The Journal of Systems and Software 117 (2016) 245–257

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

Multi-level agile project management challenges: A self-organizing team perspective

Rashina Hoda*, Latha K. Murugesan

SEPTA Research Electrical and Computer Engineering, The University of Auckland, Auckland, New Zealand

ARTICLE INFO

Article history:
Received 15 May 2015
Revised 3 February 2016
Accepted 25 February 2016
Available online 18 March 2016

Keywords:
Agile software development
Project management
Self-organizing teams

ABSTRACT

Agile software development advocates self-organizing teams that display high levels of autonomy. Self-organizing agile teams are meant to share project management activities such as estimation, planning, and requirements elicitation with managers and customers. While prior literature has explored some individual management-related issues, little is known about how the high involvement of self-organizing agile teams influences everyday project management activities. Through a Grounded Theory study involving 21 agile practitioners across six software companies implementing scrum and XP, we identified a set of eight project management challenges as experienced by and as a result of self-organizing agile teams at multiple levels. These include delayed/changing requirements and eliciting senior management sponsorship at the project level; achieving cross-functionality and effective estimations at the team level; asserting autonomy and self-assignment at the individual level, and lack of acceptance criteria and dependencies at the task level. A mapping between the emergent challenges and standard project management activities is also presented. The article also shares practical implications and guidelines for agile teams, their managers, and customers for overcoming some of these challenges.

1. Introduction

Agile software development and agile project management have rapidly gained popularity in the software industry over the past two decades (Begel and Nagappan, 2007; Dingsøyr et al., 2012). Initially emerging as an alternative to traditional, specification-driven methods such as waterfall, agile methods—which embody the core values and principles of agile software development—have rapidly reached a mainstream status in software industries worldwide (Stavru, 2014). This wide-spread adoption is commonly attributed to their ability to respond to rapidly changing business requirements, technologies, and market conditions (Augustine, 2005; Stavru, 2014).

Agile methods emphasize the importance of a collaborative, people-oriented approach to software development (Fowler and Highsmith, 2001). Consequently, agile methods ushered in a number of changes in traditional software development roles and practices. The role of manager or the scrum master on agile projects is to provide adaptive leadership (Augustine, 2005), facilitate the process, remove impediments, and motivate their teams (Chau and Maurer, 2004b). This is in contrast to the traditional role of the manager, following a command and control style of management (Nerur, Mahapatra, and Mangalaraj, 2005). The key functions of a product owner (or customer representative) include maintaining and prioritizing product backlogs (or list of desired product features), specifying and gathering the user-stories/requirements and their associated acceptance criteria (Singh, 2008). Similarly, the role and practices of the software development team have also changed in agile settings. Agile teams are meant to be self-organizing (Fowler and Highsmith, 2001). Agile software development advocates self-organizing teams that display high levels of autonomy (Hoda et al., 2013; Moe et al., 2008). In a self-organizing context, project management activities are meant to be shared between managers, customers, and team members (Hoda et al., 2011b). In practice, as the self-organizing teams become more closely involved with the everyday project management activities such as estimation, planning, and requirements elicitation, a unique set of challenges emerge.

Prior literature has explored some management-related challenges on agile projects (Moe et al., 2009; Nerur et al., 2005). Achieving cross-functionality was seen as an organizational-level barrier to self-management by Moe et al. (2008). Vidgen and Wang (2009) attributed the loss of cross-functionality to the challenges arising from poor self-assignment of tasks, such as

* Corresponding author at: Building 903, 368 Khyber Pass, New Market, Auckland 1023, New Zealand. Tel.: +64 9 923 1377.
E-mail addresses: r.hoda@auckland.ac.nz, rashina@gmail.com (R. Hoda), lmur778@aucklanduni.ac.nz (L.K. Murugesan).
URL: http://www.rashina.com (R. Hoda)

http://dx.doi.org/10.1016/j.jss.2016.02.049
0164-1212/

repeatedly selecting simple tasks. Moe et al. (2008) identified lack of individual commitment as a team-level barrier to self-management while Strode et al. (2015) shed light on how task dependencies can influence project management activities such as estimation and planning. While some such management challenges have been explored in prior literature, little is known about how the self-organizing context of agile teams influences project management. Our aim, therefore, was to explore and illustrate how project management challenges arise as a result of self-organizing teams and how they, in turn, influence the team.

Through a Grounded Theory (GT) study of 21 agile practitioners from 6 software companies, we identified a set of challenges resulting from the high levels of team involvement in project management activities due to their self-organizing context. Data was collected via face-to-face, semi-structured interviews with agile practitioners in companies implementing scrum, or combinations of scrum and XP (eXtreme programming). Analysis was conducted using GT's open coding, selective, and theoretical coding, and constant comparison procedures (Glaser and Strauss, 2009; Glaser and Strauss, 1967). GT is well-suited to study human and social aspects and has been increasingly used in agile teams research (Cockburn, 2003; Coleman and O'Connor, 2007; Hoda et al., 2013; Martin et al., 2009; Whitworth and Biddle, 2007). GT is marked by the absence of a clear research hypothesis upfront and allows the researcher to uncover the primary concern of participants through an iterative and incremental approach, described later in the paper.

A key contribution of our study is to present a grounded theory: *multiple levels of project management challenges*, which describes the project management-related challenges resulting from and experienced by self-organizing teams at multiple levels of project, team, individual, and task on agile projects. We map the identified challenges to standard software project management activities as classified by the Software Engineering Body of Knowledge (SWEBOK). We also highlight some of the relationships that exist between the challenges across the levels. Another contribution is a list of practical implications and guidelines for agile teams, their managers, and customers practicing project management in a self-organizing context.

This paper is organized as follows: Section 2 presents a background on self-organizing agile teams and agile project management. This is followed by Section 3, which specifies our research design including data collection and data analysis via Grounded Theory and a description of participant demographics. This is followed by the Findings in Section 4. Section 5 presents a discussion of how our findings relate to other literature; the multiple levels of the challenges and how they map to standard project management activities; the interdependencies between the challenges across the levels; and implications for practice in the form of potential solutions gathered from related literature. As per Grounded Theory guidelines, related works were referred to after the findings were formulated and are therefore presented in the same order. The paper concludes in Section 6 listing areas for future work.

2. Background

2.1. Self-organizing agile teams

Self-organization is one of the core concepts of agile software development and is one of the twelve principles behind the agile manifesto (Fowler & Highsmith, 2001). While the self-organizing team concept was introduced into software engineering through the agile principles, it has been studied extensively in various disciplines in the past. One of the earliest references to self-organizing teams can be found in socio-technical literature where autonomous groups of English coal-miners were described as self-managing, learning systems composed of 10-15 people sharing responsibilities of former supervisors (Trist, 1981). Principles of self-organization have been described in organizational theory literature as: *minimum critical specification* in terms of high-level guidance and vision provided by senior management while leaving everyday decisions to the teams; *requisite variety* in terms of sufficient variety in the combined skills of the team to cater to a variety of business requirements; *redundancy of function* in terms of the cross-functional ability of the team to supplement and replace each other as required; and *learning to learn* in terms of double-loop learning such that teams not only learn new skills, but also learn new and better ways to accomplish tasks (Morgan, 1986). Takeuchi and Nonaka (1986) define the conditions of self-organizing teams as: *autonomy, cross-fertilization* (or cross-functionality), and *self-transcendence* (or continuous self-improvement.) Morgan refers to *bounded* or *responsible* autonomy where the freedom of decision-making is accompanied by a set of boundaries and responsibilities defined within specific organizational contexts (Morgan, 1986). These descriptions, principles, and conditions have direct bearing on the fundamental definition of self-organizing teams as applied in agile software development (Hoda et al., 2012, 2013).

Self-organized teams in agile software projects are teams that enjoy high levels of autonomy, commit to, select, and accomplish their own tasks, to organize themselves (Hoda et al., 2013). Members on self-organizing teams are known to take on informal, implicit, transient, and spontaneous roles to satisfy various organizing needs of the team (Hoda et al., 2013) and perform a set of 'balancing acts' between freedom and responsibility; cross-functionality and specialization; and continuous learning and iteration pressure (Hoda et al., 2012) to accomplish the principles (Morgan, 1986) and conditions of self-organization (Takeuchi and Nonaka, 1986): autonomy, cross-fertilization; and self-transcendence, respectively. Autonomy in this context is described as the team's ability to define goals, delineate its own identity, secure required resources, and self-organize (Gemünden et al., 2005). Furthermore, autonomy is classified into: individual autonomy which refers to the freedom enjoyed by individuals in undertaking their own tasks (Langfred, 2000); internal autonomy which refers to the extent of shared decision making within the team as opposed to centralized decision making through an individual (Hoegl and Parboteeah, 2006); and external autonomy which refers to the influence of external parties such as management on the affairs of the team (Hoegl and Parboteeah, 2006). Self-organizing teams are also believed to influence productivity (Moe and Dingsøyr, 2008).

