The Journal of Systems and Software 86 (2013) 1333–1353

Contents lists available at SciVerse ScienceDirect

The Journal of Systems and Software

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

Agile requirements prioritization in large-scale outsourced system projects: An empirical study

Maya Daneva[a], Egbert van der Veen[a], Chintan Amrit[a,*], Smita Ghaisas[b], Klaas Sikkel[a], Ramesh Kumar[b], Nirav Ajmeri[b], Uday Ramteerthkar[b], Roel Wieringa[a]

[a] University of Twente, Enschede, The Netherlands
[b] Tata Research, Development and Design Centre, 54 Hadapsar Industrial Estate, Pune 411013, India

ARTICLE INFO

Article history:
Received 8 August 2011
Received in revised form
15 November 2012
Accepted 18 December 2012
Available online 29 December 2012

Keywords:
Agile requirements engineering
Requirements prioritization
Outsourced software development
Requirements dependencies
Large projects
Distributed project management
Qualitative research
Case study

ABSTRACT

The application of agile practices for requirements prioritization in distributed and outsourced projects is a relatively recent trend. Hence, not all of its facets are well-understood. This exploratory study sets out to uncover the concepts that practitioners in a large software organization use in the prioritization process and the practices that they deem good. We seek to provide a rich analysis and a deep understanding of three cases in an exploratory study that was carried out in a large and mature company, widely recognized for its excellence and its engagement in outsourced software development. We used in-depth interviews for data collection and grounded theory techniques for data analysis. Our exploration efforts yielded the following findings: (i) understanding requirements dependencies is of paramount importance for the successful deployment of agile approaches in large outsourced projects. (ii) Next to business value, the most important prioritization criterion in the setting of outsourced large agile projects is risk. (iii) The software organization has developed a new artefact that seems to be a worthwhile contribution to agile software development in the large: 'delivery stories', which complement user stories with technical implications, effort estimation and associated risk. The delivery stories play a pivotal role in requirements prioritization. (iv) The vendor's domain knowledge is a key asset for setting up successful client-developer collaboration. (v) The use of agile prioritization practices depends on the type of project outsourcing arrangement. Our findings contribute to the empirical software engineering literature by bringing a rich analysis of cases in agile and distributed contexts, from a vendor's perspective. We also discuss the possible implications of the results for research and in practice.

1. Introduction

Agile project development and management approaches are becoming the preferred choice for an increasingly large number of software organizations, be it large or small (Ramesh et al., 2010). A key pillar in any agile process of systems delivery is the close and continual collaboration between clients and developers (Maiden and Jones, 2010), which culminates in making project decisions that optimize both the client's and vendor's business value. The client–developer collaboration is also a well-recognized feature of most agile requirements engineering (RE) processes, and specifically – of requirements (re)prioritization. Reprioritizing requirements at inter-iteration time plays a pivotal role in agile projects (Racheva et al., 2010). However, the question of how the reprioritization happens in real life has been investigated only in certain contexts, for example, in small and medium sized projects (Eckstein, 2004; Ramesh et al., 2010). Our literature review revealed that in the area of agile RE, relatively few empirical studies address the project realities of large and distributed projects (Barlow et al., 2011; Sarker and Sarker, 2009). Even fewer studies focus on the complex RE settings unique to outsourced software development (Sarker and Sarker, 2009). Also, in the area of global software engineering (GSE), a tertiary study by Hanssen et al. (2011) on the signs of agile trends in GSE research found that most empirical studies address the context of small and medium size projects.

In this paper we explore the trade-off between the value for clients vs the value for vendors in agile requirements prioritization, in large and distributed projects, by means of an embedded case study (Yin, 2008; Scholz and Tietje, 2002). This case study was carried out in a context whereby three large, multi-site enterprise systems were delivered under a contract between a

* Corresponding author.
E-mail addresses: m.daneva@utwente.nl (M. Daneva), e.vanderveen@student.utwente.nl (E. van der Veen), c.amrit@utwente.nl (C. Amrit), smita.ghaisas@tcs.com (S. Ghaisas), k.sikkel@utwente.nl (K. Sikkel), ramesh.kumar@tcs.com (R. Kumar), nirav.ajmeri@tcs.com (N. Ajmeri), uday.ramteerthkar@tcs.com (U. Ramteerthkar), r.j.wieringa@utwente.nl (R. Wieringa).

0164-1212/$ – see front matter © 2013 Elsevier Inc. All rights reserved.
http://dx.doi.org/10.1016/j.jss.2012.12.046

large and mature outsourcing vendor in Asia and three dispersed client organizations. The overall research objective was to uncover how mid-course requirements prioritization takes place in large projects in industry, and what notions of value are included in it from the vendor's perspective.

We performed an embedded case study (Scholz and Tietje, 2002) that was exploratory in nature. Our purpose was to 'look under the hood', to observe and identify those mechanisms that drive the requirements prioritization as a risk-value-balancing process. The outcome of the study is a rich analysis of how three real-life cases coped with RE when delivering large systems to clients as part of an outsourcing relationship, and doing it successfully.

