Problematizing Agile in the Large: Alternative Assumptions for Large-Scale Agile Development

Completed Research Paper

Knut H. Rolland
SINTEF *and* Westerdals School of Arts, Communication and Technology
Oslo, Norway
rolknu@westerdals.no

Brian Fitzgerald
Lero—the Irish Software Research Centre, University of Limerick
Ireland
bf@lero.ie

Torgeir Dingsøyr
SINTEF *and* Norwegian University of Science and Technology
Trondheim, Norway
torgeir.dingsoyr@sintef.no

Klaas-Jan Stol
Lero—the Irish Software Research Centre, University of Limerick
Ireland
klaas-jan.stol@lero.ie

Abstract

In this paper we critically examine the underlying assumptions in existing studies of large-scale agile software development. We use Alvesson and Sandberg's problematization methodology and find that existing studies of large-scale agile share a number of underlying assumptions relevant to small rather than large-scale projects. Empirically, we draw on a case study of a large-scale agile project lasting nearly four years and involving more than 120 participants. Interestingly, the findings of the study contradict many of the assumptions in the literature review. For example, work across boundaries becomes at least as important as work within teams. We contribute by developing an alternative set of assumptions better suited to the characteristics of large-scale agile software development. Based on this, we re-conceptualize agile in the large, emphasizing both the complex knowledge boundaries within the project itself, as well as the interactive complexity and tight coupling with technologies and processes outside the project.

Keywords: Large-scale agile, problematization, assumptions, literature review, case study

Introduction

Agile methods continue to increase in popularity to become the dominant mode of software development today (Ambler 2014). The initial fundamental assumptions underpinning agile methods suggested that they were best suited to (a) small projects with (b) co-located teams and (c) non-critical projects (Abrahamsson et al. 2009; Williams and Cockburn 2003). However, several authors have started to challenge these assumed constraints. For example, several studies have investigated the use of agile methods in distributed environments (Boland and Fitzgerald 2004; Kircher et al. 2001; Moe et al. 2014;

Stotts et al. 2003), and researchers have also begun addressing the use of agile in safety-critical and regulated domains (Cawley et al. 2010; Fitzgerald et al. 2013; Stålhane et al. 2012). However, the use of agile methods in large development projects remains a significant challenge (Booch 2015) with little empirical evidence of successful cases (Cao et al. 2004; Kähkönen 2004).

Further compounding the issue, there is by no means universal agreement as to the actual definition of a large project. Proposed measures include the number of development teams (Dingsøyr et al. 2014), the number of developers (Elshamy and Elssamadisy 2006; Hannay and Benestad 2010), the number of lines of code (Petersen and Wohlin 2009), and the duration of the project (Bjarnason et al. 2011). In order to better understand the application of "Agile in the Large," we challenge the assumptions that underpin large-scale projects as they pertain to the adoption of agile methods. This is of particular importance as agile methods emerged largely from practice without a great deal of consideration for sound conceptual foundation for the topic (Ågerfalk et al. 2009; Conboy 2009). Our objective is to problematize some of the widespread assumptions underlying current research on large-scale agile development and to develop a novel conceptualization that better captures the essence of large-scale agile projects.

We draw on Alvesson and Sandberg (2011; 2013) who propose the problematization of both conventional assumptions and the consensus of prevailing theoretical perspectives. They point out that the predominant mode of framing research currently is to identify research questions through spotting, or even "constructing" gaps in current theories—what they call the gap-spotting approach. Alvesson and Sandberg (2011) offer an alternative research approach to 'gap-spotting,' which goes further in that it seeks to challenge the assumptions that underpin these theories—which we term the assumption-challenging approach. Interestingly, the agile movement itself emerged from practitioners challenging the assumptions of traditional, so-called plan-driven software development methods. The Agile Manifesto offers four value propositions that highlight some of the key issues that underpin the differences between traditional software development paradigms that were dominant at the time the Agile Manifesto was written. Thus, we believe a problematization approach to study "agile in the large" is all the more fitting.

In order to problematize agile in the large, we reviewed the literature on large-scale agile and identified the basic underpinning assumptions using Alvesson and Sandberg's typology. We then considered how well these assumptions applied in practice through an empirical case study of a large and complex information system development project in a governmental organization in Norway. The literature review revealed the assumptions underpinning large-scale agile as an unproblematic and linear continuation of agile principles in general and the corresponding composition of the agile practices initially intended for small-scale projects. However, the findings from the case study revealed a considerable amount of additional work for coordinating and resolving unexpected problems related to a multitude of complex interdependencies across development teams, roles, software components, legacy systems, and organizational boundaries. These issues we argue, are not marginal exceptions—but rather an *essential* part of scaling agile methods to large-scale projects.

This study seeks to make two contributions. First, we challenge the current underlying assumptions underpinning much of the literature on large-scale agile development which tends to simply extrapolate agile practices and coordination mechanisms in small projects (e.g. scaling scrum to scrum-of-scrums, scaling user stories to epics). We believe there is a need to emphasize the *boundary work* and *boundary infrastructures* that are required for working across contexts resolving and coordinating complex socio-technical interdependencies (Bowker and Star 1999; Carlile 2002; Carlile 2004; Levina and Vaast 2005). Re-conceptualizing agile in the large in this way, the 'largeness' of agile projects can be perceived in terms of the complexity of the various 'knowledge boundaries' across the actors and technologies involved. Following Carlile (2004), a knowledge boundary refers to the complexity of transferring, translating and transforming knowledge across different actors. Thus, we shift our focus to how complex socio-technical interdependencies require boundary work on different syntactic, semantic and pragmatic levels of communicating and coordinating. In contrast, for example, some of the underlying assumptions in the extant literature on large-scale agile tend to retain a small-scale focus on practices and artifacts *within* teams. Moreover, we argue that large-scale agile projects do not simply imply "more of the same" communication and coordination challenges found in small-scale agile projects (e.g. Robinson and Sharp (2010)), but surface an altogether different kind of problem associated with potential 'complex interactions' and 'tight coupling' between involved teams, software subsystems and systems, an installed base of existing systems, and actors more peripheral to the project (Perrow 1999). To summarize, we re-

conceptualize large-scale agile projects as a function of both the complex knowledge boundaries within the project (e.g. across teams), as well as the interactive complexity and tight coupling with technologies and processes outside the project itself (e.g. integration with existing external information systems). These issues are thus likely to be intensified when the number of teams in a project increases (Dingsøyr et al. 2012).

Second, this paper contributes methodologically by demonstrating Alvesson and Sandberg's approach to problematization, challenging long-standing assumptions in agile software development. We believe this serves as a useful exemplar to demonstrate how IS researchers can take an *assumption-challenging approach* by drawing from and analyzing extant theoretical frameworks, and then seeking to identify alternative assumptions empirically in real contexts.

The paper proceeds as follows. First, we outline Alvesson and Sandberg's framework for problematization and its relevance for information systems and agile software development. We then describe our two-part research approach, comprising a review of the literature on large-scale agile and an in-depth interpretive case study. We then present the findings from the two parts of the research. We conclude with a discussion of the key findings and offer an alternative set of assumptions for agile in the large.

Background and Related Work

From 'Gap Spotting' to Problematizing

In general, research in Information Systems (IS) and in many other scientific fields is expected to draw from and build upon established bodies of research. In so doing, researchers often construct their research questions through finding 'gaps' in an existing body of literature. For example, in the context of large-scale agile software development, Paasivaara and Lassenius (2012: p.236) identified a 'gap' in the extant scientific literature: *"[We] found only three research articles briefly reporting experiences from projects over ten Scrum teams."* The authors position and legitimate their research on 'Scrum-of-Scrums' based on finding and demonstrating a 'gap' in current literature. Webster and Watson (2002: p. xiii) note that a literature review *"uncovers areas where research is needed."* While gap-spotting is clearly a useful approach that has led to significant contributions in a host of research field,a fundamental problem with the gap-spotting approach is that it can result in under-problematization which *reinforces*, rather than *challenges* existing theories. Researchers tend to build excessively on existing literature rather than challenging consensus. Alvesson and Sandberg identify this as a problem in several disciplines, including IS (e.g., Barrett & Walsham 2004). For example, researchers may 'downgrade' assumption-challenging research questions to frame them as merely gap research questions as this is the conventional approach and is consequently less likely to be unfamiliar and therefore potentially problematic for reviewers in the publication process. Davis (1986) argues that the most significant papers often challenge the assumptions of existing theories, rather than merely spotting gaps and ultimately reinforcing existing theories. Alvesson and Sandberg identify two scenarios whereby consensus challenging may occur. Firstly, researchers may draw on "master thinkers" to challenge the existing theory base, citing Baudrillard and Foucault as examples. Secondly, consensus-challenging research can emerge through what they label as 'stock' questions. For example, seeking a feminist narrative for a discourse to which it had not been previously applied. However, Alvesson and Sandberg call for more direct consensus- and assumption-challenging research which can be achieved by problematizing existing assumptions. can be achieved by problematizing existing assumptions.