2.2. Software and agile project management

The Software Engineering Body of Knowledge (SWEBOK) defines software engineering management as "*the application of management activities—planning, coordinating, measuring, monitoring, controlling, and reporting—to ensure that software products and software engineering services are delivered efficiently, effectively, and to the benefit of stakeholders*" (Abran et al., 2001.) The Project Management Body of Knowledge, PMBOK® and software extension (SWX) (Rose, 2013) use the terms initiating, planning, executing, monitoring and controlling, and closing for these management activities. Irrespective of the lifecycle model, seven topics of management are classified as (Abran et al., 2001): initiation and scope definition (*e.g. requirements determination, negotiation, and review*); software project planning (*e.g. effort, cost and schedule estimation and resource allocation*); software project enactment (*e.g. monitoring process and reporting*); review and evaluation (*e.g. determining satisfaction of requirements*); closure (*e.g. determining closure*); software engineering measurement (*e.g. perform the measurement process*); and software engineering management tools (*e.g. project*

planning and tracking tools). These are referred to later in the findings and discussion sections to describe what specific project management activities were found to be challenged in our study and how.

Management in traditional settings is typically categorized as command and control (Nerur et al., 2005). Team members focus on core software development and engineering practices through well-defined specialist roles such as developers and testers (Brooks, 1975). Project management activities such as requirements elicitation, clarification, and effort estimation are mainly performed by managers or specialists such as business analysts in traditional setups. Work is allocated to team members by their managers.

Let us now consider project management as applied in scrum, the most popular agile method (Deemer et al., 2010). Scrum defines a set of practices which encompass project management activities such as requirements determination, effort estimation, sprint reviews, and retrospectives, and three roles on agile projects as product owner, scrum master, and the self-organizing team (Deemer et al., 2010). A result of these redefined roles in agile methods is that self-organizing teams are increasingly involved in project management activities on an everyday basis (Fowler and Highsmith, 2001). In addition to traditional responsibilities of software development, self-organizing agile team members are meant to display high levels of autonomy and take ownership of the product development and share project management responsibilities such as estimation, planning, requirements elicitation, task allocations, project tracking, and stake-holder collaboration (Fowler and Highsmith, 2001) which were hitherto limited to project managers and technical leaders in traditional settings. The team's increased autonomy and involvement in management practices and decision making is meant to enhance both the speed and accuracy of problem solving by bringing decision making authority to the level of operational problems (Moeet al., 2009). In practice, however, such close involvement of the team in project management activities can lead to other problems (Hoda et al., 2010b). Our aim was to explore and illustrate such challenges and their impact on the teams. Some related works in this area are discussed later in Section 5.1, following Grounded Theory guidelines.

3. Research design

This study adopted the Grounded Theory (GT) method (Glaser and Strauss, 2009; Glaser and Strauss, 1967) and its various procedures for data collection, analysis, and reporting. GT allows the researcher to uncover the primary concerns of the participants through identification of common patterns and themes that emerge from the constant comparison of data across participants at increasing levels of abstraction. The distinguishing feature of GT is the absence of a research hypothesis to be validated through the study, rather the researcher attempts to uncover the main concerns of the participants in the process. In this case, the focus was on the experience of being a self-organizing team and the associated challenges and strategies in real-world settings. GT was employed as the research method for several reasons: (a) GT facilitates research on relatively less investigated areas, and challenges relating to software project management as a whole resulting from self-organizing contexts have not been explored in much depth; and (b) GT is increasingly being employed to study agile software teams and project management issues as it facilitates the investigation of human and social aspects (Cockburn, 2003; Coleman and O'Connor, 2007; Hoda et al., 2013; Martin et al., 2009; Whitworth and Biddle, 2007). The following sub-sections present descriptions of the GT procedures as they were applied in this study, including examples to elucidate their application.

3.1. Data collection

Data were collected from 21 participants in six software companies in India through semi-structured interviews and some observations of work place and practices. The Indian software industry was selected as it is home to a well-established and active agile software community (http://www.agileindia.org/.) Participants were selected based on responses to a general call for participation posted in popular agile user groups and communities. The criteria for selection included agile practitioners in different roles with at least 2 years of experience in practicing agile methods in industrial settings to allow for a range and depth of experiences related to both project management and working in self-organizing team contexts. A majority of the individuals belonged to teams that were supported by management structures and policies conducive to self-organization; and were practicing scrum or a combination of scrum and eXtreme programming. Participants held a wide range of work titles and performed different roles such as developer, tester, delivery manager, scrum master, and project manager which provided a well-rounded perspective.

The interviews lasted for approximately an hour on average and were conducted at the participants' workplaces. Interviews were largely semi-structured addressing some basic questions about professional and agile experience, project background and team composition, self-organizing abilities, challenges encountered, and strategies for addressing those challenges. Table 1 presents the demographics of the participants.

The first column represents participant codes as P numbers from P1 to P21 to refer to different individuals while maintaining confidentiality. The second column lists the roles or job tiles relevant to each participant. Next columns represent the years of professional experience of the participants and agile method used by the participants' teams respectively. The next column denotes the organizations that the participants belonged to, numbered O1–O6. A majority of the participants belonged to the same organization. The team sizes varied from as small as five members to as large as 40 members in a distributed team as listed in the second last column. The team sizes were not directly discernible in some cases where the product boundaries were flexible as members moved around frequently to attend to multiple projects. The last column lists whether the participant had a formal scrum certification such as the scrum master certification.

3.2. Data analysis

We employed Grounded Theory's data analysis and synthesis procedures i.e., open coding and constant comparison method to synthesize data from the interviews by identifying the patterns in the dataset. The word 'open' refers to keeping an open mind when analyzing the data i.e. not being biased by previous literature and/or personal researcher experiences; and 'coding' refers to the task of data analysis (Glaser, 1978, 1992; Glaser and Strauss, 2009). Thus, open coding involves thoroughly analyzing the data to capture as many key points and concepts as possible. Fig. 1b shows the levels of data abstraction achieved in GT data analysis.

To explain these procedures as applied in this study, we present an example of working from raw data from an interview transcript to the resulting theoretical model using one of the categories: *Individual challenges*.

First, we obtained the *key points* from the raw data (interview transcripts). *Key points* are the summarized points from sections of the interview (Georgieva and Allan, 2008). We then assigned a *code* to the key point. A code is a phrase that summarizes the key point in two or three words. One key point can lead to several codes, however, we show a single code example below for simplicity.

Table 1
Participant demographics (P1–P21 used for participants; O1–O6 used for organizations; NE: not explicit; ∗distributed).

Participants	Role	Participant experience	Agile method	Organization	Team size	Scrum certified
P1	Delivery Manager	7–8 years	scrum, XP	O1	6-7	Yes
P2	Scrum Master, Project Manager	1.5 years	scrum	O2	9	Yes
P3	Project Manager/Scrum Master	5 years	scrum	O3	8-10	NE
P4	Scrum Master	5 years	scrum	O4	12	NE
P5	Director-Engineering& scrum Master	5 years	scrum, XP	O5	NE	No
P6	Senior Manager	3 years	scrum	O6	7-10	No
P7	Manager	12 years	scrum, XP	O5	40∗	No
P8	Senior Developer	10 months	scrum	O5	15	No
P9	Project Manager	2–3 years	scrum	O5	6-7	No
P10	Senior Developer	6 years	scrum	O5	5	No
P11	Quality Analyst	7–8 months	scrum	O5	NE	NE
P12	Software Developer	1 year	scrum	O5	NE	NE
P13	Software Tester	8 months	scrum	O5	NE	NE
P14	Testing Lead/Scrum Master	2.5 years	scrum	O5	8-10	Yes
P15	Software Developer	3 months	scrum	O5	8	No
P16	Developer/Tester	8–9 months	scrum	O5	NE	No
P17	Software Developer	4 years	scrum	O5	NE	No
P18	Software Engineering Lead	3 years	scrum, XP	O5	7	No
P19	Project Manager	1.5 years	scrum	O5	7	NE
P20	Module Lead	1.5 years	scrum	O5	7	No
P21	Software Quality Engineer	2.5 years	scrum	O5	7	No