The research reported in this paper offers two contributions: to agile RE, in particular, and to empirical SE, in general. First, while most case studies dealing with agile RE have focused on small-medium project organizations, our research explicitly specifies the RE knowledge pertaining to those practices which (depending on the outsourcing arrangement) work in large projects associated with a large and mature outsourcing vendor. Knowledge about how to adopt agile practices in large organizations has accumulated over the past years (Hanssen et al., 2011), but this knowledge has been evaluated from a client organization's perspective, not from a vendor's perspective. In a systematic review on distributed software development, Prikladnicki and Audy (2010, p. 790) called for more studies from the vendor's angle. Furthermore, the reviewed studies focused on problems associated with agile in the large, not on the solutions. The present research yields a rich analysis aimed at understanding the vendor from a technical standpoint. In particular, we explicate what the vendor does within its own walls in order to make agile practices work for a client in an outsourcing setting. Our explicit focus on what the vendor found to work is a unique feature of this study. This paper also provides a direct response to two other calls: (1) the call by the empirical SE community for more empirical research on which SE process to use in which specific context (Sjøberg et al., 2007) and (2) the particular call by the RE community for more empirical research in the sub-areas of RE (Cheng and Atlee, 2007).

In what follows, we first present our motivation, and then in Section 2, we review related work on agile RE in large projects. We then describe our embedded case study research design, its execution, and its outcomes. Finally, we discuss the results and the limitations of the study, reflect on the experiences and the practical and theoretical implications, and conclude with suggestions for future research activities.

1.1. Motivation

Our motivation for this empirical research is grounded on the recent trends in the software industry, indicating the increased diffusion of agile into large projects in distributed and outsourcing contexts. Facing the fact that we, as a community, know relatively little of how agile should or could work, we felt motivated to initiate empirical research so as to approach this knowledge gap systematically. The authors of the present paper brought relevant professional background and experience to this task. While some of the authors have been involved in empirical research on agile RE practices (Racheva et al., 2010), others have direct access to the type of projects and the organizational settings important to our understanding of the phenomenon of agile requirements prioritization. In 2009 two of the authors (Daneva and Sikkel) were actively involved in a systematic literature review (Racheva et al., 2009) on business value in agile RE, with the conclusion that the phenomenon of agile requirements prioritization was only partly explored. The review found no studies that clearly indicated how exactly individual agile practices or groups of such practices not only create value, but also keep accumulating it over time, while minimizing the risk for the vendor. In 2010, three of the authors (Daneva, Sikkel and Wieringa) studied the phenomenon of agile requirements prioritization in small and medium-size agile companies. The result of this work was a descriptive model explaining how agile requirements prioritization happens in agile small and medium project settings. Drawing on these prior experiences, our motivation for the present embedded case study was to understand the mechanics of agile RE practices in large organizations. We acknowledge that large and complex projects (for example, distributed outsourcing projects) are likely to have many interdependencies and to require more coordination. This in turn, forces the organizations to adopt standardization and planning as a less expensive form of coordination (Barlow et al., 2011). As we will see in Section 2, little is however known about how a large organization with explicit standardization policies and norms engages in agile projects, while preserving the coordination forms on the vendor's side and delivering agile business value on the client's side. The fact that not much has been published beyond the four high-visibility research papers (Barlow et al., 2011; Sarker and Sarker, 2009; Hanssen et al., 2011; Prikladnicki and Audy, 2010) and anecdotal evidence, motivated us to provide insight into actual agile RE in the large. In the next section we report on published empirical studies and indicate that there is an important gap in knowledge.

2. Related work

This section summarizes related empirical research publications on agile requirements prioritization in large projects and draws on the consequences for our research.

Essentially, requirements prioritization is a decision-making process (Alenjung and Persson, 2008; Herrmann and Daneva, 2008). Therefore, in this section we evaluate the related work from a decision-maker's perspective. According to Alenjung and Persson (2008), such a perspective implies looking into the roles involved, the prioritization criteria being used, the contextual settings affecting the prioritization process, and the kind of trade-offs being made. Our choice of analysing the agile requirements prioritization process in this particular way agrees with the general characterization of a software process recommended by Shari L. Pfleeger (2001). According to her, a process is defined by people's roles, artefacts (used or produced), and resources. In Section 2.1 we present the kind of studies dedicated to agile software engineering in the large. In Section 2.2 we evaluate in detail what these studies say about the characteristics of the requirements prioritization process from a decision-making perspective. In Section 2.3 we discuss the literature and motivate our research questions.

2.1. Studies on agile RE in the large

To identify related work on agile RE at the large, we searched the Scopus digital library for papers published in the major agile conferences (AGILE and XP). We also looked at the papers included in the 2010 and 2011 systematic literature reviews that dealt with agile topics in global software engineering (Jalali and Wohlin, 2010), in distributed software development (Prikladnicki and Audy, 2010) and in large projects in general (Barlow et al., 2011). These reviews provided examples of the kinds of large projects and organizations described in recent publications by both researchers and practitioners. The relevant published sources indicated that many large organizations were indeed preoccupied with the question of how agile scales up. In most publications a discussion is provided by the respective company as to what works in their settings. We note that all the reviewed literature sources (and included in this section) treated RE as part of agile SE for large projects. In spite of our best efforts, we could not find a report that was specifically

Table 1
Selected literature sources on agile at large.