The literature on agile software development offers several examples of consensus building around important research challenges (Gregory et al. 2016), complemented by suggestions from editors of special issues such as Ågerfalk et al. (2009) on agile/distributed systems development, and Abrahamsson et al. (2009) on agile systems development. Several authors have proposed research agendas based on a systematic review of empirical studies in the field (Dybå and Dingsøyr 2008), or based on workshops with experts in this area (Dingsøyr et al. 2014). Others have focused on the needs of industry participants to define a research agenda (Freudenberg and Sharp 2010; Gregory et al. 2016). Most of these initiatives can be characterized as "gap-identifying," with gaps being uncovered in the extant literature or between our current knowledge and industry needs. To the best of our knowledge, nobody has challenged the very assumptions that underpin "agile in the large."

A Framework for Problematizing Research

Alvesson and Sandberg (2011) propose a typology of five different kinds of assumptions in a scientific field, including: 1) in-house assumptions, 2) root metaphor assumptions, 3) paradigmatic assumptions, 4) ideological assumptions, and 5) field assumptions. We briefly discuss each of these assumption categories.

The first category is labeled **in-house assumptions** and refers *"to a set of ideas held by a theoretical school about a specific matter"* (p. 254). In-house assumptions are shared by subgroups of researchers within an area of research and affects how they conceptualize a particular subject matter. In the information systems field, Iivari et al. (1998) identified five different approaches to IS development, each with different goals, guiding principles and fundamental concepts. For example, Iivari et al. discuss how the *Interactionist* approach conceptualizes information systems as 'institutions' whereas the *Trade Unionist* approach considers computers as 'tools.'

Root metaphor assumptions signify a deeper aspect of the subject matter by using conceptual images to understand the topic of study. A well-selected metaphor can be a powerful analytical tool to understand the world—however, a poorly chosen metaphor may misrepresent it. Alvesson and Sandberg (2011) draw on Morgan's (1997) work on using different metaphors to analyze organizations. The use of metaphors has also been discussed within the IS literature (Hirschheim and Newman 1991; Walsham 1993). For example, Hirschheim and Newman (1991) discuss IS development as a "battle," characterizing the interaction between users and developers. A very common metaphor for traditional IS development approaches is the "waterfall."

Paradigmatic assumptions are concerned with the underlying epistemological and ontological views of the research in the dominant literature. This has long been an issue within IS research, and many authors have argued that a large amount of the literature is based on a largely (post-)positivistic epistemology and ontology, rather than, for example, critical and interpretive paradigms of research (Orlikowski and Baroudi 1991; Walsham 1995). The literature on agile software development is typically not explicit on the epistemological assumptions of the studies. Paradigmatic assumptions are not limited only to the *research* paradigm, but the term 'paradigm' may also refer to the many different theoretical lenses that are used within IS—Actor/Network Theory, Agency Theory and Social Network Theory, to name just a few. Furthermore, alternative conceptualizations of the topic at hand also fall within this category, such as the four paradigms of IS development discussed by Hirschheim and Klein (1989).

Ideological assumptions include the *"various political-, moral-, and gender-related assumptions"* underlying the research (Alvesson and Sandberg: p. 255). One example is a study by Razavian and Lago (2016), who investigated the expertise in IS architecting teams brought in by female architects. The authors motivated their study through gap spotting—lacking any previous studies on this topic, researchers had evidently not considered that female architects bring specific types of expertise.

The last category of the typology, that of **field assumptions**, entails a broader set of assumptions that can be identified within several different branches or genres of literature and research. Field assumptions are those that are shared across different schools of thought, or across different theoretical perspectives. The scope of the 'field' may vary from a particular subfield (e.g. IS development within the wider IS research field) to a complete research discipline, and this is defined by the researcher who adopts the problematization approach. Several researchers have discussed the nature of the information systems field (Bacon and Fitzgerald 2001; Hirschheim and Klein 2003), and while such analyses often result in a classification of research streams or topics, the information systems field as a whole is based on a number of assumptions that are widely accepted within the community. While the field itself is changing, driven by a constant stream of technical advances, it is fair to state that a field assumption shared by all IS researchers is that information systems help organizations and individuals to achieve their goals—rejecting this assumption would arguably disqualify the IS field itself.

In addition to the typology of assumptions summarized above, Alvesson and Sandberg (2011) propose the following steps which should be pragmatically iterated when problematizing the dominant literature: 1) Identifying a domain of literature for assumption-challenging investigations; 2) Identifying and articulating assumptions underlying the chosen domain of literature; 3) Evaluating articulated assumptions; 4) Developing an alternative assumption ground; 5) Considering assumptions in relation to target audience; and 6) Evaluating the alternative assumption ground.

The first step is to identify a particular domain of literature. The researcher, having achieved an overview of the domain, could concentrate on a few exemplars for *"deep readings and rereadings"* (p. 256). An alternative strategy is to find a path-defining study or an authoritative summary. The next steps are to further identify underlying assumptions using the topology, and then to evaluate these assumptions. Alvesson and Sandberg note that not all assumptions in research texts are made explicit, but that many assumptions remain implicit. Furthermore, Alvesson and sandberg highlight the role of the researcher's creativity. While evaluating assumptions, Alvesson and Sandberg state that a useful test could be whether an assumption turns out to be perceived as *obvious, absurd, or interesting*. Researchers should strive for interesting assumptions or ones that are not directly absurd or obvious. Following this, an important step is to develop new alternative assumptions that can form the foundation for novel research questions. Here, Alvesson and Sandberg recommend that the researcher consult available *"reflexive and critical literature, representatives of competing schools, and various forms of heuristic tools"* (p. 258). They caution that simply substituting functionalist assumptions with, for example, an interpretive stance, is not sufficient. Rather, the researcher should play with different theories and concepts in order to facilitate a creative process for surfacing new alternative assumptions. For example, in relation to large-scale agile, this could be theories of infrastructures (Star 2010) and complex socio-technical interdependencies in large technical systems (Perrow 1999). In the final step, assumptions should be considered in relation to the target audience. Our target here, for example, comprises the broad information systems and software engineering audiences.

Research Approach

Our research approach comprised two parts. The first part was to identify the dominant literature on large-scale agile development through a comprehensive literature search. We analyzed the resulting corpus to identify assumptions according to the Alvesson and Sandberg typology discussed earlier. The second part involved an empirical case study of large-scale agile development where we sought to evaluate assumptions in the dominant literature and develop a set of alternative assumptions. These two phases did not occur in a strictly sequential fashion. Parts of the case analysis were also conducted after the literature review, reflecting the findings in the literature review. In this regard, our study also involved an element of *"problematization of empirical material*" (Alvesson and Sandberg 2013).

Identifying the dominant literature on large-scale agile

We conducted a search of relevant studies in several digital libraries (Appendix A provides further details). This resulted in an initial set of 88 publications. Studies that were clearly unrelated to large-scale agile development were removed following examination of titles and abstracts. The remaining 43 articles were divided amongst the four authors and read in detail. A further six articles were subsequently found to be out of scope, leaving 37 for analysis to identify underlying assumptions. While reading the articles, we recorded the assumptions, their source and type according to Alvesson and Sandberg's typology in a spreadsheet. We discussed their applicability to "Agile Large" through several workshops.

Case study setting, selection and method

We conducted a longitudinal case study of a large-scale agile software development project from September 2014 to March 2016. The project, referred to as the *Replacement project*, involved over 120 participants during a period spanning almost four years from 2011 to late 2014. The project was organized as four Scrum development teams and one additional team more loosely affiliated with the project, and which was responsible for developing a business intelligence (BI) solution. One of the Scrum teams had specific responsibility for integration and back-end development, whereas the other three teams were feature teams working on a domain-specific part of the solution. This way of organizing was perceived by both the customer and the consultant company as optimal for minimizing the need to coordinate across teams. The customer was a governmental organization (here referred to as "GOV") with over 7,000 employees distributed across more than 70 locations in Norway.

We selected this case largely because it was one of Norway's largest ongoing IT-projects with extensive use of agile methods. The project was also interesting because GOV had been involved in two failed attempts at conducting the project using both agile and traditional development approaches during the last decade.

Furthermore, the project involved complex integration among a wide variety of internal and external information systems, involving various stakeholders with divergent interests. Moreover, before starting the Replacement project, the supplier, an international consulting company (hereafter referred to as ConsultCorp), had been part of a prestigious large-scale agile software development project that was celebrated nationally as a great success. This project is often used as a template for other large-scale agile projects in Norway. Accordingly, we consider this case as an atypical or extreme case exhibiting a number of aspects relevant for problematizing existing studies as such cases are often well positioned for identifying "black swans" (Flyvbjerg 2006). Henceforth, we also see this as a particularly relevant case study for theorizing (Eisenhardt 1989; Langley 1999). Our study draws on the established tradition with theoretically informed interpretive case studies in IS (Walsham 1995; Walsham 2006) and hence aims at following relevant guidelines for such research (Klein and Myers 1999; Sarker et al. 2013).

Data Collection in the Case Study

As recommended when conducting case studies, our study draws on multiple data sources (Yin 2013). Our main source of empirical data is 20 in-depth semi-structured interviews with participants from both the customer and ConsultCorp (Table 1). Participants were carefully selected based on their knowledge of the project. We interviewed informants across all roles in the project, including project managers, designers, architects, developers, testers, and Scrum Masters. In terms of customers, we interviewed the project manager, test manager and project architects. The interviews lasted 60 to 90 minutes each.