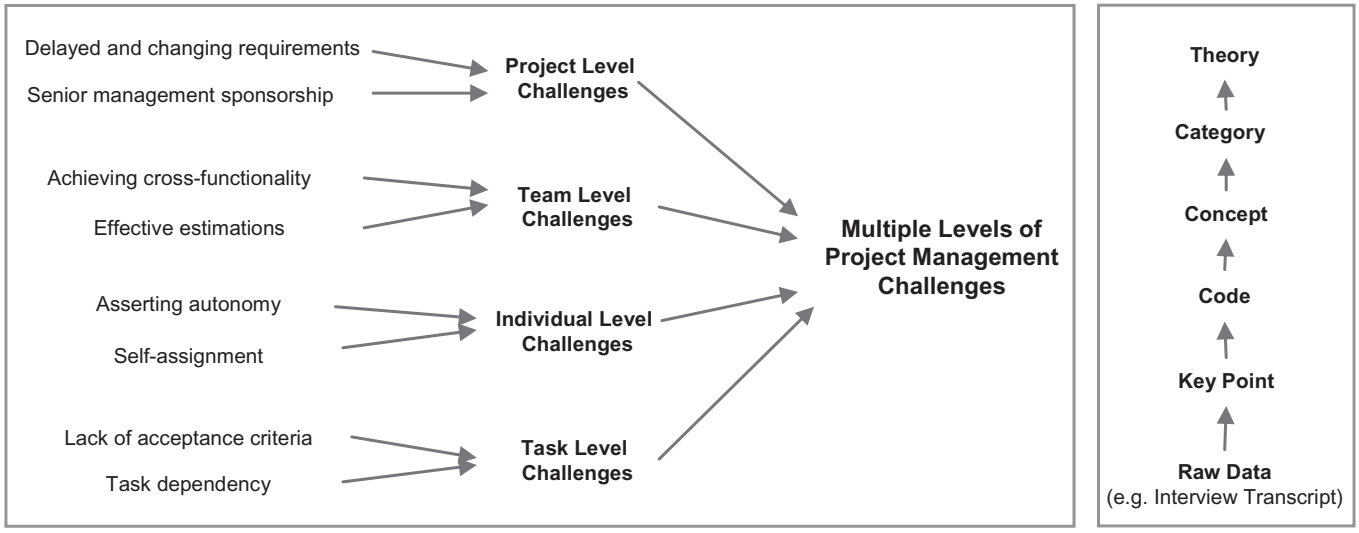

Fig. 1. (a) Emergence of "multiple levels of project management challenges" from underlying categories and concepts. (b) Levels of data abstraction in grounded theory.

Raw data: "But, initially they will not pick, they will pick very simple tasks…"
Key Point: Simple tasks assigned by team members
Code: Task Assignment (*motivation*)

Particular aspects of a code can be captured in brackets as *properties*. In the above example, 'task assignment' was a code derived from the raw transcripts and 'motivation' was captured as a property of the code. In other words, it reminds us that the original context of the raw data was referring to the motivations for task assignment, which in this case was the simplicity of the tasks. The codes arising out of each interview were constantly compared against codes of the same interview, and those from other interviews. This is GT's *constant comparison method*. In this example, another similar code identified was "*cross-functionality*". Using the constant comparison method, we grouped these codes to produce a higher level of abstraction called *concepts*. In this example, the concept that emerged was 'self-assignment' which captured the activity of self-task allocation motivated by simplicity of the task.

Concept: Self-assignment

Another concept that emerged was 'autonomy'. Finally, the constant comparison method was repeated on concepts to produce a third level of abstraction called *categories*. In this example, self-assignment and autonomy were identified as challenges faced on an individual level and thus classified as 'individual-level challenges'.

Category: Individual-level challenges

Similarly, *Project-level challenges, Team-level challenges* and *Task-level challenges* were the other categories identified from the data analysis.

The final step in GT's data analysis process involves conceptualizing how the categories relate to each other to be represented as a theory, referred to as *theoretical coding* (Glaser, 1978). Glaser (2005) has suggested several common structures of theories—or theoretical coding families—that can be used to describe the GT findings. Some theoretical coding families include: the *Six Cs* (contexts, causes, consequences, contingencies, conditions, and covariance); and *process* (e.g. stages or phases). While a comprehensive discussion of these theoretical coding families is beyond the scope of this article, the nature of theory produced by the Grounded

Table 2a
Level classifications from prior literature (Nonaka, 1994; Marrone, 2010; and Moe et al., 2008) and four levels identified in this study.

Nonaka (1994)	Moe et al. (2008)	Marrone (2010)	Hoda and Murugesan (2015)
Inter-Organization			
Organization	Organization		
		Multi-Team	Project
Group	Team	Team	Team
Individual		Individual	Individual
			Task

Theory method and issues of its generalizability are addressed later in Section 5.4. Further details of open coding and other GT procedures can be found in a dedicated paper on the topic (Hoda et al., 2012.)

The theoretical coding family that best suited our findings was *the degrees family* which allows the findings to be represented as *levels*. The four categories—each, representing a level—form the overall theory: *multiple levels of agile project management challenges,* which describes the project management-related challenges resulting from and experienced by self-organizing teams at multiple levels of project, team, individual, and task on agile projects. Fig. 1(a) depicts how the categories emerged from the underlying concepts and formulate the overall theory. The next section describes these levels and the challenges on each level as the main findings of this study.

4. Findings

In this section, we present the findings that describe the common challenges reported by the participants in this study faced as self-organizing agile teams on a regular basis as they performed everyday project management practices. These challenges were found on multiple levels: *project, team, individual,* and *task*-levels. Prior research has classified software organization hierarchy in terms of individual, group, organization, and inter-organization levels when viewed from a knowledge creation perspective (Nonaka, 1994). Moe et al. (2008) have described challenges to self-management at two levels: team and organizational. Another tiered structure employed in the context of describing coordination activities across team boundaries comprised of individual, team, and multi-team levels (Marrone, 2010). Table 2a summarizes these level classifications and presents the four levels identified in our study.

We describe the multiple levels identified in our study as: *project-level* which included project management activities involving the team, scrum master (or manager), and customers; *team-level* which included activities involving only the core development team and their scrum master (or manager); *individual-level* which included activities involving individual members of the team; and *task-level* which included activities involving technical tasks. These are collectively referred to as multi-levels of agile project management. Table 2b presents a summary of these levels, including definitions, and the challenges identified on each level.

4.1. Project level challenges

Project-level challenges included those related to *delayed and changing requirements* and *senior management sponsorship*.

4.1.1. Delayed and changing requirements

An early and important challenge faced at the project-level was associated with requirements elicitation. Adequate understanding of the customer requirements is critical to developing good software. In a self-organizing context, agile teams can no longer rely on project managers or business analysts to work with the customers to procure and manage requirements. The tasks of eliciting, clarifying, and estimating requirements as well as dealing with any changes in requirements now falls with the team. In practice, requirements were sometimes delayed at the customer-end to the extent that teams were not able to proceed without them. In some cases, the teams pulled out other user stories from the backlog and continued working on them while waiting on new requirements or clarifications to arrive. In other cases, a planned sprint would have to be cancelled.

> *If the requirements were not in time, or if some data, where our effort is dependent upon, does not arrive in time, we really cannot do anything.*
>
> *- P7, Project Manager*

Despite their focus on embracing change, some agile teams saw frequently changing requirements as a significant challenge. As a part of the feedback provided by the customers on latest features produced in a sprint, new requirements or changes in existing requirements were introduced. This is very much the norm in scrum projects. However, some participants noted that the requirement changes were not constrained to the beginning of the new sprint cycles but could occur at any time in the project as customers frequently changed their minds.

> *Requirements change at any time of the project development.*
>
> *- P18, Software Engineering Lead*

> *Biggest challenges, I always see that sprint is following approach where the customers keep changing his mind over the requirements.*
>
> *-P8, Senior Developer*

Changing or delayed requirements also adversely affected the team's ability to estimate effort. If the requirements were delayed, it was difficult for the team to estimate the user stories for the current sprint. If the requirements were changed, the estimations needed to be redone.

Table 2b
Multiple levels of project management challenges faced by self-organizing teams.

Levels	Description	Project management challenges identified
Project level	Includes activities involving the self-organizing team, scrum master (or manager), and the product owner (or customer representative.)	• Delayed and changing requirements • Senior management sponsorship
Team level	Includes activities involving only the self-organizing team and their scrum master (or manager.)	• Achieving cross-functionality • Effective estimations
Individual level	Includes activities involving individual members of the team.	• Asserting autonomy • Self-assignment
Task level	Includes activities pertaining to technical tasks.	• Lack of acceptance criteria • Task dependency

We tried to do sizing [effort estimation], but in the meantime the requirements change and everything changes….

-P14, Software Testing Lead/Scrum Master

Usually when the requirements were changed, the revised requirements were often accommodated in the same sprint if they were marked as high priority by the customers.

If it is a high priority, obviously we have to fix it first and we have to adjust the sprint plan accordingly. Sprint may be the same, may be the task is switched over to the next sprint.

-P19, Project Manager

Teams suffered from the negative impact of unsystematic changes in requirements as these changes sometimes led to cancelled sprints. For example, the changes in requirements requested within the current sprint, in some projects, resulted in cancellation of the current sprint, which in turn impacted the overall delivery plan.

Sometimes due to heavy requirements from customer, so we need this feature very urgently, then we need to drop a particular sprint

P18, Software Engineering Lead

If suppose the client says that this is a much needed thing right now and will not go live without this, then, you (we) need to dissolve the sprint

P17, Software Developer

It was interesting to discover this challenge despite clear guidance in agile methods such as scrum for teams to reject any change requests made by the customers *during* a running sprint. All changes are meant to be absorbed and managed in future sprints. However, this was not the case in practice. When teams were probed about why they did not simply reject the change requests as recommended, they explained that often this was the case for 'urgent' needs from the customers which were difficult to ignore or postpone. The situation was likely also effected by the distributed setup of such teams where the teams were located in India and the customers belonged to off-shored companies in the USA or Europe, in these projects. The team culture seemed to encourage compliance to the customers' wishes. In such cases, while undesirable, the teams seemingly had little choice but to 'drop' or cancel their current sprint and re-focus on the required changes.