Source	Business sector	User stories	Modified agile practices	Was prioritization discussed?
Auvinen et al. (2006)	Mobile systems	☑	☑	Hints
Elshamy and Elssamadisy (2007)	IT service delivery (a vendor)	☑	☑	Hints
Eckstein (2004)	Telecom	☑	☑	Explicitly
McDowell and Dourambeis (2007)	Telecom	☑	–	Not discussed
Sulfaro et al. (2007)	National Post Services	☑	–	Not discussed
Koehnemann and Coats (2009)	IT service delivery (a vendor)	☑	–	Not discussed
Gat (2006)	Enterprise IT infrastructure management solutions	☑	–	Explicitly
Larman and Vodde (2010)	Large government organizations	☑	–	Explicitly
Gary et al. (2011)	Safety-critical systems	–	☑	Hints
Grewal and Maurer (2007)	Oil & gas (a vendor)	–	☑	Hints
Sutherland et al. (2007)	Digital library management (a vendor)	☑	☑	Hints
Valade (2008)	ERP implementation	☑	☑	Hints
Bosch and Bosch-Sijtsema (2011)	Software product business (a vendor)	☑	☑	Hints
Hong et al. (2010)	E-commerce	☑	☑	Hints
Kendall et al. (2010)	Physics-based computational engineering	–	☑	Not discussed
Christou et al. (2010)	Banking	–	☑	Not discussed
Shatil et al. (2010)	Defence system engineering (a vendor)	☑	☑	Hints
Hajjdiab et al. (2012)	Government entity	–	☑	Not discussed

dedicated to agile RE in the large. Table 1 summarizes the examples of large projects found using agile and indicates the business sector of the company, whether the requirements were engineered according to the user-story approach peculiar to agile, and whether the publication reported the use of specific new or modified agile SE practices. The last column indicates whether the authors contextualized the use of specific practices in the observed setting, or parts thereof. Table 1 includes empirical reports which (i) refer to projects with more than 60 members of staff and (ii) describe agile RE practices in detail. We do not claim that the list of included sources is complete, but represents a sample with the relevant agile requirements prioritization in contexts similar to ours (large, distributed and outsourced). In Table 1, the '☑ ' or '–' marks signify whether or not the use of user stories and new/modified agile practices were clearly stated in the paper. The last column describes to what extent the requirements prioritization process was discussed.

Table 1 indicates that large project organizations which deliver software systems in a variety of application domains have been rethinking their requirements documentation from an agile perspective and have been adopting the user story approach to requirements. Those who do not adopt the user stories as an approach for their business requirements, redesign their requirements specification practices to "become more agile" (Grewal and Maurer, 2007; Kendall et al., 2010; Christou et al., 2010; Gary et al., 2011). We observed that while the literature sources discuss the user story elicitation and documentation practices explicitly, they mostly provide hints regarding the requirements of the (re)prioritization process at inter-iteration time. This implies that the RE community is exposed to comparatively little empirical evidence on how large-scale agile requirements prioritization happens and, in turn, little is known about whether theory (e.g. the recommendations of leading agile methodologists in their books) and practice are always consistent. As indicated by Racheva et al. (2010), deviations of the real processes from the principles of agile development are rarely reported – which does not necessarily mean that they do not exist.

2.2. Requirements prioritization

In this section we present what the sources say with respect to the characterizing aspects of agile requirements prioritization from a decision-making perspective.

2.2.1. Roles in the decision-making process

All the referenced sources in Table 1 indicate that the ultimate decision-making on priorities lies with either the product owner or the client. However, information inputs into decision-making are provided by a number of team players on the vendor's side. The literature suggests that this is necessary because in large projects there is a gap between (i) the level of detail in the user stories provided by the client for iteration planning purposes and (ii) the level of detail in the architecture and the system level features that developers need for effort estimation purposes. To close this gap, clients and vendors typically apply a 'team of teams' approach (Eckstein, 2004) or a 'stream of streams' approach (Grewal and Maurer, 2007). Examples of roles introduced by companies to large agile projects to provide input into requirements negotiation and decision-making are: (i) 'requirements architects' (responsible for "taking the high-level features defined by product management and decomposing these, on a just-in-time basis, into the more detailed requirements and stories needed to drive iteration planning" (Gat, 2006)) and (ii) business area owners (Larman and Vodde, 2010). The people in these roles collaborate with the overall project manager (or the scrum master) who is in charge of making sure that teams are working in parallel and in a timely manner. The scrum master is well aware of dependencies among teams and uses this knowledge to resolve them as soon as they arise (Hong et al., 2010).

2.2.2. Decision making criteria for requirements priorities at inter-iteration time, in an outsourced context

The agile RE studies suggest the generated business value is the key prioritization criterion for the client and for the vendor, respectively. The agile literature sources implicitly assume that the client's value and the vendor's value are both attributes of a large agile project (Eckstein, 2004; Cheng and Atlee, 2007; Larman and

Vodde, 2010; Grewal and Maurer, 2007; Bosch and Bosch-Sijtsema, 2011; Kendall et al., 2010; Christou et al., 2010; Shatil et al., 2010). Yet, how these two concepts relate to each other, e.g. whether they are opposing forces, and if so, how to balance them in a way that is a win–win for both parties, seems to be unknown. To the best of our knowledge, there has not been an empirical study to explain the process of risk-minimization during agile RE. Moreover, our earlier research on business value generation by means of agile prioritization (Racheva et al., 2010) also found that very little research has been carried out with respect to business value. Hence, how these should be balanced in an outsourcing situation is yet to be fully understood and documented.

2.2.3. Contextual factors affecting requirements prioritization

The agile-in-the-large RE literature discusses four characteristics of the project setting which influence decision-making on requirements (re)prioritization at inter-iteration time: (i) the *need to embrace change*, (ii) *project constraints*, (iii) *scope* and (iv) *the number of project staff*. However, while (i) and (ii) have an impact on how agile requirements prioritization happens in both large and small projects, the agile authors point out that characteristics (iii) and (iv) are unique to large projects. Eckstein (2004), a prominent agile author refers to them as "first order dimensions" that define the 'largeness' of a project. *Scope* characterizes the complexity of the requirements, while the *number of project staff* is related to the exposure to risk: the more people involved in a project, especially the larger the number of client representatives, the higher the risk of contradictory requirements. The agile literature sources (Table 1) acknowledge that agile RE practices need to be implemented differently in large projects because at a specific project size and mode of execution (such as offshore outsourced) "things do not work out the normal way anymore" (Eckstein, 2004), and because new problems may surface due to the largeness of the team (and those are easier to handle in small teams).