The semi-structured nature of the interviews encouraged participants to speak freely and to share their insights and interpretations, and it also allowed the participants to steer the topics of discussion as they saw fit. This helped build rapport and to capture their specific concerns as they pertained to our topic of research. However, a checklist of topics including team coordination, collaboration with customer and stakeholders, as well as the role of architecture and legacy systems was used to guide the interviews. As the interviews were conducted over a period of 18 months, the checklist was refined over time in order to cover emerging themes. In our interviewing we have drawn on recommendations from Myers and Newman's (2007) *"dramaturgical model."* As shown in Table 1, the study covered all roles in the project.

A second source of data was a series of more than ten meetings with ConsultCorp. Here we discussed topics relevant to our study, and in particular used the information from meetings to inform further in-depth interviews. These meetings also provided important historical and contextual insights on the Replacement project, the initial intentions behind its organization, as well as commercial and competence aspects. A third important source of data was two workshops, which focused on specific issues the participants identified as especially important for the outcomes of the project at different phases in the process. Finally, a fourth data source was the project documentation to which we had unrestricted access, including the issue tracker and an internal wiki containing material comprising all user stories, contract documents and other documentation used by the project management. This triangulation of data sources, as well as the triangulation among researchers, were two tactics that helped us to better establish the credibility of this study.

Data Analysis

Qualitative analysis was performed following Miles and Huberman (1994) using a combination of seed categories reflecting key themes of the research as "analytical bins," such as inter-team coordination, different roles, agile practices, and issues of architecture and integration, as well as more theoretical categories influenced by concepts and frameworks as "knowledge boundaries" (Carlile 2004), "boundary spanning-in-practice" (Levina and Vaast 2005), and "boundary infrastructures" (Bowker and Star 1999; Star 2010). We employed several iterations of coding, as well as a combination of 'narrative' and 'temporal bracketing' strategies (Langley 1999).

Additionally, at several stages in the process of analysis, issues, ambiguities and reflections were presented and discussed with participants of the study. This included formal presentations at ConsultCorp as well as more informal meetings and email discussions with participants. This helped to clarify our analysis, and also confront our own biases towards the data material. In this way, the analysis followed the principle of 'interaction between researchers and subjects' (Klein and Myers 1999).

Table 1. Overview of Informants		
Role	**No. Informants**	**Experience**
Developers/ Scrum masters	7	3 of the developers had experience in large-scale projects. Two became scrum masters/team leaders by the end of the project.
Software architects	6	All software architects were very experienced and all had participated in large-scale projects before.
Interaction designers	1	15 years experience including on large-scale agile projects.
Testers	2	One of the testers had 15+ years experience on software testing and experience from a similar large-scale agile project. The other participated for almost 3 years in the project
Managers	4	Only 1 of the managers interviewed had participated during the whole project. All managers had extensive experience with large and complex software projects and 3 of them had experience with large-scale agile.

Assumptions of Large-Scale Agile Software Development

As briefly mentioned, our analysis of the literature on large-scale agile projects in the first part of our research approach resulted in an extensive list of assumptions which were established by all four co-authors categorizing and discussing assumptions in an iterative manner, both individually and through several face-to-face and online workshops. The final set of assumptions were categorized according to the Alvesson and Sandberg typology that is presented in Table 1.

We identified several **in-house assumptions,** which arise in individual agile methods such as Scrum and XP. In-house assumptions often lack universal agreement. For example, the assumption that agile is less suitable to large development projects is generally accepted (Booch 2015; Williams and Cockburn 2003). However, what is far less clear is the assumption as to *what exactly defines* a large development project. Some authors suggest that 2-9 teams represent a large development project (Dingsøyr et al. 2014). Others suggest 50-100 developers (Elshamy and Elssamadisy 2006) or more than 20 developers (Hannay and Benestad 2010); others still suggest projects of more than 100 KLOC (Petersen and Wohlin 2009). Other examples of in-house assumptions include the collective/shared code ownership. However, for a large project, it is not feasible to rotate developers to expose them to the entire system to implement shared code ownership (Elshamy and Elssamadisy 2006). Another assumption is that the product owner role can be scaled up by simply establishing a team of product owners to manage customer requirements in large projects (Bass 2014). Agile methods place a premium on knowledge transfer among team members. Practices such as pair programming, code reviews and daily stand-ups are used to achieve knowledge transfer (Elshamy and Elssamadisy 2006). However, such mechanisms do not necessarily work in large projects (Elshamy and Elssamadisy 2007). One mechanism that has been proposed to overcome this issue is the use of Communities of Practice (CoP) which are assumed to improve effective knowledge transfer in large projects (Paasivaara and Lassenius 2014).

Root metaphor assumptions. The 'waterfall' model of software development is itself a metaphor, and agile based on short iterations (sprints) is often positioned as the antithesis to the waterfall model (Nerur et al. 2005). Metaphors abound in relation to agile software development. While the identification of an appropriate metaphor is a key practice in Extreme Programming (XP), there are several other examples of the use of metaphors in agile software development. The Scrum concept is itself a metaphor based on the rugby tactic in which several individuals perform different roles to achieve the overall objective. Within Scrum, the concept of a 'sprint' is also a metaphor which implies quick bursts of work that result in the release of working software. The concept of 'technical debt' is also a metaphor which implies a future

technical commitment to solve issues caused by certain choices made earlier in the development lifecycle. Research which draws analogies from complex adaptive systems (CAS) to study agile (e.g. Vidgen and Wang (2009)) also tends to apply an organic metaphor to agile (Cao et al. 2004).

The literature on large-scale agile software development is typically not explicit on the paradigmatic assumptions behind the studies. However, reading the dominant literature on agile software development there is a clear tendency to emphasize the homogeneity of organizations rather than viewing them as composed of different actors who assign different meanings to different software features and potentially have conflicting needs and interests. There is an over-emphasis on success stories and case studies are typically analyzed from a positivistic viewpoint (Paasivaara et al. 2012). There is a focus on normative aspects of work and less concern with agency and how agile methods are uniquely instantiated in practice

Table 2. Assumptions Underpinning Large-Scale Agile Development in the Literature

In-House Assumptions:

- Collective/shared code ownership (Elshamy and Elssamadisy 2006)
- Product owners can be scaled to manage customer requirements (Bass 2014)
- Knowledge transfer through pair programming, code review, daily stand-ups (Elshamy and Elssamadisy 2006) and establishment of Communities of Practice (Paasivaara and Lassenius 2014)

Root Metaphor Assumptions:

- Waterfall vs. iterative development (Nerur et al. 2005)
- Complex adaptive systems and organisms (Cao et al. 2004; Vidgen and Wang 2009)
- Sprints as bursts of work resulting in working software
- Technical Debt suggesting taking shortcuts results in extra effort later on in a project.

Paradigmatic Assumptions:

- Over-emphasis on success stories analyzed from a positivist viewpoint (e.g. Paasivaara et al 2012)
- Organizations seen as homogeneous entities
- Normative view with less concern for agency (e.g. Batra et al. (2010))
- Replacement (e.g. working software instead of comprehensive documentation; tests instead of specifications; user stories instead of requirements (Bjarnason et al. 2016; Meyer 2014)
- Composition - scaling up seen as unproblematic: Scrum of Scrums; user stories to epics; sprint backlogs to product backlogs (e.g. Bass 2014)

Ideological Assumptions:

- Political language of Agile *Manifesto* and labeling of zealots as *agilistas* (Kelley 2007); however, not all developers necessarily share this enthusiasm and motivation (which is more likely to occur on large projects with many developers).
- Developer-oriented rather than manager-oriented e.g. maximum 40-hour week to avoid developer burn-out (Beck 2000)
- Self-organizing teams driven by motivated individuals (Costa et al. 2014; Hoda et al. 2013)

Field Assumptions:

- Generally held assumption that agile is best suited to small projects, with co-located teams, and non-critical development contexts (Williams and Cockburn 2003).
- Close involvement of customer necessary and desirable, but this may not be possible in large projects.
- Agile principles seen as axiomatic with tailoring of agile methods accomplished by adding additional practices (Conboy and Fitzgerald 2010; Coyle and Conboy 2009; Fitzgerald et al. 2013; Sundararajan et al. 2014).
- No cost of context-switching given developer focus on finely granulated tasks (Hannay and Benestad 2010)
- Greenfield development context (Boehm 2006)

(Batra et al. 2010; Heikkila et al. 2013). There is also a tendency towards reductionism where a micro phenomenon that 'works' in one confined location or context is extrapolated to a different level of analysis. For example, reading the textbook version of agile (e.g. Rubin (2013)), inter-coordination between teams is expected to be operationalized through the practice of Scrum-of-Scrums. In the literature these issues remain implicit, however; the ethnographic study by Sharp and Robinson (2004) describes, but does not problematize whether localized XP practices scale or not; nor do they discuss issues of conflicting users' interests and interpretations. An exception is a study by Paasivaara and Lassenius (2014) which suggests that the practice of Scrum-of-Scrums is usually inefficient and insufficient for coordinating among many different teams. However, this observation is not theoretically explained in further detail.