4.1.2. Senior management sponsorship

The next project-level challenge was eliciting senior management support or project sponsorship. It was difficult for the project managers and the team members to convince their senior management to adopt agile methods and the self-organizing context.

If you want some change, you have to receive acceptance from management as well. But, first thing to convince the management is also a challenge

P2, Project Manager/Scrum Master

One of the primary reasons why the senior management were hesitant in some cases was that they were focused on the final goal and delivery of the product and did not want to risk it in the slightest way. As such, the introduction of a new development process was not encouraged. In other cases, while projects received attention from their senior management with regards to status updates, much remained to be desired in terms of actual involvement and support.

He (reporting manager) was not involved, but he [comes] along with the team, finds out how things are going

P3, Project Manager/Scrum Master

A subsequent challenge was that even when the senior management accepted the adoption of agile methods, such as scrum, they still required their teams to submit relatively heavy documentation such as frequent status reports. Teams perceived this as a challenge to practicing agile which is meant to be a light-weight process involving minimum documentation, and wished for a change in senior management approach in this regard.

And still the senior management are with the notion of status reports and etc. and so, that is still continuing irrespective of the projects. So, I think that, that shift should happen or that is still a challenge

P6, Senior Manager

While the self-organizing context demands increased team involvement in project management activities, the teams frequently require appropriate training to perform them effectively since traditionally these activities were not part of the team's responsibility. However, there were differing opinions held regarding the value of professional training and certification such that while some senior management were actively encouraging professional training and certification, others did not believe in promoting or sponsoring such activities. They believed that it was the manager's responsibility to train their teams on everyday agile practices rather than sponsoring the teams to undergo professional training.

The management says that they are okay to train more people with scrum.

-P2, Project Manager/Scrum Master

If I want to be self-organizing, I should have to somehow learn to motivate/convince my team members as a manager, without spending money or external training.

P1, Delivery Manager

In the latter case, the teams were not able to utilize the support provided through professional training opportunities for transitioning from a traditional to a self-organizing context. The pressure of enabling a smooth transition fell on the shoulders of the managers who in many cases were themselves new to agile practices and self-organizing ways of working.

One of the approaches to eliciting senior management support was to provide them with evidence of successful agile project delivery. Once the management saw that agile methods could be used for successful project delivery, it was easier for the team to elicit their support in the future as the managers now had some confidence in the new methodology and on the new self-organizing team's abilities.

What management wants? You have to convince on how you would deliver the things. So, we have delivered now. They have confidence. This is one area we have to focus on…If you prove something with that and deliver, they would provide some other chance.

-P2, Project Manager/Scrum Master

4.2. Team level challenges

Team-level challenges included *achieving cross functionality* and *effective estimations*.

4.2.1. Achieving cross-functionality

Cross-functionality can be defined as the ability of the team members to work across various technologies and functional areas to achieve the same organizational task (Pinto and Pinto, 1990). Cross-functionality has been described as one of the conditions of self-organization (Takeuchi and Nonaka, 1986). A team is said to possess cross-functionality when (a) it is composed of individual members with varying specializations, thought processes, and behavior patterns and (b) these individuals interact among themselves leading to better understanding of each other's perspectives

(Takeuchi and Nonaka, 1986). It is important for the team members to be cross-functional to minimize personnel dependency, so that the team would be functional despite the absence of particular individuals on some occasions. Cross-functionality, as opposed to specialization, removes dependencies on specialists/individuals and thereby reduces threats to timely product delivery.

While self-organization requires cross-functionality, teams often face the dilemma of selecting between specialization and cross-functionality. On the one hand, specialization allows the team to achieve faster results by harvesting an individual's expertise. Cross-functionality, on the other hand, requires venturing outside areas of expertise and spending time on learning new functionalities. Achieving cross-functionality was easier said than done.

> *But, still one aspect which is missing I see when we started discussing was, rotation is not happening. Same kind of people, they are familiar with certain modules, they are doing only [just] that.*
> *-P5, Director of Engineering/Scrum Master*

This challenge is also related to and affects self-assignment as described later.

4.2.2. Effective estimations

Estimating the effort required in implementing tasks, often done during planning meetings, involves all members of the self-organizing team sharing their views on the estimations. In traditional contexts, effort estimation is typically performed by managers or technical leads and does not typically involve the whole team (Brooks, 1975). In contrast, in a self-organizing context, it is recommended to perform the estimation exercise with the whole team so that individual members can provide their input and perspectives into the estimations. Collaborative estimation exercises were also seen to increase the understanding of requirements expressed as user stories.

> *So, what has actually helped is like the whole team is part of the sizing, the estimation exercise, and everyone is clear about the user stories.*
> *P14, Software Testing Lead/Scrum Master*

It was found that achieving a unanimous estimate was non-trivial. It was seen in some cases that while experienced professionals tended to buffer some extra time for unexpected issues and therefore inflated their estimation, the newer team members tended to underestimate the workload. Consequently, a mismatch arose that needed to be sorted out.

> *First thing is, the estimation was very critical for us… The senior team member says 20 days, the junior says 10 days*
> *P2, Project Manager/Scrum Master*

It was up to the scrum master to identify the reasons behind the estimation mismatch and facilitate a consensus. The scrum master elicited rationales from all the team members regarding their individual estimations and based on the discussion they helped the team resolves mismatches and finalize the estimations.

> *So, I would ask the new guy, how would you complete in 10 days? Then, the new guy would explain everything. Once he finishes, the senior one would be enquired (about) the reason behind 20 days. After this kind of discussion, sometime we end up with up it is not 10 or 20, it is 15 days' job.*
> *- P2, Project Manager/Scrum Master*

It was very important for a scrum master to tackle such problems quickly and achieve better teamwork and trust leading to better performances.

Another interesting challenge related to team estimation was that the sizing/estimation process took considerable time. For example, some teams conducted estimation exercises every week for a maximum of 4 h but still complained that it was difficult for them to accomplish the estimations for all the user stories within that time. So, some of the teams decided to conduct estimation exercises for an hour every day or every other day. In this way, they managed to spend time on estimations while also effectively working on the sprint tasks.

> *The main challenge is firstly that, for sizing, it takes a lot of time.*
> *-P15, Software Developer*

> *But, yeah, sizing actually goes on regularly. Everyday one hour or every two days one hour, team is spending on sizing to clarify all the things.*
> *P14, Software Testing Lead/Scrum Master*

> *Earlier we used to have one sizing meeting every week Friday and that meeting would be for 4 h. And, in those 4 h, we were not able to size lot many user stories… Then, we decided to meet every day for 1 h. So, when we started to meet every day for an hour, then, we were able to size one story and so, over a period of a week, we were able to size 6-7 stories, so that we are ready for the next sprint.*
> *-P15, Software Developer*

4.3. Individual challenges

The next level of challenges included those faced by individuals within teams: *autonomy* and *self-assignment*.

4.3.1. Asserting autonomy

As described earlier, self-organizing teams are meant to display high levels of autonomy and manage their own tasks with minimal involvement from the managers (Hoda et al., 2013; Moe et al., 2008). Better self-management is likely to propagate to a better team and better project outcomes (Moe et al., 2008).

We found that the ability to assert autonomy varied largely across individuals. Some individuals were self-motivated and assumed responsibilities of accomplishing their own tasks.

> *To get a skill, there is one way. I go, take my initiative and get myself skilled.*
> *- P1, Delivery Manager*

However, others struggled to assert individual autonomy and take ownership of tasks as they moved from traditional to self-organizing contexts. In particular, those new to the self-organizing environment continued to work in traditional ways and looked up to managers for directions.

> *They [team members] think that the manager should be responsible but not them.*
> *- P2, Project Manager/Scrum Master*

Transitioning from a manager-driven environment to a context where autonomy was encouraged was seen to require a change in mindset. However, unlike tools and technologies, changing a mindset was acknowledged to be very difficult and to take its own time.

> *We know agile is good because it looks good on resume. After that you need to push yourself further and you need to come out of your comfort zone to learn new things, right. Agile is not learning a new language or something, but it is the change in the mind set. I think that it is the toughest part.*
> *- P1, Delivery Manager*

> *It is very hard to convince the people. People are in master-slave environment most of the time. Because you have to instruct and they will do. But in agile and scrum, everyone is the owner of the product. So, that is the thing. Initially, we find very difficult to convince the people, but later on, it happens.*
> *- P2, Project Manager/Scrum Master*

4.3.2. Self-assignment

Self-assignment is a defining practice of a self-organizing team. Self-assignment can be defined as the ability of every team member to assign a task or user story to themselves (Hoda et al., 2013). This is in contrast to traditional teams were managers are responsible for task allocation. The self-organizing contexts of agile teams effectively modify the traditional software project management practice of task allocation by managers into self-directed task-allocation, or self-assignment of tasks, by team members, which in turn leads to new challenges.