2.2.4. Value creation trade-offs: vendor vs client

Eckstein (2004), Larman and Vodde (2010), and Grewal and Maurer (2007) indicate that a win–win client–vendor collaboration is possible if the project team is aware of the need to balance vendor's value and client's value. However, the question of what is a 'good' balance and what is a 'good' way to achieve it in specific contexts, is by and large under-researched. Racheva et al. (2010) give a clear indication that the vendor and the client perspectives differ. However, the way in which these two perspectives may differ depends on project-specific context factors, e.g. if the contractual agreement is fixed-price, or the level of trust between client and vendor. Eckstein (2004) and Larman and Vodde (2010) provide advice on how to consolidate these two perspectives in a project. However, the advice is in the form of guidelines, assuming that the readers have enough knowledge and experience to imagine how the implementation of these guidelines would work in their own setting.

2.3. Implications of the related work and research questions for our study

Reflecting on what is known from empirical agile RE literature on our phenomenon of interest (agile requirements prioritization in the large), one might ask why so little, or why the important facets of the agile prioritization process (roles, criteria, context and trade-offs), are under-researched. There are various possible reasons. Generally, there are many more small projects than larger projects. Ambler (2008) indicates that most agile teams have less than 10 people and are co-located. It is easier to turn smaller projects into success stories. In contrast, large projects have a higher chance of failure and, moreover, extending agile practices to large projects is a more recent trend. Eckstein (2004), a prominent agile-at-the-large author suggests that fewer experiences from large projects are published merely because the failure rate of large projects is generally higher. As we know, failed projects are less often the context of empirical research published in scientific journals (Sjøberg et al., 2007). Specifically regarding agile, Cohen et al. (2004) observed: "there is a lack of literature describing projects where agile methods failed to produce good results. There are a number of studies reporting poor projects due to negligent implementation of agile method, but none where practitioners felt they executed properly but the method failed to deliver on its promises".

There are recent examples of successful agile-in-the-large projects. However what these publications do not provide is a thorough analysis of the underlying mechanisms that make these projects successful. Also, as described in the introduction, few papers focus on the complex RE settings unique to outsourced software development (Sarker and Sarker, 2009). For example, Batra (2009) states that in outsourced projects, requirements sign-off is usually done after initial discussions between project managers and clients and there is little scope for negotiations or reprioritizations throughout the development life cycle. This practice is followed in order to overcome the challenges posed by distance and limited in-person communication thereof. Agile RE however assumes frequent discussions, negotiations and reprioritizations. Doing justice to agile practices in outsourced projects is a challenge, and it is even more so in fixed price contractual projects. Furthermore, it is also unclear how different outsourcing arrangements (Dibbern et al., 2004) can impact agile RE practices. For example, the agile RE work practices of fixed price projects could differ from more flexible cost/in-sourcing projects. More specifically, the question of how the client and the vendor can balance value generation with agile has only recently come to the attention of practitioners and researchers. As a result, the phenomenon is only partly understood. Yet, while justifiably deemed important in any project, we note that in large projects – where large vendors and large clients are contractually bound, poor value and risk trade-offs might often have severe consequences (Eckstein, 2004).

These issues and gaps in literature have motivated our following research questions (RQs):

RQ1: Who are the decision makers in the agile requirements prioritization process? What are their roles and what are they responsible for?
RQ2: What prioritization criteria do large project teams use to make value driven decisions at inter-iteration time in an outsourced mode?
RQ3: What is the relationship between project settings and requirements prioritization?
RQ4: How does the vendor's team combine value creation for their own organization with value creation for their client?

We conducted an embedded case study with the aim to answer these questions. Our case study along with a discussion of the resulting insights could help reduce and eventually fill this gap in knowledge. We present our research methodology and describe the embedded case study in more detail in the next section. In Section 4 we discuss the results of our embedded case study along with specific answers to the research questions posed above.

3. Research method

Our embedded case study design follows the guidelines by Yin (2008) as well as Scholz and Tietje (2002). We chose the embedded case study form because we wanted to obtain a detail-rich, holistic and contextualized description – from multiple projects

undertaken by a large outsourcing vendor, regarding how requirements reprioritization took place. Yin (2008) makes a distinction between a holistic and an embedded case study. While a holistic case study examines the global nature of a program or an organization, an embedded case study includes outcomes from individual projects within the program. We applied purposeful sampling (Charmaz, 2007) when looking for suitable projects. Our purpose was to find projects that had dispersed users, a large size (expressed in person/years to complete a project), and a project team that had already had some experience in using an agile delivery model, or had at least used a collection of agile practices. Hence, as described in Section 3.1, our embedded case study involved more than one unit, or object, of analysis (Charmaz, 2007).

As the context of this study refers to large projects in a large organization and little is known about the phenomenon of interest in this context (see Section 2), we kept an open mind and did not have any preconceived ideas of what the answers to our research questions would be. Recognizing the nascent stage of knowledge on our phenomenon of interest, we designed our exploratory case study process by including the following steps:

(1) compose an interview questionnaire;
(2) validate the questionnaire with the help of an experienced researcher;
(3) implement changes to the questionnaire based on the feedback;
(4) do a pilot interview to check the applicability of the questionnaire in a real-life context;
(5) carry out in-depth interviews with practitioners according to the finalized questionnaire;
(6) sample (follow-up those participants who possess deeper knowledge or a more specific perspective).

