constitutes a set of methods and practices based on twelve principles formulated in the Agile Manifesto [2]. The leading agile methodologies (Scrum, XP, Kanban) suggest different strategies and practices to ensure smooth development to achieve project outcomes.

An agile team is a cross-functional group of people who brings a different set of skills to the team. The essence to successful agile teams is their capability to self-organize accompanied by ownership. We find many contributions by researchers made

Supervisor: Dr. Rashina Hoda, The University of Auckland, email: r.hoda@auckland.ac.nz.
Co-Supervisor: Dr. Kelly Blincoe, The University of Auckland, email: k.blincoe@auckland.ac.nz.

H. Baumeister et al. (Eds.): XP 2017, LNBIP 283, pp. 292–297, 2017.
DOI: 10.1007/978-3-319-57633-6_22

exclusively on self-organization and self- organizing nature of the teams [3]. However, there is a dearth of research on how task allocation is done in self-organizing agile teams and what are the common practices followed by agile practitioners to achieve their goals.

Agile methodology uses self-assignment method for the allocation of tasks among team members [4]. However, we do not have enough studies and evidences regarding how software engineers tend to choose these tasks for themselves. There are certain factors that tend to motivate the engineers and developers to prioritize while self-assignment of tasks. During the process of self-assignment, they also have to face issues that need to be addressed for proper allocation and self-assignment. This study will be focusing on mainly these two aspects of self-assignment i.e. strategies and challenges for self-assignment in agile methodology. The contribution will be twofold. Firstly, it will add theoretical knowledge about self-assignment as a way of task allocation. Secondly, the results of this study will benefit developers and managers to overcome challenges during the process of task allocation.

2 Feedback or Areas Seeking Advice

At this time, we are seeking feedback and advice for the following things:

- What is the best way to compare findings from different sources and present overall findings in a way that data integrity is not compromised?
- Advice on reaching theoretical saturation for different task allocation strategies under different contexts.

3 Related Work

The success of a software development project depends heavily on the way the related project management activities are executed [5]. These activities primarily include managing the resources, organizing the software teams, allocating tasks to relevant stakeholders, monitoring time, budget, and resources [4]. These activities are carried out differently depending on the project management approach followed. In traditional software development, a project manager plays a key role in task allocation. The main duty of a project manager is to assign tasks to the project teams. This work is assigned keeping in mind the knowledge, skills, expertise, experience, proficiency and technical competence of the team member [6].

The benefits of agile methodologies include but are not limited to teams empowerment, collaborative atmosphere, shared decision-making and a transparency with a client [4, 7]. In addition, the concepts of ‘light touch’ management and self-organizing teams are the essence of agile teams [1]. These benefits have taken many software firms by a storm, as a result adopting many of these practices in their everyday project management activities including task allocation [4, 8]. This has affected the way the tasks allocation takes place in agile teams. Instead of manager directing or assisting the tasks, these teams are meant to practice picking up tasks or volunteering for tasks [7].

Self-directed task allocation or self-assignment is an attribute of agile teams [4, 8]. In theory, every member of the agile team is meant to assign a task or user story to themself [4]. This method of assigning tasks has also been observed in open source software (OSS) development in both commercial and non-commercial projects [9, 10]. Research on industry practices gives some evidence to support this method of task allocation but how this takes place is not very deeply investigated. For this reason, it is potentially a promising area for study leading to both academic and practical implications.

4 Research Basis

In this study, we intend to explore self-assignment of tasks in agile software development teams. The main research questions governing the research are:

RQ1: How agile practitioners practice self-assignment of tasks? What are the best strategies for self-assigning different types of tasks (new feature, enhancement, bug fixation) in agile software teams?

RQ2: What are the challenges associated with self-assignment of tasks? How agile practitioners overcome these challenges?

5 Research Plans

To answer RQ1 and RQ2, we will focus on how task allocation is played out in agile teams. In particular, this will center on the strategies that teams and individuals undergo in practice using different agile methodologies and for different projects including challenges associated with using these practices. We also plan to study self-assignment as task allocation in different scenarios and contexts and the study will not be limited to a single domain.

- Identifying strategies for tasks of different nature(New feature, Bug Fix, Enhancement)
- Classification of common strategies
- Strengths and weaknesses of the common strategies
- Identifying best strategies in their settings
- Factors affecting self-assignment of tasks
- Comparing self-assignment to alternative methods of task allocation
- Challenges faced with different strategies(threats to autonomy and cross-functionality, complexity and dependency dimensions)
- Identifying the areas of improvements with these strategies
- Evaluating the generated theory using GT guidelines
- Proposing a context-driven set of guidelines which agile practitioners may take into account while self-assigning tasks to get the best out of it.
- If time permits, evaluating the effectiveness of these guidelines through survey based feedback.

6 Research Method

We studied few research methodologies [13] and selected Grounded Theory (GT) for our study [11, 12, 14]. Grounded theory was developed in the early 1960's by Glaser and Strauss. It is chosen as the research methodology mainly due to listed reasons.

- Interest of the researcher towards generating theory explaining how self-assignment is practiced by agile practitioners
- GT is suitable for research areas which have not been explored thoroughly before.
- GT is extensively used for studying agile software teams, human and social aspects of software engineering, and many project management issues [3].
- GT *treats everything as data* giving researcher the freedom to use quantitative data, qualitative data, video, diagrams, and existing theories [14].

Initially, literature and related work are explored generally on identifying how and when tasks are allocated using traditional and non-traditional software methods for software projects. As recommended by Glaser, a minor literature review is conducted in the area of research [12] i.e. self-assignment as a practice of agile teams and individuals. Additionally, we went through articles describing grounded theory in other areas which helped to understand the research methodology and the emergence of the theory from the data [15–17].