We found that a list of features was initially assigned to a particular team and later the team members in those teams were able to assign the individual tasks to themselves. A task or a user story was self-assigned by an individual through online project management tools such as JIRA. The tool listed all the user stories in the product backlog along with the corresponding team member assigned to them.

> *…we have a list with us that these are the user stories, so firstly, we have assigned them to teams, and later on within the team, they will decide, who will do what and accordingly, we had a tool, where we put it in all our task and user stories.*
>
> *- P9, Project Manager*

> *The team would basically pick up task by themselves after deciding among themselves, who is best to do which things. And, they would also take some decisions on … what is best order to do things.*
>
> *- P4, Scrum Master*

Although, self-assignment sounds simple and promising, it bears its own challenges. The primary challenge was that self-assignment was seen to indirectly threaten cross-functionality in a team. For example, if a team member always selected simple and familiar tasks to assign to themselves it would lead to increased specialization and a loss of cross-functionality in the team.

> *But, initially they will not pick, they will pick simple, simple tasks. So, that was the challenge.*
>
> *- P2, Project Manager/Scrum Master*

In some other cases, the team members picked up the task based on their individual expertise or specializations. In either case, cross-functionality was threatened.

> *If we break down, again to 4 h task or something, and according to your expertise area, you get that assignment.*
>
> *- P1, Delivery Manager*

> *Basically, it (task assignment) is based on the availability and the ability.*
>
> *P10, Senior Developer*

> *Assigning the user stories based on their comfort and based on the skills they have.*
>
> *- P14, Software Testing Lead/Scrum Master*

In some other cases, the task assignment was done with the help of the scrum master or the project manager. Even in such cases, the task assignment was generally driven by team members' previous experience.

> *Assignment is that they are now deciding on their own and scrum master is there to help as and when required. And, they mainly go by, whatever they had done similar kind of work previously.*
>
> *- P5, Director of Engineering/Scrum Master*

The participants reported that the main consequence of self-assignment driven by specialization (as opposed to cross-functionality), comfort and prior experience was that some of the tasks remained unassigned. In such cases, it is the responsibility of the scrum master or the project manager to discuss them with the team members and facilitate the assignment of the remaining tasks.

> *If somebody says that I am not interested and I wouldn't take that, then we will sit and talk.*
>
> *- P1, Delivery Manager*

One of the main reasons leading to the self-assignment challenge, as reported by some participants, was that the task requirements may have been unclear or may be missing their corresponding acceptance criteria and so the individuals were hesitant to select such tasks.

4.4. Task level challenges

Finally, we encountered some challenges that applied at the low, task-level: problems pertaining to particular user stories and tasks. In particular, they included: *lack of acceptance criteria* and *task dependency*.

4.4.1. Lack of acceptance criteria

Every user story is meant to have their associated acceptance criteria: a checklist created by the customer that defines the completeness and accuracy of a well implemented user story (Deemer et al., 2010). Teams note the acceptance criteria corresponding to each user story in their project management tools such as JIRA which helps provide a clear focus for every story.

> *When you have acceptance criteria, at least you have focus.*
>
> *- P6, Senior Manager*

Some teams struggled to implement the functionalities and tasks as their customers did not provide clear acceptance criteria.

> *One of the aspect that people not discussing is that they are not clear about it [functionality], or they are careless about it, and the main trigger for that is the lack of acceptance criteria.*
>
> *- P6, Senior Manager*

Much like the challenge of eliciting requirements, this issue is also related to the increased customer-team interactions required in a self-organizing work environment. Unlike a traditional software team, agile teams are required to not only elicit requirements directly from their customers but also seek clarifications and acceptance criteria. Requirement elicitation and clarification demand strong inter-personal communication and negotiation skills that take time to acquire as team members transition from traditional to self-organizing contexts (Hoda et al., 2010b.)

4.4.2. Task dependency

Another important task related challenge is the task dependency that exists between the tasks within a sprint or across sprints. Strode (2015) and Crowston (1994) categorize dependencies into task, resource, knowledge, and technical-based dependencies. Task dependencies refer to situations where the completion of one task is necessary for the next to begin (Strode, 2015) and that is the type of dependency we refer to in this paper.

The main consequence of task dependency was cancellation of sprints.

> *There is one module, which has dependency on third party. That leads to the cancellation of sprints a lot of time.*
>
> *- P7, Project Manager*

Cancellation of sprints could easily result in degrading the performance and velocity of the project and delaying the product delivery. The issue of unresolved task dependencies can be seen to follow from the redefinition of the team and manager roles in

a self-organizing context. For instance, in a self-organizing team, individuals are held responsible for achieving personal tasks but the responsibility of actively assessing and addressing dependencies between tasks can be assumed to be an emergent outcome of the process, typically achieved in the initial planning and requirements definition stages, with no clear role-assignment. Depending on the nature and extent of dependencies, such assumptions can prove problematic if the task dependencies are not actively identified, monitored, and resolved.

As we have seen from the challenges identified, the ability of a self-organizing agile team to perform various project management practices is challenged on multiple levels, i.e., project, team, individual and task-levels. In other words, high levels of involvement of the self-organizing team in project management activities has effects on multiple levels.

5. Discussion

5.1. Related works

While agile project management as a whole has not been investigated in industrial settings from a self-organizing perspective before, various individual project management challenges on agile projects have been discussed in the past (Moe et al., 2009; Nerur et al., 2005). Here we present some such challenges on the project, team, individual, and task-levels relevant to this study and its findings.

On the project level, we found that responding to delayed and changing requirements was a constant struggle for teams. Direct and frequent interaction with customers exposes the self-organizing team to potentially unreasonable and unsystematic change requests from the customers. This was evidenced in Indian teams where it is known that cultural values encourage compliance in general and acceptance of highly volatile change requests in particular (Abraham, 2009; Hofstede and Hofstede, 2001). As such, when faced with the dilemma of following scrum guidelines of no changes *within* sprints on the one hand and abrupt customer demands for 'urgent' changes on the other, teams were seen to succumb to the pressures of pleasing customers. This highlights the need for the team to be provided some shielding/protection by the manager or the scrum master in cases where customers may be unintentionally or consciously exploiting the agile values of embracing change. Such frequent requests for changes also hint at a lack of common product vision which is meant to be established between the customer and the team.

Another project-level challenge was eliciting senior management sponsorship. Prior literature has linked senior management support to successful self-organizing practice (Hoda et al., 2011a). To this end, a specific *champion* role was identified as one responsible for eliciting senior management support in self-organizing contexts (Hoda et al., 2013). Senior management was found to be typically quite keen to support the managers and their teams once they were provided with sufficient evidence of progress with regards to adoption of the new methodology (Turner et al., 2010). This observation is echoed in our study as providing evidence of success often helped gain senior management support. However, our study discovered that in some cases the support did not permeate through to management practices and traditional expectations of status-reporting and heavy documentation persisted despite general support.

We identified effective estimation as a team-level challenge. In this context, Haugen (Haugen, 2006) suggested that planning poker can improve estimation performance when compared to other unstructured group estimation processes. In other words, perfectly measured estimations and planning is likely to enable teams to better respond to changing requirements. However, achieving effective estimations, as we found in our study, is non-trivial. Further research efforts are required to better understand the challenge of performing effective estimation in the face of changing requirements.

On the individual-level, team members are meant to self-allocate tasks and take responsibility for their completion. Ideally, neither the manager nor the scrum master assigns tasks to them. This is meant to help motivate the team members. An associated challenge is that of losing cross-functionality in the team if individuals repeatedly select familiar and simple tasks (Vidgen and Wang, 2009). We found evidence to suggest that this threat to cross-functionality existed and some amount of the scrum master's non-invasive surveillance and support was necessary to maintain a healthy balance between cross-functionality and specialization.

Achieving cross-functionality on the team-level was found to be difficult in other studies (Moe et al., 2008) where high levels of specialization proved to be barrier to self-organization. Some level of balance has been suggested as a useful practice to resolve the dilemma of specialization versus cross-functionality (Hoda et al., 2012).

Autonomy has often been discussed as a key aspect of self-organization in literature (Moe et al., 2008; Takeuchi and Nonaka, 1986). Both the issue of type as well as amount of autonomy has been discussed (Janz et al., 1997). As discussed earlier, individual, internal (or team), and external (or stakeholder influence on team) autonomy are important in a self-organizing team (Hoegl and Parboteeah, 2006; Langfred, 2000). In our study, we found that the lack of external autonomy was impeding the team's ability to be self-organizing on account of the senior management still expected heavy documentation in some cases. This is similar to the observations of Moe et al. (2008) who identified reduced external autonomy as a critical barrier to self-organization. While they observed high individual autonomy as a barrier to internal/team autonomy, we found individual autonomy to be restricted as people new to agile struggled to take on ownership of tasks and frequently looked up to their managers for guidance for a while before embracing the freedom afforded to them by their context. Possibly, this can be explained on account of India's low individualism score (IDV) as per Geert Hofstede™ study (Hofstede, 2001) and as confirmed in software development contexts (Abraham, 2009). Additionally, we found that the lack of external and individual autonomy posed problems for effective project management. While the self-organizing context required increased involvement in project management activities, reduced external and individual autonomy inhibited effective involvement and contribution by the team and individuals.