Consistent with the fact that our study was exploratory and that we wanted to get a rich analysis of real-life cases, we used in-depth interviews (King, 2010) as a data collection technique and grounded theory (Charmaz, 2007) as a data analysis technique. The in-depth interviews technique was selected for two reasons: (1) it is a suitable technique for an inquiry like ours, and (2) the resulting data offers a robust alternative (King and Horrock, 2010) to more traditional survey methods. This is especially the case when the absolute number of participants is less important than a rich investigation of content. We triangulated the data collected from multiple sources (e.g. participants across three different projects and in different roles, whose experiences varied broadly based on the specific role each one played). This was done to ensure that our interviews provided a multidimensional image of composing activities in our particular project setting.

Our data analysis was guided by the reasoning and the logic that underlies the sense-making techniques associated with less procedural versions of the grounded theory (GT) (Bryant and Charmaz, 2007). Specifically, we applied the techniques of coding and constant comparison as recommended by Charmaz (2007). These techniques helped us to identify concepts grounded in the collected data and to link these concepts to higher-level categories. Our choice of using GT for data analysis agrees with Matavire and Brown (2011) who profiled the use of GT in information system research. A notable example of a qualitative study of this nature is the one by Ramesh et al. (2010). Below, we provide a summary of GT as a research method, and in Section 3.3 we present how the GT sense-making techniques were used for data analysis.

GT is a qualitative method applied broadly in social sciences to construct general propositions (referred to as "theory" in this approach) from verbal data. This approach is exploratory and well suited for situations where the researcher does not have pre-conceived ideas. By this, GT methodologists (Charmaz, 2007) mean situations in which the researcher does not start with any hypotheses or a predefined theory which requires proof. Instead, the researcher is driven by the desire to capture all facets of the collected qualitative data and then allows the "theory" to emerge from the data. In the field of empirical SE, researchers have been using GT to find answers to questions that address relatively "uncharted land" (as Baskerville et al. (2011) refer to it), or a phenomenon about which little is known. Recent examples of GT studies in SE include the publications by Baskerville et al. (2011), Coleman and O'Connor (2008), and Rose et al. (2007). In the RE field, examples of applications of GT as an empirical research approach were published by Urquhart (1997), Martin et al. (2009), Martin et al. (2012), and Ramesh et al. (2010).

Grounded theory can be used for data collection and data analysis, as done in the studies of Urquhart (1997), Martin et al. (2009), and Martin et al. (2012). In other studies, like the one by Ramesh et al. (2010), it is used only for data analysis. We followed the latter approach. The GT process is presented in more detail in Section 3.3.

3.1. The data collection process

We carried out 16 interviews in total. They were about an hour and a half long on average. The interviews were conducted either face-to-face or facilitated through video conferencing. In each of the embedded cases, the first interview was treated as a pilot interview. A special set of factual questions related to the context, scope, etc. of the project, were asked in these interviews. Before the start of an interview, the interviewee was provided with information on the research purpose, the research process and the rights and responsibilities of the companies participating in the embedded case study. During the meeting, one researcher (van der Veen) and the interviewee went through the questionnaire, which served as a guide to the interviews. The questionnaire was composed of two parts: the first part discussed the prioritization practice that each interviewee experienced in a project, while the second part included questions related to risk, business value perception and how value and risk were balanced as an essential part of the requirements prioritization decisions. The rationale behind this structure was to focus the attention of the participants on a concrete example, and then clarify the consideration of value for the client and for the vendor as part of the prioritization decision-making process. Here, we note: (1) no substantial changes in both the questionnaire and case study protocol took place after the pilot interviews, thus these could be considered as being an integral part of the case study and (2) during the interviews there were instances when questions, other than those included in the questionnaire arose. These questions had not been previously anticipated; however, the researcher conducting the study considered them interesting and pursued the interview in that direction.

3.2. The case study project organizations and interview participants

To investigate the vendor's perspective of how value is created in agile RE, we examined three separate projects within the larger organization. All projects used Agile, a form of Scrum that was adapted to the specific context. All the projects practiced Agile RE, with (re-)prioritization occurring regularly. Finally, all were distributed projects, with at least two geographically separated locations being involved. Our study included 16 practitioners working in various capacities on one of the three projects. This allowed a variety of perspectives of the studied phenomena to be included, which in turn, ensured triangulation of data in the research process (Yin, 2008; Patton, 1999).

The case study projects are described in Table 2 in terms of (i) type of engagement, (ii) scope, (iii) number of team members (staff), (iv) contractual agreement regarding pricing structure and

Table 2
Details of the different embedded case study projects.

Outsourcing project	Project Alpha	Project Beta	Project Gamma
Type of engagement (outsourcing arrangement)	Single external client	Collaborative external client	Inter-departmental project
Scope	Large	Large	Medium
Number of team members	Nearly 300	35–40 + Client Team ± 100	35–40 + client team ± 50
Contractual agreement regarding cost and duration	Fixed	Flexible	Flexible
Size in person/years	400	343	85
Modularity of product architecture	Low	Low	High

project duration and (v) project size in terms of person/years. For confidentiality reasons, we refer to these projects as project Alpha, project Beta, and project Gamma. The three projects had a number of broad similarities, as well as differences. All projects were outsourcing contracts. In terms of scope, projects Alpha and Beta were large. Gamma on the other hand, was a medium size project. Concerning the nature of contractual agreement between vendor and client, Alpha was a fixed-price/fixed-duration project, while Beta and Gamma had a flexible timeline.

