The Journal of Systems and Software 119 (2016) 87–108

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

Challenges and success factors for large-scale agile transformations: A systematic literature review

Kim Dikert[a], Maria Paasivaara[a,b], Casper Lassenius[a,b,*]

[a] School of Science, Department of Computer Science and Engineering, Aalto University, Finland
[b] Massachusetts Institute of Technology, Sloan School of Management, United States

ARTICLE INFO

Article history:
Received 6 October 2015
Revised 5 June 2016
Accepted 6 June 2016
Available online 7 June 2016

Keywords:
Agile software development
Organizational transformation
Large-scale agile
Adopting agile software development
Challenges
Success factors
Systematic literature review

ABSTRACT

Agile methods have become an appealing alternative for companies striving to improve their performance, but the methods were originally designed for small and individual teams. This creates unique challenges when introducing agile at scale, when development teams must synchronize their activities, and there might be a need to interface with other organizational units. In this paper we present a systematic literature review on how agile methods and lean software development has been adopted at scale, focusing on reported challenges and success factors in the transformation. We conducted a systematic literature review of industrial large-scale agile transformations. Our keyword search found 1875 papers. We included 52 publications describing 42 industrial cases presenting the process of taking large-scale agile development into use. Almost 90% of the included papers were experience reports, indicating a lack of sound academic research on the topic. We identified 35 reported challenges grouped into nine categories, and 29 success factors, grouped into eleven categories. The most salient success factor categories were management support, choosing and customizing the agile model, training and coaching, and mindset and alignment.

1. Introduction

Agile methods were originally designed for use in small, single-team projects (Boehm and Turner, 2005). However, their shown and potential benefits have made them attractive also outside this context, particularly both for larger projects and in larger companies. This despite the fact that they are more difficult to implement in larger projects (Dybå and Dingsøyr, 2009). Compared to small projects, which are ideal for agile development, larger ones are characterized by the need for additional coordination. A particular problem in applying agile to larger projects is how to handle inter-team coordination. Large-scale agile involves additional concerns in interfacing with other organizational units, such as human resources, marketing and sales, and product management. In addition, large scale may cause users and other stakeholders to become distant from the development teams. Despite these known problems related to large-scale agile, there is an industry trend towards adopting agile methodologies in-the-large (VersionOne, Inc, 2016; Paasivaara et al., 2013, 2014; Dingsøyr and Moe, 2014).

The State of Agile Survey that Version One has been conducting annually since 2007, has recently asked a few questions related to large scale as well, e.g. on scaling methods used and tips for success with scaling agile. According to the latest survey (VersionOne, Inc, 2016), 62% of the almost 4000 respondents had more than a hundred people in their software organization and 43% of all the respondents worked in development organizations where more than half of the teams were agile. Of course, the sample of this study is limited to a selected subset of companies and countries (of the almost 4000 respondents to the latest survey 65% were from North America and 26% from Europe). However, this indicates that there seems to exist a large number of companies that have taken or are taking agile into use in large-scale settings (VersionOne, Inc, 2016).

While the research literature contains several experience reports and some case studies on large-scale agile adoption, a systematic overview and synthesis of this growing body of research is still missing. Freudenberg and Sharp (2010) asked the industrial practitioners at the XP2010 conference to create a backlog of topics they think should be studied. The practitioners voted "Agile and large projects" as the top burning research question. Moreover, among the top ten items three focused on distributed agile development, which is relevant especially for larger organizations as

* Corresponding author.
E-mail addresses: kim-karol.dikert@aalto.fi (K. Dikert), maria.paasivaara@aalto.fi (M. Paasivaara), casperl@mit.edu, Casper.Lassenius@aalto.fi (C. Lassenius).

http://dx.doi.org/10.1016/j.jss.2016.06.013

they are often geographically distributed. In two recent workshops on large-scale agile development organized in XP2013 and XP2014 conferences, adoption of agile methods was one of the highlighted themes needing more research (Dingsøyr and Moe, 2013, 2014).

While research on agile software development is accumulating and maturing, and has provided a basis for conducting systematic literature reviews (Dybå and Dingsøyr, 2008; Jalali and Wohlin, 2012; Senapathi and Srinivasan, 2013; Kaisti et al., 2013), the area of large-scale agile development has not yet been studied through secondary studies. In this paper we start filling in this gap by presenting a systematic literature review of large-scale agile transformations.

2. Background

2.1. Agile software development

Agile software development is a set of iterative and incremental software engineering methods that are advocated based on an "agile philosophy" captured in the Agile Manifesto (Fowler and Highsmith, 2001). While mostly repackaging and re-branding previously well-known good software development practices, the agile movement can be considered as an alternative to so called traditional software development methods. Traditional methods focus on up-front planning and strict management of change, but agile methods were designed to accept and efficiently manage change (Highsmith and Cockburn, 2001; Cockburn and Highsmith, 2001).