Defining and performing acceptance tests is one of the many responsibilities of a product owner (Deemer et al., 2010). In practice, the lack of acceptance criteria has been identified as a critical challenge for teams as it is difficult to estimate/size user-stories in their absence (Lee, 2008). Our study corroborates this observation and highlights the lack of acceptance criteria as a contributing factor to the teams' incomplete understanding of requirements which in turn leads to diverse cascading problems such as inaccurate estimation and rework at the task-level. A deeper investigation is required to better understand the role of personal characteristics in the success of practices such as customer collaboration which demand self-motivation, self-discipline, and effective communication and negotiation skills.

Another important challenge at the task-level is the dependency within and across the tasks in the project. The major drawbacks of this issue in agile projects include prolonged estimation efforts. It is indeed essential to reduce the dependencies inside/across the task to enable faster and easier development, testing and release. The study by Strode et al. (2012) recommended that dependency

can be easily addressed by means of scrum/XP practices such as sprint planning meetings, daily scrum meetings, and sprint review meetings. Our study discloses that dependencies, extending beyond the team to third-party vendors, raises severe challenges such as sprint cancellations and also impacts the efficiency of estimation and planning. In the software industry where dependencies on external vendors (Crowston, 1994; Strode, 2015) is a common reality, this issue needs further empirical investigation.

5.2. Multiple levels of project management challenges

Extant literature on agile project management describes some challenges faced on various levels as discussed above. However, little is known about their relationship to the self-organizing context of the agile teams. Our study of agile teams addressed this research gap by presenting a grounded theory: the multiple levels of project management challenges faced by self-organizing agile teams. In particular, we assert that it is because of the self-organizing nature of agile teams that they become increasingly involved in standard project management activities which were hitherto limited to project managers in traditional settings. The involvement of the team in these activities in turn leads to a plethora of challenges as described in our findings section. In other words, new project management challenges arise as a result of the increased involvement of the self-organizing team into everyday project management activities.

We further mapped the challenges identified in our study with software project management activities, such as initiation and scope definition and software project planning (Abran et al., 2001), as described earlier in Section 2.2. The project management challenges identified in our study map to SWEBOK's standard software engineering management topics and activities (Abran et al., 2001), highlighted in italicized text, as follows:

On the project-level, the challenge of delayed and changing requirements adversely impacted the project delivery schedule which is covered under the topic of *software project planning;* and also involved software project management activities of requirements determination and negotiation with the customer, which are covered under the topic of *initiation and scope definition.* The other project-level challenge of senior management sponsorship involved the management's hesitance to experiment with new methodologies on account of the perceived risk to the project delivery schedule—a concept covered under the topic of *software project planning*; and also impacted teams as they were required to present heavily documented status reports, which is covered under the topic of *software project enactment*.

On the team-level, the challenge of achieving cross-functionality related to the concept of effective resource allocation, which is covered under the topic of *software project enactment*. The other team-level challenge of effective estimations related directly to effort, cost, and schedule estimation as covered under the topic of *software project planning*.

On the individual-level, the challenge of asserting autonomy is seen to be related to nearly all aspects of software project management such as *software project planning, enactment, measurement* etc. as it requires the individuals to take responsibility and ownership of all tasks involved in these activities. It can said that autonomy is an imperative condition for self-organization and proper execution of software project management in a self-organizing context. The other individual-level challenge of self-assignment is related to a new model of resource allocation where individual team members self-allocate themselves to tasks; resource allocation in turn is covered under the topic of *software project planning*.

On the task-level, the challenge of lack of acceptance criteria is directly related to determining satisfaction of requirements which is covered under the standard project management topic of *review and evaluation*. The other task-level challenge of task dependency is seen to relate to the lack of identifying dependencies in the initial planning sessions—covered under the standard topic of initiation and scope definition; and to the lack of actively monitoring dependencies with the aim of resolution—covered under the topic of *software project enactment*.

Fig. 2 presents a model that depicts the emergent grounded theory: multiple levels of project management challenges in self-organizing contexts. It shows the four levels identified as project, team, individual, and task-levels, in circles; the roles involves in each level; the challenges identified in this GT study at each level; and a mapping to the related software project management activities (based on SWEBOK, Abran et al., 2001) challenged at each level.

Another interesting outcome of this study is the identification of project management challenges on multiple levels such as team, individual, task, besides the typical project level. This multi-level structure serves to emphasize the increased involvement of agile teams in everyday project management activities (note presence of the team or individual team members on all levels in Fig. 1). For example, in traditional software projects where the project manager is responsible for most project management activities, the team and individuals are relatively screened from the challenges related to the activities on the team and project levels (e.g. effective estimations, eliciting requirements, and senior management sponsorship.) In a self-organizing context, however, project management related challenges were found to permeate into all levels as the project management activities are shared between managers and team members. Heightened exposure of the team to the customers can leave the team vulnerable to customer-based challenges such as unsystematic change requests. Some well measured or balanced intervention by the manager or scrum master may be required in such cases. In other words, an ideal form of self-organization is difficult to achieve in the real world and in practice, managers may need to move between a hands-off and a hands-on approach depending on the specific contexts. Furthermore, our study suggests that the traditional definition of project management and the role of the manager on agile projects need major recalibration so that both the managers and their teams are aware of their respective responsibilities and role boundaries. This is necessary in order to ensure smooth and effective functioning of the self-organizing team.

We also found that challenges on one level were related to and affected those on other levels. We summarize these relationships here:

- Delayed or changing requirements at the project level have an adverse effect on the practice of estimation at the team level. That is, when the requirements from the customers are delayed, effort cannot be estimated and assumptions are made which are either under-estimated or over-estimated. Also, if the requirements are changed, the time spent on the team estimation is prolonged due to rework.
- Lack of senior management support at the project-level leads to reduced external autonomy at the individual level. This was exhibited in the form of senior management expecting heavy documentation and status reports from agile teams thereby reducing their agility and autonomy.
- The challenge of achieving cross-functionality at the team level influenced the challenge associated with self-assignment at the individual level and vice-versa. If the team valued or focused on specialization as opposed to cross-functionality, individuals were more likely to self-assign tasks based on their expertise rather than exploring new areas which require learning. Similarly, the more the individuals self-assigned tasks based on expertise and experience (i.e. specialization), the more the team

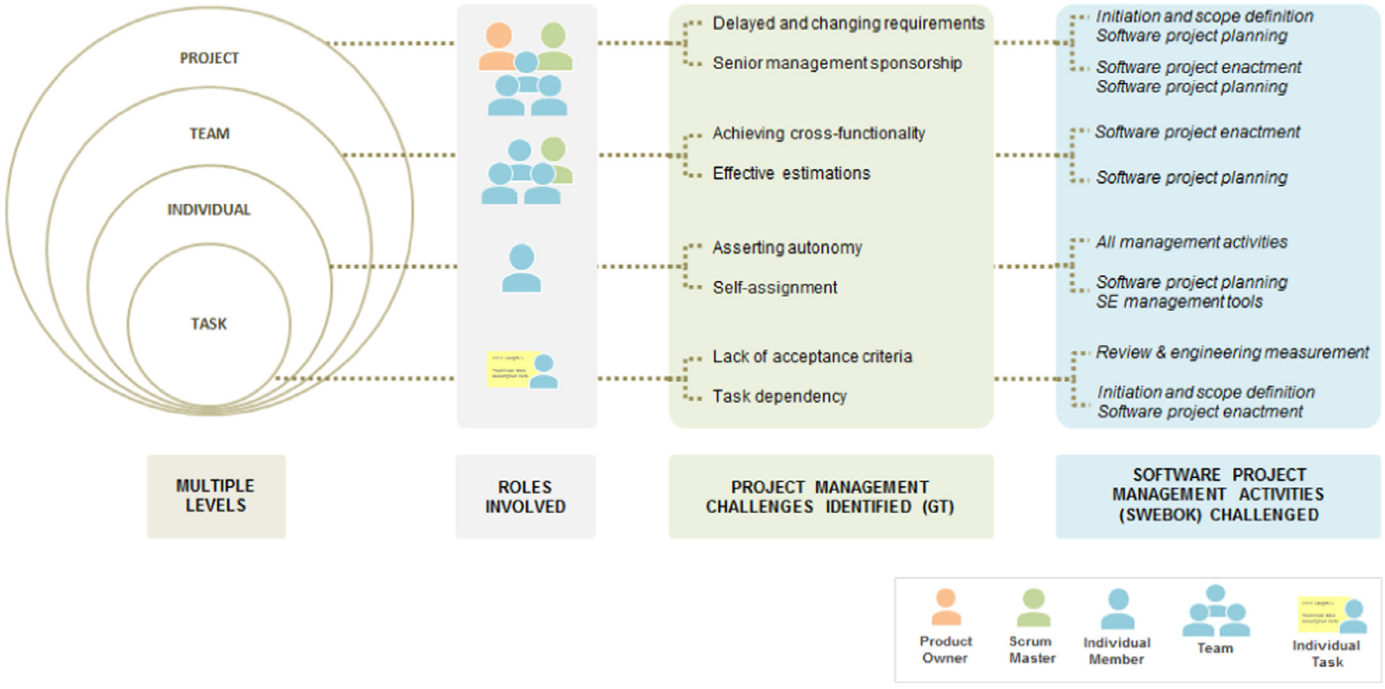

Fig. 2. Multiple levels of project management challenges identified; mapped to the roles involved at each level, and the SWEBOK Software Project Management Activities (Abran et al., 2001) affected.

as a whole was likely to be specializing rather than be cross-functional.
- Task dependency on the task-level influenced the challenge of effective estimations at the team level. Dependency on third parties led to delays and cancellations which in turn had an adverse effect on the team's ability to estimate effort, requiring re-sizing and re-work.