A related paradigmatic assumption in agile is that of replacement. This is clearly evident in the four underpinning values (e.g. working software as a replacement for comprehensive documentation). Often it seems that in agile approaches, practice X—if unpalatable or difficult to achieve—can be replaced by practice Y (Meyer 2014); an example might be that of user stories as a replacement for formal requirements, or tests instead of specifications (Bjarnason et al. 2016).

The principle of composition is also a paradigmatic assumption and arises in the simplistic view that a Scrum-of-Scrums is an appropriate scaling up mechanism. This view is also present in the view of user stories scaling to become epics, or sprint backlogs scaling to a product backlog.

In terms of **ideological assumptions**, the Agile Manifesto is clearly an ideological call-to-arms for *agilistas*. The moral principle of the 40-hour week in XP and the broad focus on delivering customer value constantly are ideological value statements. Emancipatory ideals can be observed in the strong focus on developers rather than managers as espoused in the Agile Manifesto. This represents a shift from management and control-oriented (plan-driven) approaches to IS development, to a more egalitarian view underscoring the competence and creativity of developers, although the naive idealistic assumption of developers in consistent harmony has been questioned (Conboy et al. 2011). In their study of large-scale agile development, Paasivaara and Lassenius (2014) explain the success of software development in Ericsson through the establishment of various *Communities-of-Practice* (CoP). According to the authors these CoPs are, among other things, voluntary, open, and also have a certain decision-making authority. Reinforcing this is the view that projects be built by self-organizing teams comprising motivated individuals (Costa et al. 2014) with trust being seen as especially important. This tendency to view development efforts as activities within self-organizing teams and that agility implies optimizing self-organization (Vidgen and Wang 2009) could be regarded as an ideological assumption. However, the very notion that teams "self-organize" is rarely, if ever, questioned in the literature.

The last category of the typology entails a broader set of **field assumptions** that can be traced within several different branches or genres of literature and research. In agile software development this could imply assumptions that are shared across communities of practitioners, IS researchers and software engineering researchers. Across these different literatures, there is an assumption that agile principles are sacrosanct, and that tailoring of methods by adding practices to resolve problems that arise in specific contexts is an appropriate solution (Conboy and Fitzgerald 2010; Coyle and Conboy 2009; Fitzgerald et al. 2013; Sundararajan et al. 2014). But generally there is no inherent questioning of the actual assumptions underpinning the agile principles. A further example of such thinking is the assumption that agile methods are best suited to small projects (Costa et al. 2014; Sundararajan et al. 2014) with co-located teams (Williams and Cockburn 2003). In these contexts, small teams are easier to manage (Elshamy and Elssamadisy 2006), and are expected to possess different competences and work in close collaboration (Elshamy and Elssamadisy 2007). At the field level also, is the assumption that change is inevitable in a software development context (Batra et al. 2010; Hannay and Benestad 2010; Heikkila et al. 2013) and that requirements can only be finalized quite late in the development cycle (Elshamy and Elssamadisy 2006). Close involvement of users or user proxies/Product Owner and Scrum Master is assumed (Costa et al. 2014). Also, the typical agile focus is on *intra-team* issues rather than *inter-team* issues (Cao et al. 2004). There is an assumption that there is no context switching as all team members are fully focused on the low granularity of the single job in hand (Hannay and Benestad 2010). This is reinforced by the principle of collective code ownership (itself, an in-house assumption for XP) which assumes that everyone involved can oversee everything. However, in large projects, not all will possess the big picture of what all teams are working on (Hannay and Benestad 2010).

Furthermore, a number of specific assumptions arise in relation to large projects. Firstly, communication is assumed to be difficult in large teams, as is knowledge-sharing (Elshamy and Elssamadisy 2006). Stand-up meetings cannot be relied on as the means of communication (Elshamy and Elssamadisy 2007). Also, large-scale development typically involves more dependencies (Gunyho and Gutiérrez Plaza 2011; Hannay and Benestad 2010; Moe et al. 2014).

Finally, an underlying field assumption is that there is little consideration as to the specific kind of software artifact developed and in particular a tendency to assume 'greenfield' development of new software systems, rather than 'brownfield' development which involves building and integrating legacy systems. Arguably, the larger the project, the more likely is it that it is a brownfield with a heterogeneous installed base of existing systems and practices (Rolland et al. 2015).

Case Study of a Large-Scale Agile Project

The second part of our research involved an in-depth case study. In this section we present the development context followed by a narrative of the project we studied.

Development Context

In the early 2000s, GOV started an ambitious IS development project in order to modernize an existing legacy system. Modernization was urgent but complex as the core legacy system had been developed for mainframes and was difficult to extend. Over the years, this had been complemented with a large number of add-on systems that were more or less integrated with the mainframe system. The legacy system is a core system for GOV, and is of strategic importance for the functioning of the transport infrastructure in Norway. It has approximately 1,600 internal users and several thousand external users, including both private companies and other governmental agencies.

After a lengthy process, a contract was signed with a major consulting company, ConsultCorp (a pseudonym) in 2010. The contract specified a delivery model based on Scrum with additional roles and four different deliverables (D0-D3) that were planned to be executed partly in parallel. The incorporation of agile methods was chosen by GOV in order to "*maximize flexibility–and to avoid specifying all details up front,*" as stated by a customer representative. However, GOV had little to no experience with agile methods, as they were accustomed to a waterfall approach. Moreover, the IT department in GOV rarely performed in-house development on their own, but typically outsourced the development work to major consulting companies.

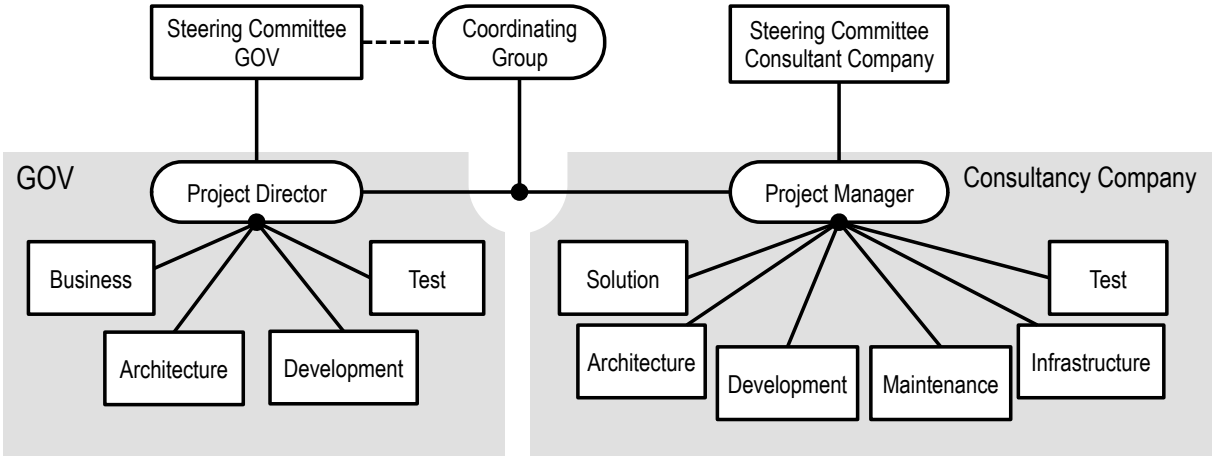


Figure 1. Organization structure of the Replacement Project

The main goals of this project (referred to as the *Replacement project*) was to establish a common IT architecture for a large portfolio of integrated systems based on off-the-shelf products from Oracle as well as replacing the old mainframe-based systems with a new system written in Java with a modern user-interface.

The typical Scrum team in the Replacement project consisted of around 10-13 participants of whom 7 or 8 were developers, potentially a User-Experience (UX) designer, one architect, one or two functional architects (customer representatives), and one team leader/Scrum master. During the first three phases there were five teams with one so-called integration team, one BI (Business Intelligence) team, and three feature teams. In the last phase of the project, only two feature teams were employed. The project used modern tools for Continuous Integration (Jenkins), an issue tracker for managing user stories, and a source code repository (GitHub). The project exhibited a matrix organization, implying that people could be member of a specific team, and also a member of an organizational unit for Solution Description, Development, Maintenance, Test, Architecture or Infrastructure. In addition, a similar structure of organizing was replicated on the customer-side of the project (see Figure 1).

The project followed a Scrum-based model that had been used in another large-scale project where ConsultCorp was involved. This previous project was perceived as highly successful, and is generally regarded as 'best practice' for doing large-scale agile in the Norwegian IT industry. The Scrum-based delivery model is characterized by splitting up a large project into different deliverables as shown in Figure 2. For each deliverable, a semi-agile process is followed by first defining user stories on a low-level, then architectural design, overall user-experience (UX) design, and refinement of user stories – but with minimal effort in order to refrain from too much up-front planning.

Case Narrative

Complex Interdependencies in IT Architecture

During the first deliverable (D0), the project soon ran into problems related to the pre-defined architectural principles in GOV. In meeting the goal of developing a common platform and a service-oriented software bus for all applications, GOV architects had selected off-the-shelf software systems, *Documentum* and *Oracle BPEL Process Manager*. These generic systems and the publication of all interfaces on a software bus turned out not to be feasible in practice as they had dramatic impacts on performance and design of user interfaces. Accordingly, the project had to 'work around' these constraints, and relinquished the architectural principle that all interfaces be published on the service bus.