As the research is mainly qualitative in nature, the intended data source is semi-structured interviews with agile practitioners of the relevant industry. In terms of data collection, we intend interviewing a total of 40–50 agile practitioners with team observations. But for some parts of the study e.g. factors affecting self-assignment, survey-based data collection will be pursued. Ongoing data analysis and synthesis procedures will be employed on collected data leading to findings of the research. In later stages, when the findings will be sufficiently developed latest and previous related literature will be reviewed again. We intend to assess the generated theory on the basis of four criteria: fit, work, relevance, and modifiability as recommended by Glaser [11].

The main components as adopted by some of the researchers are listed below [14]:

- Data Selection and Collection: Theoretical Sampling (Recruiting participants; Interviews; Observations; Surveys; Questionnaires);
- Data Analysis: Open Coding; Selective Coding; Theoretical Coding; Constant Comparison; Memoing; Sorting; Theoretical Saturation; Generating Theory

7 Validity Threats and Control

The most relevant validity threats to the research along with some checks to be taken to minimize them are given below.

- To reduce researcher bias, we intend to collect data from different sources interviews, observed meetings, and questionnaire. Such data triangulation will help us to generate more substantial data.

- Additionally, to collect multiple perspectives we plan to collect data from different contexts so that we do not limit this study to a particular setting, also we will be interviewing different roles belonging to variant sized organizations working on different software types.
- The supervisor and the co-supervisor, have strong expertise in empirical methods, especially GT and will keep a constant check to make sure that the researcher is not inclined to some side at some point during the study.

8 Current Status

- We have completed an initial round of literature review of related work on task allocation from a pool of papers published between 1990 to 2016 and gathered related work on task allocation generally in software projects and explicitly for agile projects.
- We have also explored some research methodologies to analyze and synthesize the data. After studying few research methodologies we decided to use grounded theory.
- Additionally, we conducted a pilot study to explore self-assignment in agile teams and investigated few aspects associated with it on a relatively small number of agile practitioners. During this study, we found self-assignment to be a potential area for further research as it has not been addressed extensively in the literature. The findings of this study are formulated and submitted to XP 2017 conference and accepted as a short paper ("Exploring Workflow Mechanisms and Task Allocation Strategies in Agile Software Teams") and the social aspects of the study are formulated, submitted to CHASE2017 and accepted as notes paper ("Motivation for Self-Assignment: Factors Agile Developers Consider").
- At present, we are collecting data. We have been successful in gaining some agile practitioner participants and continue to approach others.

References

1. Hoda, R., Noble, J., Marshall, S.: Agile project management. In: New Zealand Computer Science Research Student Conference, vol. 6, pp. 218–221 (2008)
2. Manifesto for agile software development (2001). http://agilemanifesto.org. Accessed 7 Jan 2017
3. Hoda, R., Noble, J., Marshall, S.: Developing a grounded theory to explain the practices of self-organizing Agile teams. Empirical Softw. Eng. **17**(6), 609–639 (2012)
4. Hoda, R., Murugesan, L.K.: Multi-level agile project management challenges: a self-organizing team perspective. J. Syst. Softw. **117**, 245–257 (2016)
5. Pinto, J.K., Slevin, D.P.: Critical success factors across the project life cycle. Proj. Manage. J. **19**(3), 67–75 (1988)
6. Acuna, S.T., Juristo, N., Moreno, A.M.: Emphasizing human capabilities in software development. IEEE Softw. **23**(2), 94–101 (2006)
7. Deemer, P., Benefield, G., Larman, C., Vodde, B.: A lightweight guide to the theory and practice of scrum. Version 2 (2012)
8. Hoda, R., Noble, J.: Becoming agile: a grounded theory of agile transitions in practice. In: IEEE International Conference on Software Engineering (ICSE 2017) (2017)

9. Crowston, K., Li, Q., Wei, K., Eseryel, U.Y., Howison, J.: Self-organization of teams for free/libre open source software development. Inf. Softw. Technol. **49**(6), 564–575 (2007)
10. Kalliamvakou, E., Damian, D., Blincoe, K., Singer, L., German, D.M.: Open source-style collaborative development practices in commercial projects using github. In: Proceedings of the 37th International Conference on Software Engineering, vol. 1, pp. 574–585. IEEE Press (2015)
11. Glaser, B.G.: Basics of Grounded Theory Analysis: Emergence vs. Forcing. Sociology Press, Mill Valley (1992)
12. Glaser, B.G.: Theoretical Sensitivity: Advances in the Methodology of Grounded Theory. Sociology Pr., Mill Valley (1978)
13. Myers, M.D.: Qualitative research in information systems. Manage. Inf. Syst. Q. **21**(2), 241–242 (1997)
14. Stol, K.J., Ralph, P., Fitzgerald, B.: Grounded theory in software engineering research: a critical review and guidelines. In: Proceedings of the 38th International Conference on Software Engineering, pp. 120–131. ACM (2016)
15. Hoda, R., Noble, J., Marshall, S.: Self-organizing roles on agile software development teams. IEEE Trans. Softw. Eng. **39**(3), 422–444 (2013)
16. Adolph, S., Kruchten, P., Hall, W.: Reconciling perspectives: a grounded theory of how people manage the process of software development. J. Syst. Softw. **85**(6), 1269–1286 (2012)
17. Stray, V., Sjøberg, D.I., Dybå, T.: The daily stand-up meeting: a grounded theory study. J. Syst. Softw. **114**, 101–124 (2016)