5.3. Implications for practice

Our study focused on the multi-level project management *challenges* faced by self-organizing teams. Fostering self-organizing agile team requires overcoming these challenges. We now present an overview of some of the recommended *solutions* proposed in related literature for overcoming the challenges identified in our study as well as make other recommendations for practice.

- ***Holistic approach to planning:*** Delayed or changing requirements in an agile project are one of the project level challenges (described in Section 4.1.1). One approach to overcoming this challenge is the use of planning to establish a common product vision between the customer and the team so that excessive change requests and urgent, *within* sprint change requests can be avoided. While agile methods recommend against a Big Design Up Front (Cao et al., 2004), some amount of long-term release planning during the initial project estimation or sizing is desirable in practice (Therrien, 2008). Once the estimation and plan is based on a longer term view (e.g. beyond a few sprints), the dependency among the features can be significantly minimized thereby alleviating the task dependency challenge to a reasonable extent (Mak and Kruchten, 2006).
- ***Knowledge sharing and reflective practice is key to achieving cross-functionality:*** A significant challenge at the team level is achieving cross-functionality (described in Section 4.2.1). One strategy to overcome this issue is knowledge sharing. With the help of a scrum master or an expert consultant, the team benefits from reflecting upon and identifying its own strengths and weaknesses, which would help the team to achieve enhanced communication; build trust among the team members; and ensure that the team is equipped to execute the plan (Therrien, 2008). It also helps increase knowledge redundancy in the team thereby assisting the team in *achieving cross-functionality*(Mohamed et al., 2004).Ideally, each team member's interest in learning new things helps achieve cross-functionality to a great extent. However, some cases call for a different strategy: task delegation instead of self-assignment in cases where team members tend to repeatedly select tasks which they are comfortable with instead of selecting highest priority tasks. In such cases, either direct or random task delegation by the scrum master or project manager would enforce the development of cross-functionality until such members become used to performing a variety of tasks and are ready to self-select more responsibly.
- ***Harnessing technology:*** Tools play a critical role in the effective implementation of any software methodology. Organizations planning to implement agile methodologies should consider investing in tools that facilitate and support rapid iterative development (Nerur et al., 2005). There are several tools available in the market to help agile teams practice, including JIRA, Trello, Scrumwise, and Acunote and other research-based tools such as MASE (Modeling Assisted Software Environment) (Chau and Maurer, 2004a). Furthermore, the team will benefit from training on how to best use the tools.
- ***Effective communication:*** is one of the key factors that integrate agile practices within teams. Diverse individuals including developers, end-users, and stakeholders go through iterations of "*thought-action-reflection*" that improves learning and adaptation (Nerur et al., 2005). Effective communication and co-ordinations also seen to assist in acquiring senior management support and successful team estimations. Effective communication and co-ordination among the team helps in estimating the tasks in a better way, which in turn aids successful project delivery. The successful project delivery on time would raise

the credibility of the team among the senior managers (Strode et al., 2012).

- ***Role of the agile project manager:*** We recommend a thorough investigation of the role of the project manager in self-organizing contexts. Our findings show an increased involvement of the team in everyday project management activities such that the role and responsibilities of the traditional project manager are now shared with the team and individual members. The boundaries between the role of the project manager and that of the team, however, were found to be blurred in many cases. For example, in some cases, it was unclear as to who was responsible for certain project management activities such as dependency identification, monitoring, and resolution, and such activities could easily slip into the gaps. This has implications for better defining the roles and responsibilities of the project managers and the team members. Furthermore, it also impacts how the project manager is perceived and expected to function: a high level of compatibility and trust is imperative for the project manager and team to execute project management activities collaboratively and effectively. These aspects need to be taken into consideration from a human resource management perspective when hiring and forming self-organizing teams.

5.4. Evaluation and limitations

A Grounded Theory research study is said to produce a "mid-ranged" theory, which means that while the theory is not claimed to be universally applicable, it can be modified by constant comparison to accommodate more data from new contexts (Glaser, 1992.) As such, generalizability is one of the inherent limitations of Grounded Theory studies as the findings can only be said to explain the contexts investigated (Hoda et al., 2013.) In other words, the findings are grounded in the data and its contexts which in turn were restricted by the available participants. Also, the adoption, practice, and customization of agile methods and specific agile practices varies largely in the software industry worldwide (VersionOne, 2015) and 'standard' or by-the-book agile software development is rarely encountered (Hoda et al., 2010a). However, what is more relevant to evaluating the validity and reliability of a GT study is its ability to be modified to fit, work, and be relevant in different contexts (Glaser, 1992). For instance, the multi-level project management challenges can be potentially applied and modified to fit in other areas and disciplines where a self-organizing context may be present, for example in the open-source community. Comparison of GT findings with those from prior literature also serve to further validate them as synergies with similar concepts may be discovered or new insights on established concepts may have been found as evident in our case and discussed in Section 5.1 earlier.

6. Conclusion

With the self-organizing team's increased level of involvement in various project management activities such as estimation, planning, and requirements elicitation, a unique set of challenges arise at multiple levels on agile projects. Based on a Grounded Theory study of 21 participants from six different companies, this paper presented the issues and constraints associated with practicing agile project management in a self-organizing team context. In general, these fall into four main categories: project, team, individual and task level challenges. The eight main challenges found to exist were: *delayed/changing requirements and eliciting senior management sponsorship* at the project level; *achieving cross-functionality* and *effective estimations* at the team level; asserting *autonomy* and *self-assignment* at the individual level; and *lack of acceptance criteria* and *dependencies* at the task level. These collectively constitute a grounded theory: multiple levels of project management challenges which are experienced by and as a result of self-organizing teams on agile projects.

Customer-related challenges such as delayed and changing requirements and the lack of acceptance criteria point at the need to better understand the new dynamics of increased team-customer interactions resulting from a self-organizing environment and the role of the manager in protecting the team in such contexts.

Several of these challenges hint at the lack of clarity in the role of the project manager on self-organizing teams as teams struggle with issues that were traditionally handled by project managers while the project managers struggle to find the right balance between empowering their teams with autonomy on the one hand and guiding and assisting their teams when required on the other.

Recommendations for overcoming some of these challenges were also discussed based on related literature. Future studies will need to focus in-depth on specific challenges, along with strategies to overcome them. Some areas that demand immediate and in-depth investigation include: role of personal characteristics in asserting autonomy in practice; strategies for addressing risks arising from external dependencies; the role of team and national culture in relenting to customer demands for unsystematic changing requirements; and role of managers in providing the optimum level of guidance and support while maintaining team autonomy and self-organizing ability.

Acknowledgments

Our sincere gratitude to all those who participated in this research. This study is supported by a FRDF-NS fund (#3703888) from The University of Auckland.

References

Abraham, L.R., 2009. Cultural differences in software engineering. In: Proceedings of the Second India Software Engineering Conference (ISEC '09). New York, NY, USA. ACM, pp. 95–100.

Abran, A., Bourque, P., Dupuis, R., Moore, J.W., 2001. Guide to the software engineering body of knowledge-SWEBOK. IEEE Press.

Augen, N.C., 2006, July. An empirical study of using planning poker for user story estimation. In: Agile Conference, 2006. IEEE, p. 9.

Augustine, S., 2005. Managing Agile Projects. Prentice Hall PTR.

Begel, A., Nagappan, N., 2007, September. Usage and perceptions of agile software development in an industrial context: an exploratory study. In: First International Symposium on Empirical Software Engineering and Measurement, 2007. ESEM 2007. IEEE, pp. 255–264.

Brooks, F.P., 1975. The Mythical Man-Month, 1995. Addison-Wesley, Reading, MA.

Cao, L., Mohan, K., Xu, P., Ramesh, B., 2004, January. How extreme does extreme programming have to be? Adapting XP practices to large-scale projects. In: Proceedings of the 37th Annual Hawaii International Conference on System Sciences, 2004. IEEE, p. 10.

Chau, T., Maurer, F., 2004. Knowledge sharing in agile software teams. Logic Versus Approximation. Springe, Berlin, Heidelberg, pp. 173–183.