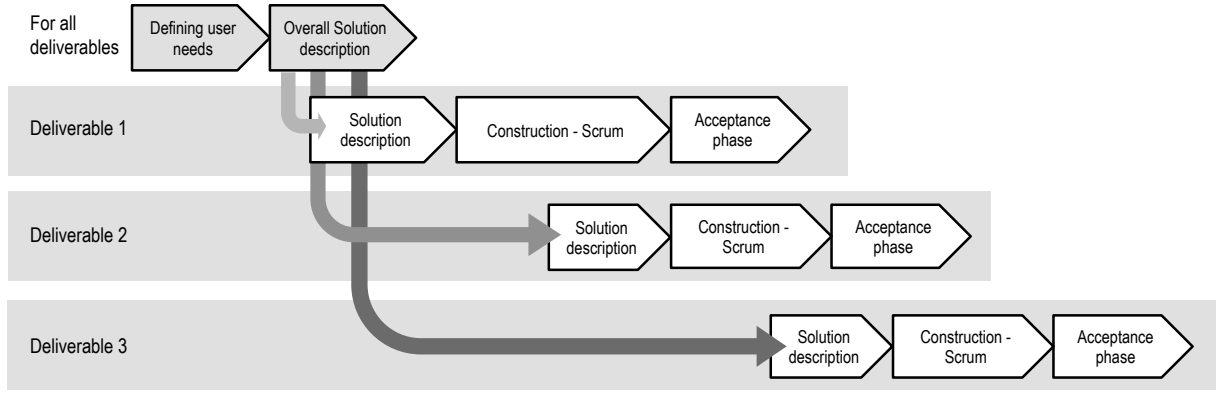


Figure 2: Scrum-based delivery model of the project.

Working in an iterative manner greatly helped in this process as the teams slowly found ways of improving the Oracle BPEL Process Manager solution. This resulted in it being replaced with similar features developed on a Java platform:

> *"In the end we had a system on the outside implemented from scratch on the Java platform. And, then the Process Manager suddenly was no longer necessary. It didn't cost us much to replace it [with a Java-based solution]. So, during the last deliverable [D2] we got rid of the Process Manager. You would have expected that over three deliverables it had grown into the solution, but that was not the case."* — Software architect, ConsultCorp

In this respect, agile practices helped address this problem since they focused on developing a running software system early in the development process. Had this not been done, these performance issues would have been discovered much later. Furthermore, the Service Bus problem caused a re-design during the next deliverable (D1a). From the customer's point of view, the first implementation that put all module interfaces on the Service Bus was partly a misinterpretation of the requirements by ConsultCorp. ConsultCorp denied this, however, arguing that this was explicitly stated in the original requirements document in the contract. This situation – with hugely different interpretations, underscore that rapid scaling of an agile project can make significantly contributed to difficulties in establishing a common understanding of architectural challenges. In particular, this becomes challenging because of unfamiliar platforms and technologies and requires specific competencies for configuration as well as integration with other parts of the software system. In other words, it implies complex interdependence that need to be accounted for in the development process.

Moreover, the large scope of the new software system integrating with numerous existing systems turned out to be more challenging than anticipated. For example, during work on Deliverable 3 alone, the project involved a staggering 40 different integrations, which required significant effort. Some of these integrations involved complex interdependencies with both internal and external information systems. One informant described the complex interdepencies:

> *"When the other system being developed was down, it almost stopped the entire project. There was a tight coupling between this other system and our project, which we had not accounted for initially. We realized that this system needed to follow the same production schedule as our project – although they were not at all part of the official project."* —Software Architect, GOV

Continuous 'Translating' of Requirements

A major challenge in the Replacement project was also to scale product owners so that they manage to facilitate the necessary 'translation' of user stories, that are typically numerous, involve complex interdependencies, and relatively high-level to development teams. A GOV architect described the situation:

> *"It becomes hard to follow the supplier [ConsultCorp] and answer their requests and make [timely] decisions. They want to work in an iterative and agile manner, and expect to have two or three domain experts in each team. We were never able to deliver those resources – we were not organized for that."* —Software architect, GOV

The 'translation' role emerged as a particularly important one when dealing with a large number of off-the-shelf software components, since this required specialized knowledge on these generic products and also domain knowledge. As a consequence, the project was almost canceled by the top management of GOV. The D1 delivery was never put into production at GOV as planned but split in two, and one was eventually put into production in late 2013.

Need for Continuous ad hoc Coordinating

Most participants had experience with agile methods, but not in a comparable large-scale and complex setting as found in the Replacement project. Architectural principles, the use of off-the-shelf software components, integration with a wide and varied range of systems, and the case-handling process—these all resulted in numerous cross-team interdependencies in the project. This awareness was something that the team members learned during the process:

> *"No day passes without having to think about what the other teams are doing. After each sprint there is a demo that everybody attends, so based on that we know at least approximately what people are doing."* —Developer, ConsultGroup

The teams used the Jenkins tool for continuous integration and had several "information radiators" (large screens) across the open space office that continuously displayed results of the software build and test process. In addition, the developers created a system to alarm if a build broke. Moreover, the team architect had cross-team architecture meetings, and also implemented 'Scrum-of-Scrums.' Hence, these cross-team technologies and practices comprised a "boundary infrastructure" shared between the teams (Bowker and Star 1999; Star 2010).

However, these more pre-planned practices for coordination were not wholly sufficient. Complex interdependencies tended to arise unexpectedly, involving different types of problems and different individuals. Coordination efforts were needed as these dependencies and irregularities emerged. These issues need a more continuous *ad hoc coordinating:*

> *"By and large [coordination] is ad hoc. It was common practice to just walk over to each other [other teams] to discuss and solve issues there and then. And it was also a common understanding that such issues needed to be solved at once. And if [everyone] did so, this would certainly reduce the frictions between teams."* —Developer, ConsultGroup

This illustrates the need to focus on agency i.e. the *coordinating* and performative dimension of practices, and not only pre-planned practices and artifacts such as Scrum-of-Scrums, whiteboards, and common IT tools.

Discussion and Conclusions

In this paper we seek to challenge the existing underlying assumptions of large-scale agile software development. To that end, this section draws on both the analysis of the GOV case and also alternative theoretical assumptions portraying *complex socio-technical interdependencies* as essential characteristics of large-scale agile projects in order to problematize assumptions in the existing dominant literature. We draw from a multidisciplinary body of literature that conceptualizes the intertwined social and technical complexity in *boundary work* (Carlile 2004; Levina and Vaast 2005; Pawlowski and Robey 2004), boundary infrastructures (Bowker and Star 1999; Star 2010), performative conceptualization of coordinating (Jarzabkowski et al. 2012) and *complex technical systems* (Perrow 1999). This represents a conceptual shift from focusing on extrapolating traditional agile practices and principles from small to large-scale, and to complex socio-technical interdependencies across various 'local' practices, teams and software systems and subsystems. Consequently, we see large-scale agile as comprising quite different issues and types of challenges compared to smaller agile projects.

Re-conceptualizing Agile in the Large

The case study of large-scale agile at GOV illustrates a tension among a number of assumptions underpinning large-scale as presented in the extant literature. First, it is problematic, at least in the initial phases of large-scale projects, to 'scale up' the Product Owner role in order to manage a continuous stream of quite detailed issues regarding requirements and user needs. This issue, which is a typical in-house assumption (as it relates to one agile approach, Scrum), often requires on-going coordination between a team of Product Owners. Additionally, because of urgency and the potential consequences of such requirements in large organizations, Product Owners often need to act as *knowledge brokers* (Pawlowski and Robey 2004), translating between a wide variety of (external and internal) actors and the agile software development project teams and other roles. Similar situations can arguably also exist in small-scale projects. However, dealing with an increasing number of actors, interfaces with existing systems, and unexpected interdependencies – this is what distinguishes large-scale projects from traditional ones. As illustrated in the case narrative, the issue here is that interdependencies are usually neither completely known nor understood. This requires continuous coordinating *across* knowledge boundaries, and not only *within* boundaries (e.g. teams) (Carlile 2004). Consequently, it is not surprising that practices such as Scrum-of-Scrums have limitations for scaling agile in the large (Paasivaara et al. 2012). Adding Product Owners is only part of the solution. Competence and organizational capability to

work across boundaries as knowledge brokers (Pawlowski and Robey 2004) or boundary-spanners in-practice (Levina and Vaast 2005) is essential regardless of how many they are and how they are organized.

Second, the case study findings indicate that the related in-house assumption regarding knowledge transfer practices, the ideological assumption of self-organizing teams, and the paradigmatic assumption of *replacement* (Table 3) are all problematic. These assumptions emphasize intra-team knowledge sharing and coordinating, whereas the case study suggests the inter-team knowledge sharing and coordinating are not only crucially important, but also tend to be continuous activities intrinsic to large-scale development work. As shown in the case narrative, while managing unexpected interdependencies that are discovered as the work proceeds, coordination is urgently needed. Team members' *awareness* of what other teams, roles and individuals are doing becomes critically important. Social awareness across boundaries, or a well developed 'gut feel' of how one's work affects others, and vice versa, seems crucial in large-scale agile. Furthermore, as noted by Pawlowski and Robey (2004), this involves not only translation across different roles and communities, but *"evaluating and explaining the relevance to the recipient's practice"* (p. 664). In this way, team members increasingly become knowledge brokers who are not only focused on their fellow team members. But as a consequence of interdependencies with issues outside their local team, their attention turns increasingly towards various knowledge boundaries. Local practices in teams become more dependent upon other teams' practices so that teams can hardly be described as self-organizing.