Furthermore, the projects were called "agile", but in reality a mix of agile and structured practices were used. The interaction provided to the client incorporated all the required agile practices: delivery of working (and tested) software in regular increments, intensive contact with the client, incorporating changes as appropriate, and on-site customer representation throughout the project. Internally the vendor complied with those agile practices needed to make the project 'agile' to the external world. However, the project was organized along traditional lines, hierarchical in structure and supported by extensive documentation. A CMM-5 organization cannot really make allowances for alternatives (Barlow et al., 2011). Also, software developers cannot change the core values from one project to the next (see Banerjee et al. (2011), for an account of the tensions encountered by a CMM-5 certified vendor when working on agile projects). It is the vendor's core value to satisfy the client, and a mature organization should set up the processes accordingly.

Specific case study participants were selected from the three projects, based on the following criteria: (i) they had common professional characteristics pertaining to the topic of our study and (ii) they had the potential to offer information-rich experiences.

The list of case study participants is presented in Appendix A, indicating the details of the participants' roles in the company and the mode in which the respective interviews were conducted.The projects Alpha, Beta and Gamma were successfully completed, although, they were still running projects, at the time of our analysis (December 2010 to August 2011). We will now describe the contextual setting of each case study project in more detail.

3.2.1. Project Alpha

The practitioners in project Alpha were part of two programs involved in the development of a large software solution in the application domain of insurance process automation. The practitioners were experienced in working on large projects with the objective to deliver a large enterprise system to a client in the insurance business. The system was aimed at automating the core business processes of this client's organization.

The development project teams were co-located, and the client teams were dispersed globally. The case study participants included: one scrum master, one business analyst, two business analyst leads, one delivery head (responsible for transforming the clients' user stories into 'delivery stories', including architecture design decisions and non-functional requirements), one portfolio manager (who was responsible for the group of clients' projects, managed as a portfolio), and one test scenario team lead (responsible for end-user acceptance testing and making sure requirements are testable and verifiable).

3.2.2. Project Beta

This project had around 35–40 people in the team, whose function was to complement the existing development team on the client's side, leading to a total of about 150 people. The team was not engaged in a specific project, but was in place in order to enhance the client's existing development effort. At the time of the interview, this engagement had been in operation for three years with Agile as the adopted development methodology. Various products were under active development, and these products were divided into 'releases'; each release consisted of a number of sprints of either two or four weeks. The requirements here were in a constant flux, with the mind-set to accommodate requests from the business, whenever possible. But, following good practice, requirements changes were taken up in between sprints, not during a sprint.

3.2.3. Project Gamma

In this setting, the client was internal, but not co-located. Unlike the other two projects which had external clients, this project developed a product for an internal client, who then used the product to develop solutions for an external customer. For the purposes of this paper, we will refer to the system development team as the 'development team' and the internal client's team as 'the client' (as there was little to no direct interaction between the first development team and the external client). The project consisted of 85–90 people, with 35–40 people on the developer's side and 50 on the client's side (again, we refer here to the internal client). The product was a model-driven development environment. In contrast to projects Alpha and Beta, the Gamma development team actually owned the product, and used its interaction with the client as a way to expand the product with the aim to make it widely applicable in a broad range of settings. As the client used the tool for their own development, they actively discovered bugs or limitations and sent the change requests to the development team. The development team then tried to accommodate these requests as quickly as possible so as not to slow down the client's development effort. In contrast to Alpha and Beta, the requirements for the products in Gamma were not only determined through feedback from the client, but were also influenced by the fact that the development team had its own long-term plans for the tool and worked towards those plans. Another factor influencing requirements was a separate process of internal proof-of-concept development for showcasing the tool's capabilities. As the tool's development was part of a broader research initiative, the research team also played a role in designing the project. Although the model-driven development environment project had been underway for at least 10 years, the development team had switched to Agile only four months before the interviews were conducted. This transition was triggered by the advantages that the client team had achieved through their own transition to Agile earlier. Individual sprints were four weeks in duration at the start of the transition, but were later extended to six weeks to allow for more time for testing and consolidation. Between the sprints, a period of one or two weeks was used for planning the next sprint (in the other two projects, this happened concurrently). As with project Beta, changing

requirements were accommodated whenever possible, but intra-iteration re-prioritization was kept to a minimum.

3.3. The data analysis process

The interviews were tape-recorded and then transcribed by one researcher (van der Veen). The analysis was supposed to be done in collaboration, but one was located in Asia (van der Veen) and the other researcher (Daneva) was in Europe, thus the researcher responsible for data collection (van der Veen) took special care of the transcribed information. For example, he added notes about the non-verbal language of his interviewees, and clearly indicated whenever an interviewee paused and expressed doubts regarding the completeness of the information provided. This turned out to be critical information for the other researcher who was engaged in the coding. Once the transcripts were ready, the data analysis guidelines of the GT approach (Charmaz, 2007) were applied. Essentially, GT analysis includes 'coding' and 'constant comparison' of the interview data. Coding is a way of learning to know the data. It is the process of conceptualizing the data by reading the data line-by-line and marking a segment of data with a descriptive word. Constant comparison is a process by means of which one constantly compares instances of data that are given a specific category name, with other instances of data, to see if these categories fit and are workable. This process helps in grouping the data into categories. The resulting codes and categories guide the writing-up of the results and aid in improving the accuracy of the claims (Charmaz, 2007).