Chau, T., Maurer, F., 2004. Tool support for inter-team learning in agile software organizations. In: Advances in Learning Software Organizations. Springer, Berlin, Heidelberg, pp. 98–109.

Cockburn, A., 2003. People and methodologies in software development. Fac. Math. Nat. Sci.

Coleman, G., O'Connor, R., 2007. Using grounded theory to understand software process improvement: a study of Irish software product companies. Inf. Softw. Technol. 49 (6), 654–667.

Crowston, K., 1994. A Taxonomy of Organizational Dependencies and Coordination Mechanisms. Center for Coordination Science, Alfred P. Sloan School of Management, Massachusetts Institute of Technology.

Deemer, P., Benefield, G., Larman, C., Vodde, B., 2010. The scrum primer. Scrum Primer is an indepth introduction to the theory and practice of Scrum, albeit primarily from a software development perspective. available at http://assets.scrumtraininginstitute.com/downloads/1/scrumprimer121.pdf, 1285931497, 15.

Dingsøyr, T., Nerur, S., Balijepally, V., Moe, N.B., 2012. A decade of agile methodologies: Towards explaining agile software development. J. Syst. Softw. 85 (6), 1213–1221.

Gemünden, H.G., Salomo, S., Krieger, A., 2005. The influence of project autonomy on project success. Int. J. Project Manage. 23 (5), 366–373.

Georgieva, S., Allan, G., 2008. Best practices in project management through a grounded theory lens. Electron. J. Bus. Res. Methods 6 (1), 43–52.
Glaser, B., 2005. The Grounded Theory Perspective III: Theoretical Coding. Sociology Press, Mill Valley, CA.
Glaser, B., 1978. Theoretical Sensitivity: Advances in the Methodology of Grounded Theory. Sociology Press.
Glaser, B., 1992. Emergence vs Forcing: Basics of Grounded Theory Analysis. Sociology Press.
Glaser, B., Strauss, A., 1967. The Discovery Grounded Theory: Strategies for Qualitative Inquiry. Chicago, Aldin.
Glaser, B., Strauss, A.L., 2009. The Discovery of Grounded Theory: Strategies for Qualitative Research. Transaction Publishers.
Fowler, M., Highsmith, J., 2001. The agile manifesto. Softw. Dev. 9 (8), 28–35.
Hoda, R., Kruchten, P., Noble, J., Marshall, S., 2010a, October. Agility in context. In: ACM Sigplan Notices, 45. ACM, pp. 74–88.
Hoda, R., Noble, J., Marshall, S., 2010b. What language does agile speak? Agile Processes in Software Engineering and Extreme Programming. Springer, Berlin, Heidelberg, pp. 387–388.
Hoda, R., Noble, J., Marshall, S., 2011a. Supporting self-organizing agile teams. Agile Processes in Software Engineering and Extreme Programming. Springer, Berlin, Heidelberg, pp. 73–87.
Hoda, R., Noble, J., Marshall, S., 2011b. The impact of inadequate customer collaboration on self-organizing Agile teams. Inf. Softw. Technol. 53 (5), 521–534.
Hoda, R., Noble, J., Marshall, S., 2012. Developing a grounded theory to explain the practices of self-organizing Agile teams. Empirical Softw. Eng. 17 (6), 609–639.
Hoda, R., Noble, J., Marshall, S., 2013. Self-organizing roles on agile software development teams. IEEE Trans. Softw. Eng. 39 (3), 422–444.
Hoegl, M., Parboteeah, P., 2006. Autonomy and teamwork in innovative projects. Hum. Resour. Manage. 45 (1), 67–79.
Hofstede, G.H., 2001. Culture's Consequences: Comparing Values, Behaviors, Institutions and Organizations across Nations. Sage.
Janz, B.D., Colquitt, J.A., Noe, R.A., 1997. Knowledge worker team effectiveness: The role of autonomy, interdependence, team development, and contextual support variables. Personnel Psychol. 50 (4), 877–904.
Langfred, C.W., 2000. The paradox of self-management: individual and group autonomy in work groups. J. Organizational Behav. 21 (5), 563–585.
Lee, E.C., 2008, August. Forming to performing: transitioning large-scale project into agile. In: Agile 2008 Conference. IEEE, pp. 106–111.
Mak, D.K., Kruchten, P.B., 2006, May. Task coordination in an agile distributed software development environment. In: Canadian Conference on Electrical and Computer Engineering, 2006. CCECE'06. IEEE, pp. 606–611.
Marrone, J.A., 2010. Team boundary spanning: a multilevel review of past research and proposals for the future. J. Manage. 36 (4), 911–940.
Martin, A., Biddle, R., Noble, J., 2009, August. The XP customer team: a grounded theory. In: Agile Conference, 2009. AGILE'09. IEEE, pp. 57–64.
Moe, N.B., Dingsøyr, T., 2008. Scrum and team effectiveness: Theory and practice. In: Agile Processes in Software Engineering and Extreme Programming. Springer, Berlin, Heidelberg, pp. 11–20.
Moe, N.B., Dingsoyr, T., Dyba, T., 2008, March. Understanding self-organizing teams in agile software development. In: 19th Australian Conference on Software Engineering, 2008. ASWEC 2008. IEEE, pp. 76–85.
Moe, N.B., Dingsoyr, T., Dyba, T., 2009. Overcoming barriers to self-management in software teams. IEEE Softw. 26 (6), 20–26.
Mohamed, M., Stankosky, M., Murray, A., 2004. Applying knowledge management principles to enhance cross-functional team performance. J. Knowl. Manage. 8 (3), 127–142.
Morgan, G. (1986). Images of Organization, Sage Publications, Beverly Hills.
Nerur, S., Mahapatra, R., Mangalaraj, G., 2005. Challenges of migrating to agile methodologies. Commun. ACM 48 (5), 72–78.
Nonaka, I., 1994. A dynamic theory of organizational knowledge creation. Organization Sci. 5 (1), 14–37.
Pinto, M.B., Pinto, J.K., 1990. Project team communication and cross-functional cooperation in new program development. J. Product Innovation Manage. 7 (3), 200–212.
Rose, K.H., 2013. A guide to the project management body of knowledge (PMBOK® guide)—Fifth Edition. Project Manage. J. 44 (3) e1–e1.
Singh, M., 2008, August. U-SCRUM: an agile methodology for promoting usability. In: Agile, 2008. AGILE'08. Conference. IEEE, pp. 555–560.
Stavru, S., 2014. A critical examination of recent industrial surveys on agile method usage. J. Syst. Softw. 94, 87–97.
Strode, D.E., 2015. A dependency taxonomy for agile software development projects, pp. 1–24. doi:10.1007/s10796-015-9574-1, In press.
Strode, D.E., Huff, S.L., Hope, B., Link, S., 2012. Coordination in co-located agile software development projects. J. Syst. Softw. 85 (6), 1222–1238.
Takeuchi, H., Nonaka, I., 1986. The new product development game. Harvard Bus. Rev. 64 (1), 137–146.
Therrien, E., 2008, August. Overcoming the challenges of building a distributed agile organization. In: Agile 2008 Conference. IEEE, pp. 368–372.
Trist, E., 1981. The Evolution of Socio-technical Systems. Occasional paper 2 1981.
Turner, R., Ledwith, A., Kelly, J., 2010. Project management in small to medium-sized enterprises: matching processes to the nature of the firm. Int. J. Project Manage. 28 (8), 744–755.
VersionOne, 2015. State of Agile Survey—Version One. http://info.versionone.com/state-of-agile-development-survey-ninth.html.
Vidgen, R., Wang, X., 2009. Coevolving systems and the organization of agile software development. Inf. Syst. Res. 20 (3), 355–376.
Whitworth, E., Biddle, R., 2007, August. The social nature of agile teams. In: Agile Conference (AGILE), 2007. IEEE, pp. 26–36.

Rashina Hoda is the director of the SEPTA research group at the Electrical and Computer Engineering department of The University of Auckland, NZ. She received her Ph.D. in computer science from Victoria University of Wellington, NZ. Her areas of research interests include human and social aspects of Software Engineering, in particular Agile software development teams, and Human Computer Interaction. Her research has been published in the IEEE Transactions on Software Engineering, IEEE International Conference on Software Engineering, IEEE Software, Empirical Software Engineering, and Information and Software Technology. She served as the Research Chair of the Agile India 2012 conference and regularly serves on program and editorial committees of premier SE journals and conferences. She is the Chair of the IEEE NZN Computer Society and Women in Engineering group, a member of the IEEE, and a Certified Scrum Master.

Latha Karthigaa Murugesan graduated in Information Technology from Anna University, Chennai, India in 2008, and the Engineering Masters (M.E.) degree in Computer Science Engineering from the same university in 2011. She is a recipient of The University of Auckland Doctoral Scholarship for her doctorate degree which she is currently pursuing at the Electrical and Computer Engineering department. She is also a recipient of best outgoing student award in 2008 and Chairman's medal for her academic performance in 2011. Her research interests include software engineering, cryptography, HCI, and security. She has around 3 years of teaching experience. She is an active member of IEEE.