Third, in contrast to the explanatory locus of traditional agile projects as "complex adaptive systems" that rely on self-organizing teams (e.g. Vidgen and Wang (2009)), teams are here more interdependent and need to work across knowledge borders, establishing boundary objects as ways of simultaneously standardizing across and preserving some local flexibility (Bowker and Star 1999; Levina and Vaast 2005). Again, coordinating mechanisms such as Scrum-of-Scrums do not work as intended because unexpected situations need to be acted on *in situ* often involving a varying ensemble of actors. In our case study, Scrum-of-Scrums were not reported as completely ineffective nor particularly difficult to conduct by our informants, but were simply not perceived as a coordinating mechanism of major importance. Likewise, studies from management science (e.g. Ingvaldsen and Rolfsen (2012)) suggest that inter-group coordination is a major challenge when groups are self-managing. The question is, then, what level of autonomy teams should have when collaborating with a number of other development teams. Are there models of organizing the development process that can grant team-level autonomy and still ensure efficient inter-team coordination?

Fourth, regarding the field assumptions in the literature, we submit that agile *can* be used in large-scale development projects, but involves managing complex socio-technical interdependencies. It is important to note that our perspective implies that interdependencies cannot be solved up front for a large-scale project. Thus, it is not the case that a project can simply adopt a mix of traditional waterfall and agile approaches (cf. Batra et al. 2010). In the GOV case study, for example, new interdependencies were discovered throughout the process. In the first deliverable it was the new off-the-shelf Oracle products and the service bus, but in the later stages it was related to integration with external information systems. It is certainly not the case that the actors involved are unaware that the software being developed should integrate with existing systems, but such integration is often more complex than anticipated, and thus relies on third party actors. Moreover, the lack of overview of interdependencies in complex systems leads to further unintended consequences as actors misinterpret the situation. In the case study, misunderstandings regarding the use of the service bus almost brought the project to a premature end.

Our analysis also illustrates the issue that while agile principles are often treated as sacrosanct and unchallenged in the existing literature, they are not necessarily applicable in large-scale agile. One cannot use the simple replacement model to invent another agile practice in order to fully solve complex interdependencies. Following Perrow (1999), in the case of complex socio-technical interdependencies in systems, extra add-ons in terms of components and routines, can lead to even more complex systems and networks of interdependencies.

Table 3 summarizes the predominant assumptions as they pertain to "large-scale agile" and includes a set of alternative assumptions that we offer for further deliberation and investigation. For example, one in-house assumption is that of collective code ownership—however, for large-scale systems involving many more actors and systems, this assumption willl not hold and alternative approaches will be necessary.

Table 3. Predominant assumptions of large-scale agile versus alternative assumptions	
Assumptions of large-scale agile	**Alternative assumptions**
In-House Assumptions	
Collective code ownership (Elshamy and Elssamadisy 2006)	Knowledge needs to be transferred, translated and transformed across boundaries (Carlile 2004).
Product owners can be scaled to manage customer requirements (Bass 2014)	Product owners and team members need to become knowledge brokers and boundary-spanners in practice to translate, evaluate and explain knowledge to others (Levina and Vaast 2005; Pawlowski and Robey 2004).
Root Metaphor Assumptions	
Complex adaptive systems and organisms (Cao et al. 2004; Vidgen and Wang 2009)	Large projects do not grow organically but scale up more abruptly, perhaps as a *Big Bang* when large projects are initiated with a large workforce from the outset.
Paradigmatic Assumptions	
Replacement (e.g. working software instead of comprehensive documentation; tests instead of specifications; user stories instead of requirements (Bjarnason et al. 2016; Meyer 2014)	Complex socio-technical interdependencies intrinsic to large-scale agile projects, make extrapolating practices and principles from small-scale and traditional projects problematic.
Composition - scaling up seen as 'linear' and unproblematic: Scrum of Scrums; user stories to epics; sprint backlogs to product backlogs (Bass 2014)	Because interdependencies are numerous, unexpected and emergent complex interactions across modules, teams and technologies (Perrow 1999), coordinating and knowledge sharing need to be conducted in situ as it happens with relevant actors. Coordinating is an ongoing and ad hoc activity between actors (Jarzabkowski et al. 2012).
Ideological Assumptions	
Self-organizing teams (Vidgen and Wang 2009)	Inter-team coordinating is more important than coordinating internally in teams (Ingvaldsen and Rolfsen 2012). This poses a threat to self-organization.
Value-driven in terms of 40-hour week; developer-oriented rather than manager-oriented	Increasing focus on multidisciplinary and multifaceted collaboration across and within teams and roles (cf. Nicolini et al. (2012)).
Field Assumptions	
Generally held assumption that agile is best suited to small projects, with co-located teams, and non-critical development contexts (Booch 2015; Williams and Cockburn 2003).	Agile can be used in large-scale development projects, but involves managing complex socio-technical interdependencies. Agile-Large not simply through hybrid approaches, but rather through handling complexities as they emerge.
Agile principles seen as axiomatic and sacrosanct, and tailoring of agile methods is achieved through adding practices.	Agile principles as "the best architectures, requirements, and designs emerge from self-organizing teams" should be questioned in large-scale development effort where teams need to balance the need for autonomy and local flexibility with more standardized ways of working.
Greenfield development context without dependencies on existing systems.	Brownfield development contexts involving dependencies on legacy systems and the teams involved in maintaining them.

Returning to our discussion of the concept of 'large-scale agile development,' this concept cannot be simply defined in terms of an absolute number of teams, lines of code, or person years *alone*. We argue that the number of teams is a central characteristic of large-scale development projects—as we have discussed, a definition should emphasize the high number of actors and interfaces with existing systems that add to the complex socio-technical interdependencies and which have implications for the development process. Simply adding new layers of roles and ceremonies does not suffice—in fact, one could argue this goes against the very idea of agility itself. Instead, we suggest that truly large-scale *agile* projects retain the ability to sustain flexibility and responsiveness while facing challenges that result from the complexities introduced by a sufficiently large number of actors, the number of systems involved, and the interdependencies between all.

Threats to Validity

We discuss potential threats to validity using Guba's (1981) criteria for assessing the validity of qualitative studies. The first criterion is **credibility:** The process of identifying assumptions from an existing body of literature is not a mechanical task but, as Alvesson and Sandberg acknowledge, depends partially on the insights and creativity of the researchers. Thus, the extent to which we identified assumptions is a potential threat to validity. To assess the validity of the assumptions, we triangulated across researchers: by extracting assumptions individually, comparing and discussing and categorizing them in several face-to-face and online discussions, and as a result we believe we have sufficiently addressed this concern. Another potential concern is that of **transferability**. The findings of the case study are necessarily specific to that particular case—however, challenges such as interdependencies are likely to be prevalent in other large-scale projects. We intend to conduct further studies in this area to evaluate and compare our findings. A third criterion is **dependability** – or the extent to which findings are reliable and can be traced back. We employed several tactics to establish the dependability of our study, including the triangulation across data sources. The longitudinal nature of the case study facilitates regular reflection on the case. The audit trail established by the interviews and their transcription, and the recording of our findings in several documents and spreadsheets helped to support traceability. Guba's last criterion is **confirmability**, and this is a potential threat to validity of qualitative studies given that the findings depend on the researcher's interpretation. We employed member checking as a tactic to ensure that our interpretation of the findings corresponded to the study participants' insights. As mentioned, we also employed triangulation across data sources, as well as triangulation across researchers—we discussed the assumptions, their classification and the case study findings through a number of face-to-face and online discussions. Finally, we also sent earlier drafts of this paper to the studied company, which is a form of member checking (Guba 1981).

Concluding Remarks

In this paper we seek to challenge common assumptions in the agile software development literature as they pertain to large-scale projects, the first such attempt to our knowledge. The Alvesson and Sandberg framework has proven useful for this endeavor. The problematization approach also proved fruitful when analyzing the case as it helped to critically consider assumptions from different perspectives, and to seek a more novel and alternative approach to theorizing. In this sense, the approach invited what Klein and Myers refer to as "dialogical reasoning" between the empirical material and the authors' theoretical assumptions (Klein and Myers 1999).

The Information Systems literature has devoted much attention to *how* studies should be conducted with much focus on high level research method issues—quantitative, qualitative and mixed methods, for example as well as epistemological issues. In contrast, IS researchers tend to use a gap-spotting approach to investigate research questions—without questioning whether those research questions are the right ones to pursue. Through our problematization of underlying assumptions and identification of alternative assumptions, we show how the problematization approach can generate new insights in an active area of research that is highly relevant to IS practitioners. Information Systems development is a field where practice tends to be ahead of research (Abrahamsson et al. 2009). The problematization approach adopted in this paper has generated new insights that can help researchers to ask better research questions, and offer valuable insights to IS practice. By offering a new set of assumptions for "Agile in the

Large," we aim to ask critical questions and suggest new avenues for research to help overcome the various barriers that practitioners might face in this area.