GT methodologists recommend iterative coding and constant comparison. Two researchers (van der Veen and Daneva) coded the interview text independently, at different locations and with little communication between them. This was done to ensure code validity. However, the results of the coding and interpretation of the data were discussed and peer-reviewed iteratively with two other researchers (Amrit and Sikkel), to establish consistency and categorization of the emerging clusters. Our analysis proceeded as follows: two researchers first read the interview texts and attached a coding word to a portion of the text – a phrase or a paragraph. The coding words were selected to reflect the relevance of the respective portion of the interview text to a specific part of the studied phenomenon. This could be a concept (e.g. 'requirements dependency', 'technical debt'), or an activity (e.g. 'development effort estimation'). Some of the codes were a logical continuation of the composition of the interviews, as standard aspects of the process were discussed, e.g. 'size of the team' or 'decision-maker'. In the case of specific incidents, we asked the interviewee what concept or activity the interviewee had been talking about, which was duly noted. We then clustered all pieces of text that related to the same code, in order to analyse it in a consistent and systematic way. Then the codes were compared to each other for the purpose of searching for similarities and differences between them. This inter-code comparison enabled us to identify underlying and emerging uniformities in the meanings of the codes (i.e. the concepts or activities) and with this we produced categories. Other categories, and thus unanticipated codes, emerged during the coding process. These primarily concerned concepts and aspects of the process we had explicitly not addressed in the questionnaire, for example, 'delivery stories that describe the non-functional requirements and architecture', 'domain owners', or 'risk. . .'. An illustration of our coding is presented in Appendix B (Table 6). This example uses transcribed text from the project Beta, where the code corresponds to the italicized interview text. The full list of codes is given in Appendix D.

The emerging categories were analysed in relation to our research questions. The categories were clustered into headline themes (as described in Section 4), i.e. similar categories were aggregated into a headline theme. Each headline theme addressed one of our four research questions. The results of the data analysis are presented in Section 4. The discussion of these results is in Section 5. As is customary in qualitative case studies, when describing the categories, sample quotes are provided to vividly illustrate the points that the interview participants raised or shared (Charmaz, 2007).

4. Results

4.1. The roles of the clients and developers

RQ1: Who are the decision-makers in the prioritization process? What are their roles and what are they responsible for?

From the vendor's perspective, our findings indicate that people who are assigned one of five roles, namely *Business Analyst*, *Tech Lead*, *Domain Owner*, *Delivery Team Head* and *Test Scenario Team Lead*, collaboratively make requirements decisions at inter-iteration time. The decision to sign-off is made by the client's *Product Owner*. We note that, unlike the product owner's role described in agile books (e.g. Larman and Vodde, 2010), the product owner in our case study is a person at the client organization. The *Project Owner* is responsible for overseeing the project and for making sure that the right changes are communicated to the vendor in a timely manner (e.g. before the vendor's team starts implementing the requirements that would be subjected to changes). The product owner is advised regarding business decisions by *Subject Matter Experts* who are also representatives of the client's organization. These are senior managers responsible for various parts of the business. For example, in insurance, this would be setting up insurance policies, managing client complaints or defining long-term care benefits. Thus the five mentioned roles of the vendor's organization can be described further as: (i) *Business Analysts* who document the business process and data requirements in the form of user stories, (ii) *Tech Leads* who are functional managers of software development staff (e.g. design, testing), (iii) *Domain Owners*, who are responsible for accumulating knowledge concerning the client's business domain, sharing it among team members, and packaging it in a form that is reusable in future projects, (iv) *Delivery Team Heads* who are ultimately responsible for ensuring that their teams of domain experts deliver all the desired functionality along with the architects responsible for designing a system architecture that meets all the non-functional requirements, namely maintainability, changeability and usability, and (v) *Test Scenario Team Leads* who are responsible for ensuring technology dependencies and any constraints are considered as early as possible in the agile reprioritization process. Below are some examples of our interview data used to support and illustrate the above findings.

> "It is **product owners** [who are responsible for decisions on priorities]. Then [our] delivery story team, we have **business analysts**. And then **tech leads** are also involved in that to say 'this is the sequence and this is the way we can implement it based on the design perspective'. So these are the three people who are involved. And then in some of the teams we have the **scrum masters** also. They're also part of that" (Alpha, P3)

This statement by P3 (from project Alpha) illustrates the important roles played by the *Product Owners*, the *Business Analysts* and *Tech Leads*.

Furthermore, P1 in project Beta said the following about the roles in the project's prioritization process.

> "**Business analyst** will have interaction with the business customer, and later with the **IS analysts**, you go on with the developer who does the architect work also. Ok? So the developer and the tester are here, ok? Once it is all done, you go for

the business sign-off. So it has been this, where the business customers were on the client side. **Business analysts** were both... on-site and off-shore, and the **IS analysts** were from the client side. Ah... see, I'm just specifying this from the client side to say that they take the ownership, but we will be included in all the..." (Beta, P1)

Here, with the term *IS Analyst*, P1 is referring to the *Project Owners* described in the paragraph above. This highlights the importance of the *Business Analysts* and *Project Owners* in the development process. P2, from the same project Beta commented on roles and prioritization as follows:

"[...] and the task will be... like actually, the **scrum master** will come up with the high-level information about what he is about to do. [...]. Based on the developer input, analysis and analyst explanation, we will go for the high-level estimation, the tasks will be allocated and we will go with the task" (Beta, P2)

Here we need to equate the *Scrum Master* role described by P2 to the role played by the *Tech Lead*, as described in our findings above. The statement above expands on the crucial role played by the *Tech Lead* in prioritizing the requirements whereby prioritization is taken up by the *Delivery Team Heads* (developers), *Test Scenario Team Leads* (testers) and *Domain Owners* (managers).