Agile methods have been both criticized and advocated, and research has shown that accommodating change may be a factor in both success and failure (Boehm, 2002). It has been shown that agile methods have improved satisfaction of both customers and developers, but on the other hand there is evidence that agile methods may not be a good fit for large undertakings (Dybå and Dingsøyr, 2009). A proposed solution is that each organization seeks its own balance of agile and plan driven methods (Boehm, 2002).

Two of the most popular agile methods are Extreme Programming (XP) and Scrum (Hamed and Abushama, 2013). Scrum is a method focusing on the project management viewpoint of agile development (Schwaber and Beedle, 2002), prescribing timeboxing, continuous tracking of project progress, and customer centricity. The XP development method is a collection of practices for enabling efficient incremental development (Beck, 1999). In practice, many agile development implementations combine the two in some way (Fitzgerald et al., 2006).

2.2. Adopting large-scale agile

The difficulty of introducing agile methods increases with the organization size (Dybå and Dingsøyr, 2008). The difficulty is partly related to size bringing higher organizational inertia which slows down organizational change (Livermore, 2008). Agile development is not founded on the use of individual tools or practices, but rather on a holistic way of thinking. Adopting agile often requires change of the entire organizational culture (Misra et al., 2010).

One significant difference between small and large-scale adoptions is that larger organizations have more dependencies between projects and teams. This increases the need for formal documentation and thus reduces agility (Lindvall et al., 2004). In addition to inter-team coordination, development teams must interact with other organizational units, which are often non-agile in nature. For instance, human resources unit may demand individuals to have strictly specified roles in projects (Boehm and Turner, 2005), or a change control board may inhibit the use of continuous integration or refactoring (Lindvall et al., 2004). All units affected by the agile transformation need to be informed and consulted, and the agile process must be adjusted according to their needs (Lindvall et al., 2004; Cohn and Ford, 2003; Boehm and Turner, 2005).

Agile methods also affect management and business related functions. A key challenge is that management must move away from life-cycle models and towards iterative and feature centric models (Nerur et al., 2005), which requires a change of mindset. The focus must be shifted from long-term planning to shorter term project planning (Misra et al., 2010), as agile methods emphasize that planning is only meaningful for the near future (Cohn and Ford, 2003). However, the lack of planning can be a concern as business and customer relationships often build on long term roadmapping. Enabling operation with shorter term planning requires educating stakeholders and reviewing contracting practices (Boehm and Turner, 2005).

2.3. Definition of large-scale agile

A brief literature search (Dingsøyr et al., 2014) identified previous interpretations of what large-scale agile is. Size had been regarded in terms of size in persons or teams, project budget, code base size, and project duration. The examples of cases that were called "large-scale" included 40 people and 7 teams (Paasivaara et al., 2008), project cost of over 10 million GBP and a team size of over 50 people (Berger and Beynon-Davies, 2009), a code base size of over 5 million lines of code (Petersen and Wohlin, 2010), and a project time of 2 years with a project scope of 60–80 features (Bjarnason et al., 2011). Based on their findings Dingsøyr et al. (2014) ended up measuring large-scale by the number of collaborating and coordinating teams: they categorized as large-scale 2-9 collaborating teams and as very large-scale over ten collaborating teams.

We identified a number of additional studies discussing large-scale agile software development and their interpretations of large-scale. All of these referred to the number of people involved. In early work on agile, Fowler considers the Crystal methodology to be suitable for up to 50 people (Fowler, 2000). The same number has been reported as seen by practitioners and researchers as the size of the largest organization suitable for agile (Williams and Cockburn, 2003). Other studies have referred to agile projects including up to 50 people as small (Koehnemann and Coats, 2009), and considered a development project large if it had a staff between 50 and 100 people, including all project personnel (Elshamy and Elssamadisy, 2006). The largest numbers were 300 people across 3 sites (Moore and Spens, 2008). Participants of the XP2014 large-scale agile workshop gave very varying definitions for large-scale agile development (Dingsøyr and Moe, 2014), showing that what is seen as large-scale depends very much on the context and the person defining it.

Based on these findings, we defined large-scale to denote *software development organizations with 50 or more people or at least six teams*. All people do not need to be developers, but must belong to the same software development organization developing a common product or project, and thus have a need to collaborate. For instance, Scrum masters and software architects are counted when assessing the organizational size. As some studies present the number of teams rather than the number of people, we correspondingly defined large-scale to denote development efforts involving at least six teams. Having six teams with an average size of six to seven people, plus a number of supporting staff, can reasonably be considered to form an organization of 50 people. In this definition, we include both companies that as a whole focus on software development, as well as the parts of larger (non-software focused) organizations that develop software, e.g. in-house software development units of large non-software corporations.