Acknowledgements

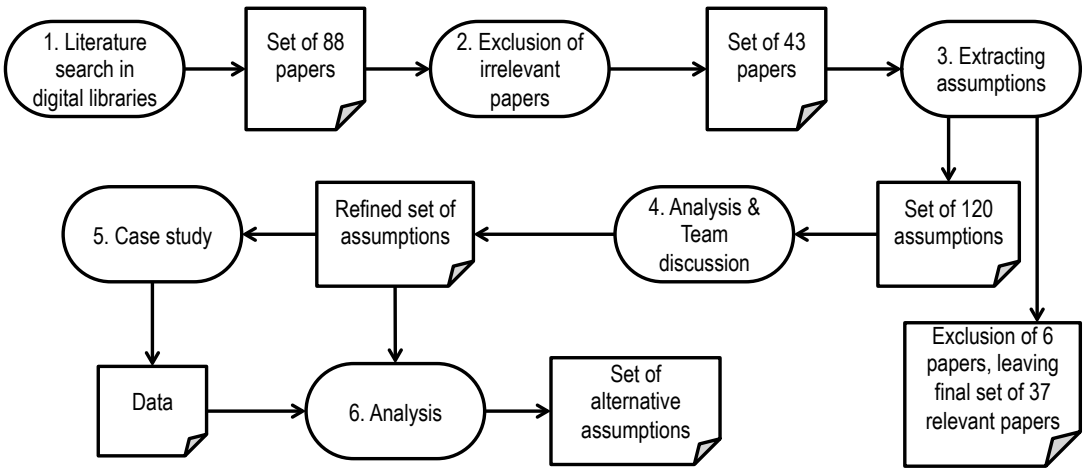
We are grateful to reviewers as well as to participants at the "research factory" at the Department of Computer and Information Science at the Norwegian University of Science and Technology for comments on earlier versions of this manuscript. The work was partially supported by the project Agile 2.0 (Research council of Norway grant 236759), Enterprise Ireland supporting the ITEA2 project SCALARE, grant no. IR/2013/0021, and Science Foundation Ireland grant 13/RC/2094 and co-funded under the European Regional Development Fund through the Southern & Eastern Regional Operational Programme to Lero—the Irish Software Research Centre (www.lero.ie).

Appendix

To identify relevant studies on large-scale agile, we used the following search string in ISI Web of Science and Scopus.

> *TITLE(("extreme programming" OR scrum OR (agile AND software) OR "agile development" OR "agile project*" OR "agile team*" OR "agile method*" OR "agile approach*" OR "agile practice*")) AND TITLE("large-scale" OR "large scale" OR (large* OR big* OR huge OR multi*) AND (organization* OR project* OR team*))*

The figure below presents the selection process of the articles and the identification of assumptions. The initial set of 88 papers was reduced to a set of 37 papers from which we identified 120 assumptions which were recorded in a spreadsheet. Through a careful selection process involving several discussions among the four authors, these were reduced and refined and used during the data analysis for the case study. The "analysis and team discussion" (Step 4 in the diagram) was done through several online meetings. It is important to note, as also discussed by Alvesson and Sandberg (2013) that the problematization methodology is necessarily subjective and depends on a researcher's creativity. However, the point of problematizing is not to identify assumptions per se, but rather to make explicit those assumptions that have remained implicit thus far, and then to question them.



References

Abrahamsson, P., Conboy, K., and Wang, X. 2009. ""Lots Done, More to Do": The Current State of Agile Systems Development Research," *European Journal of Information Systems* (18), pp. 281-284.

Ågerfalk, P.J., Fitzgerald, B., and Slaughter, S. 2009. "Flexible and Distributed Information Systems Development: State of the Art and Research Challenges," *Information Systems Research* (20:3), pp. 317-328.

Alvesson, M., and Sandberg, J. 2011. "Generating Research Questions through Problematization," *Academy of Management Review* (36:2), pp. 247-271.

Alvesson, M., and Sandberg, J. 2013. *Constructing Research Questions: Doing Interesting Research*. Sage.

Ambler, S. 2014. "2014 Agile Adoption Survey." from http://www.ambysoft.com/surveys/agileJanuary2014.html

Bacon, C.J., and Fitzgerald, B. 2001. "A Systemic Framework for the Field of Information Systems," *The DATA BASE for Advances in Information Systems* (32:2), pp. 46-67.

Bass, J.M. 2014. "Scrum Master Activities: Process Tailoring in Large Enterprise Projects," in: *International Conference on Global Software Engineering (ICGSE)*.

Batra, D., Xia, W., van der Meer, D., and Dutta, K. 2010. "Balancing Agile and Structured Development Approaches to Successfully Manage Large Distributed Software Projects: A Case Study from the Cruise Line Industry," *Communications of the Association for Information Systems* (27:1), pp. 379-394.

Beck, K. 2000. *Extreme Programming Explained: Embrace Change*. Addison-Wesley Professional.

Bjarnason, E., Unterkalmsteiner, M., Borg, M., and Engström, E. 2016. "A Multi-Case Study of Agile Requirements Engineering and the Use of Test Cases as Requirements," *Information and Software Technology* (In Press).

Bjarnason, E., Wnuk, K., and Regnell, B. 2011. "A Case Study on Benefits and Side Effects of Agile Practices in Large-Scale Requirements Engineering," in: *1st Workshop on Agile Requirements Engineering*.

Boehm, B. 2006. "A View of 20th and 21st Century Software Engineering," in: *International Conference on Software Engineering*. Shanghai, China: ACM.

Boland, D., and Fitzgerald, B. 2004. "Transitioning from a Co-Located to a Globally-Distributed Software Development Team: A Case Study at Analog Devices, Inc.," in: *3rd Workshop on Global Software Development*.

Booch, G. 2015. "Keynote at the 37th International Conference on Software Engineering: The Future of Software Engineering." from https://www.youtube.com/watch?v=h1TGJJ-F-fE

Bowker, G., and Star, S.L. 1999. *Sorting Things Out. Classification and Its Consequences*. MIT Press.

Cao, L., Mohan, K., Xu, P., and Ramesh, B. 2004. "How Extreme Does Extreme Programming Have to Be? Adapting Xp Practices to Large-Scale Projects," *Proceedings of the Hawaii International Conference on System Sciences,* R.H. Sprague Jr (ed.), Big Island, HI., pp. 1335-1344.

Carlile, P.R. 2002. "A Pragmatic View of Knowledge and Boundaries: Boundary Objects in New Product Development," *Organization Science* (13:4), pp. 442-455.

Carlile, P.R. 2004. "Transferring, Translating, and Transforming: An Integrative Framework for Managing Knowledge across Boundaries," *Organization Science* (15:5), pp. 555-568.

Cawley, O., Wang, X., and Richardson, I. 2010. "Lean/Agile Software Development Methodologies in Regulated Environments–State of the Art," in: *International Conference on Lean Enterprise Software and Systems (LESS), LNBIP 65*.

Conboy, K. 2009. "Agility from First Principles: Reconstructing the Concept of Agility in Information Systems Development," *Information Systems Research* (20:3), pp. 329-354

Conboy, K., Coyle, S., Wang, X., and Pikkarainen, M. 2011. "People over Process: Key Challenges in Agile Development," *IEEE Software* (28:4).

Conboy, K., and Fitzgerald, B. 2010. "Method and Developer Characteristics for Effective Agile Method Tailoring: A Study of Xp Expert Opinion," *ACM Transactions on Software Engineering and Methodology* (20:1).

Costa, N., Santos, N., Ferreira, N., and Machado, R.J. 2014. "Delivering User Stories for Implementing Logical Software Architectures by Multiple Scrum Teams," in: *14th International Conference on Computational Science and Its Applications, LNCS 8581*. Springer Verlag, pp. 747-762.

Coyle, S., and Conboy, K. 2009. "A Case Study of Risk Management in Agile Systems Development," in: *17th European Conference on Information Systems* S. Newell, E.A. Whitley, N. Pouloudi, J. Wareham and L. Mathiassen (eds.). Verona, Italy.

Dingsøyr, T., Fægri, T., and Itkonen, J. 2014. "What Is Large in Large-Scale? A Taxonomy of Scale for Agile Software Development," in: *Product-Focused Software Process Improvement, LNCS 8892,* A. Jedlitschka, P. Kuvaja, M. Kuhrmann, T. Männistö, J. Münch and M. Raatikainen (eds.). Springer International Publishing, pp. 273-276.

Dingsøyr, T., Nerur, S., Balijepally, V., and Moe, N.B. 2012. "A Decade of Agile Methodologies: Towards Explaining Agile Software Development," *Journal of Systems and Software* (85), pp. 1213-1221.

Dybå, T., and Dingsøyr, T. 2008. "Empirical Studies of Agile Software Development: A Systematic Review," *Information and Software Technology* (50), pp. 833-859.

Eisenhardt, K.M. 1989. "Building Theories from Case Study Research," *Academy of management review* (14:4), pp. 532-550.

Elshamy, A., and Elssamadisy, A. 2006. "Divide after You Conquer: An Agile Software Development Practice for Large Projects," in: *7th International Conference on Extreme Programming and Agile Processes in Software Engineering, XP 2006, LNCS 4044*. Oulu, Finland: Springer Verlag.

Elshamy, A., and Elssamadisy, A. 2007. "Applying Agile to Large Projects: New Agile Software Development Practices for Large Projects," in: *8th International Conference on Agile Processes in Software Engineering and eXtreme Programming, XP 2007, LNCS 4536*. Como, Italy: Springer, pp. 46-53.

Fitzgerald, B., Stol, K.J., O'Sullivan, R., and O'Brien, D. 2013. "Scaling Agile Methods to Regulated Environments: An Industry Case Study," in: *International Conference on Software Engineering*. San Francisco, CA, USA.

Flyvbjerg, B. 2006. "Five Misunderstandings About Case-Study Research," *Qualitative inquiry* (12:2), pp. 219-245.

Freudenberg, S., and Sharp, H. 2010. "The Top 10 Burning Research Questions from Practitioners," *IEEE Software* (27:5), pp. 8-9.

Gregory, P., Barroca, L., Sharp, H., Deshpande, A., and Taylor, K. 2016. "The Challenges That Challenge: Engaging with Agile Practitioners' Concerns," *Information and Software Technology* (In press).

Guba, E.G. 1981. "Criteria for Assessing the Trustworthiness of Naturalistic Inquiries," *Edu. Commun. Technol.* (29:2), pp. 75-91.

Gunyho, G., and Gutiérrez Plaza, J. 2011. "Evolution of Longer-Term Planning in a Large Scale Agile Project - F-Secure's Experience," in: *Lecture Notes in Business Information Processing*. pp. 306-315.

Hannay, J.E., and Benestad, H.C. 2010. "Perceived Productivity Threats in Large Agile Development Projects," *4th International Symposium on Empirical Software Engineering and Measurement, ESEM 2010*, Bolzano-Bozen.

Heikkila, V.T., Paasivaara, M., and Lassenius, C. 2013. "Scrumbut, but Does It Matter? A Mixed-Method Study of the Planning Process of a Multi-Team Scrum Organization," *2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2013*, Baltimore, MD, pp. 85-94.

Hirschheim, R., and Klein, H.K. 1989. "Four Paradigms of Information Systems Development," *Communications of the ACM* (32:10), pp. 1199-1216.

Hirschheim, R., and Klein, H.K. 2003. "Crisis in the Is Field? A Critical Reflection on the State of the Discipline," *Journal of the Association for Information Systems* (4:10), pp. 237-293.

Hirschheim, R., and Newman, M. 1991. "Symbolism and Information Systems Development: Myth, Metaphor and Magic," *Information Systems Research* (2:1), pp. 29-62.

Hoda, R., Noble, J., and Marshall, S. 2013. "Self-Organizing Roles on Agile Software Development Teams," *IEEE Transactions on Software Engineering* (39:3), pp. 422-444.

Iivari, J., Hirschheim, R., and Klein, H.K. 1998. "A Paradigmatic Analysis Contrasting Information Systems Development Approaches and Methodologies," *Information Systems Research* (9:2), pp. 164-193.

Ingvaldsen, J.A., and Rolfsen, M. 2012. "Autonomous Work Groups and the Challenge of Inter-Group Coordination," *Human Relations* (65), pp. 861-881.

Jarzabkowski, P.A., Lê, J.K., and Feldman, M.S. 2012. "Toward a Theory of Coordinating: Creating Coordinating Mechanisms in Practice," *Organization Science* (23:4), pp. 907-927.

Kähkönen, T. 2004. "Agile Methods for Large Organizations - Building Communities of Practice," *Proceedings of the Agile Development Conference, ADC 2004*, Salt Lake City, UT, pp. 2-10.

Kelley, J.F. 2007. "Keynote Address," in: *World Usability Day Conference*. Dayton, OH.
Kircher, M., Jain, P., Corsaro, A., and Levine, D. 2001. "Distributed Extreme Programming," in: *XP2001 - Extreme Programming and Flexible Processes in Software Engineering*.
Klein, H.K., and Myers, M.D. 1999. "A Set of Principles for Conducting and Evaluating Interpretive Field Studies in Information Systems," *MIS Quarterly* (23:1), pp. 67-93.
Langley, A. 1999. "Strategies for Theorizing from Process Data," *Academy of Management review* (24:4), pp. 691-710.
Levina, N., and Vaast, E. 2005. "The Emergence of Boundary Spanning Competence in Practice: Implications for Implementation and Use of Information Systems," *MIS Quarterly* (29:2), pp. 335-363.
Meyer, B. 2014. *Agile!: The Good, the Hype and the Ugly*. Springer.
Miles, M.B., and Huberman, A.M. 1994. *Qualitative Data Analysis: An Expanded Sourcebook*, (2nd ed.). Sage.
Moe, N.B., Šmite, D., Šblis, A., Börjesson, A.L., and Andréasson, P. 2014. "Networking in a Large-Scale Distributed Agile Project," in: *8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*.
Morgan, G. 1997. *Images of Organization*. Thousand Oaks, CA: Sage.
Myers, M.D., and Newman, M. 2007. "The Qualitative Interview in Is Research: Examining the Craft," *Information and Organization* (17:1), pp. 2-26.
Nerur, S., Mahapatra, R., and Mangalaraj, G. 2005. "Challenges of Migrating to Agile Methodologies," *Communications of the ACM* (48:5), pp. 72 - 78.
Nicolini, D., Mengis, J., and Swan, J. 2012. "Understanding the Role of Objects in Cross-Disciplinary Collaboration," *Organization Science* (23:3), pp. 612-629.
Orlikowski, W.J., and Baroudi, J.J. 1991. "Studying Information Technology in Organizations: Research Approaches and Assumptions," *Information systems research* (2:1), pp. 1-28.
Paasivaara, M., and Lassenius, C. 2014. "Communities of Practice in a Large Distributed Agile Software Development Organization - Case Ericsson," *Information and Software Technology* (56:12), pp. 1556-1577.
Paasivaara, M., Lassenius, C., and Heikkilä, V.T. 2012. "Inter-Team Coordination in Large-Scale Globally Distributed Scrum: Do Scrum-of-Scrums Really Work?," *6th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ESEM 2012*, Lund, pp. 235-238.
Pawlowski, S.D., and Robey, D. 2004. "Bridging User Organizations: Knowledge Brokering and the Work of Information Technology Professionals," *MIS Quarterly* (28:4), pp. 645-672.
Perrow, C. 1999. *Normal Accidents: Living with High Risk Technologies*. Princeton University Press.
Petersen, K., and Wohlin, C. 2009. "A Comparison of Issues and Advantages in Agile and Incremental Development between State of the Art and an Industrial Case," *Journal of Systems and Software* (82:9), Sep, pp. 1479-1490.
Razavian, M., and Lago, P. 2016. "Feminine Expertise in Architecting Teams," *IEEE Software* (33:4).
Robinson, H., and Sharp, H. 2010. "Collaboration, Communication and Co-Ordination in Agile Software Development Practice," in *Collaborative Software Engineering*. Springer, pp. 93-108.
Rolland, K.H., Ghinea, G., and Gronli, T. 2015. "Ambidextrous Enterprise Architecting: Betting on the Future and Hacking Path-Dependencies," in: *European Conference on Information Systems*.
Rubin, K.S. 2013. *Essential Scrum: A Practical Guide to the Most Popular Agile Process*. Addison-Wesley.
Sarker, S., Xiao, X., and Beaulieu, T. 2013. "Guest Editorial: Qualitative Studies in Information Systems: A Critical Review and Some Guiding Principles," *MIS Quarterly* (37:4), pp. iii-xviii.
Sharp, H., and Robinson, H. 2004. "An Ethnographic Study of Xp Practice," *Empirical Software Engineering* (9:4), pp. 353-375.
Stålhane, T., Myklebust, T., and Hanssen, G.K. 2012. "The Application of Safe Scrum to Iec61508 Certifiable Software," in: *ESREL 2012*. Helsinki, Finland.
Star, S.L. 2010. "This Is Not a Boundary Object: Reflections on the Origin of a Concept," *Science, Technology & Human Values* (35:5), pp. 601-617.
Stotts, D., Williams, L., Nagappan, N., Baheti, P., Jen, D., and Jackson, A. 2003. "Virtual Teaming: Experiments and Experiences with Distributed Pair Programming," in: *Extreme Programming/Agile Universe*.
Sundararajan, S., Bhasi, M., and Vijayaraghavan, P.K. 2014. "Case Study on Risk Management Practice in Large Offshore-Outsourced Agile Software Projects," *IET Software* (8:6), pp. 245-257.

Vidgen, R., and Wang, X. 2009. "Coevolving Systems and the Organization of Agile Software Development " *Information Systems Research* (20:3).
Walsham, G. 1993. *Interpreting Information Systems in Organizations*. Wiley.
Walsham, G. 1995. "Interpretive Case Studies in Is Research: Nature and Method," *European Journal of Information Systems* (4:2), pp. 74-81.
Walsham, G. 2006. "Doing Interpretive Research.," *European journal of Information Systems* (15:3), pp. 320-330.
Williams, L., and Cockburn, A. 2003. "Agile Software Development: It's About Feedback and Change," *IEEE Computer* (36:6), pp. 39-43.
Yin, R.K. 2013. *Case Study Research: Design and Methods*, (5th ed.). Sage.