We note that the Domain Owner and the Delivery Team (and the Delivery Team Head) roles were sensitized to the new concepts that were introduced due to the use of agile RE practices in the company. Regarding the Domain Owner role, our interviewees deemed that the complementary use of this role (in addition to the well-known roles of business analyst and product owner), increased the accuracy of development estimates and improved the collaboration between the development team and the client's product management. For example:

"There would have definitely instances of ah... development team now knowing specifics of that [...] in all the domain actually they have **domain owners** sitting here; SMEs or **domain owners**. So that issue didn't actually persist for long. So it's a... as and when we didn't know something we went and asked them (domain owner) and it was just on the spot we clarified that and they provided that knowledge." (Alpha, P4)

The above quote from participant P4 (project Alpha), describes how the client's *Domain Owner* role improves and facilitates the vendor team's interaction with the client. Development efficiency was also increased by eliminating efforts formerly spent on detailing requirements up front that were no longer relevant at the time of implementation. Furthermore, a designated delivery team comprising of senior architects considered the pragmatic approach to include non-functional requirements early in the agile RE cycles, trace those to concrete architecture design choices, and analyse the impact of changed requirements on the architecture.

4.2. Requirements prioritization criteria

RQ2: What prioritization criteria do large project teams use to make value driven decisions at inter-iteration time in an outsourced mode?

All our interviewees indicated that the concept of business value (which is native to agile, according to agile book authors, e.g. Eckstein, 2004) is more related to the realm of the client. According to them, the business value in a particular project is explained in the project's business case. The client's product owner makes sure that the business requirements that go into the user stories are indeed of value to their business. The vendor assumes that the client has profound knowledge of the business value used to represent their organization and that the client would communicate proactively any changes that could affect the business value of the product features being delivered (e.g. changes necessitated by revised or new legislation or regulatory rules specific to the insurance industry as a whole). The responsibility of the vendor is: (1) to capture the requirements in user stories and each one (according to our participants) should include scope, non-functional requirements, business rules, and acceptance criteria and (2) to transform the user stories (written, and updated by the vendor's business analysts on behalf of the client) in architecture design and in working code. This transformation is supported by the so-called *Delivery Stories*. These are translated user stories, into designs and test scenarios. A delivery story (i) can be estimated in terms of size of the functionality to be created, and also person/hours to 'deliver' it, and (ii) has an associated risk (from the perspective of the vendor's team), if it is subjected to later changes. We explain the concept of *Delivery Story* in more detail in Section 4.3.3. Hence, the risk in project Alpha was related to anything that endangered the project delivery, according to the fixed-price/fixed-schedule contractual agreement. (We will expand later in this section that risk is as important a prioritization criterion as business value). However, though less critical, projects Beta and Gamma, with flexible price/schedule agreements, had associated risk with repetitive project delays and additional costs.

All participants deemed the delivery stories as the pillars in the requirements reprioritization processes, because the calculated estimates were based on them. The delivery stories are created by a special delivery team responsible for the project. Software architects play an active role in this team, as they ensure that the architecture design choices that are made during requirements reprioritization are compatible with the earlier made choices.

Most interviewees in projects Alpha and Beta indicated that *Requirements Dependencies, Volatility, Risk, Effort, and Technical Debt* are the prioritization criteria used by the vendor's organization in the agile requirements reprioritization process. These criteria are also used to decide whether to escalate requirements issues to the client's product owner.

All interviewees in project Alpha put forward the concept of requirements dependency. For example P2 (project Alpha) said:

"[...] if at all, the **dependencies** have been identified very clearly, we can go ahead, otherwise we have to look into the **dependencies** first, and then decide, maybe not the entire series maybe we'll pick the initial, [...]. Those are, mainly, these re-prioritization is mainly triggered by the **changing requirements**, and not getting the signed-off requirements on time also. [...] For each **user story** I should know this user story **depends** on all these, ok? So, inside the design, inside the story we definitely have one explicit placeholder for marking the **dependencies**, but the same thing we want to track it in the backlog itself." (Alpha, P2)

Hence, P2 describes the prioritization difficulties faced due to the volatility (from the technical perspective, as we shall explain later), dependence between requirements and specifically due to dependent user stories.

P2 from project Beta had the following to say about requirements dependencies and prioritization:

"[...] Ah... but when, when we go for the long run, say, when we are trying to pick up our... pick up the task in backlog, obviously they'll pull out the **dependency items** or the **user stories** together. [...]. So.... yes, when we do the **re-prioritization**, or the planning, they will bring the **user stories** which are **dependent** on each other." *(Beta, P2)*

However, requirements dependencies played a lesser role in project Gamma in the prioritization of requirements. The reason for this was the design architecture for the Gamma project was

more modular and had fewer dependencies (and these were well known) compared to the architecture of the Alpha and Beta project (Table 2). This is expounded by the following statement by P0 (from project Gamma):

> "Yeah, they will have ah... from their perspective yes, definitely they will have to do that analysis. Here, more or less, the features are atomic, and the dependencies are very well known. So... it's... by definition only, they come out easily." *(Gamma, P0)*

The case study participants pointed out six types of *Requirements Dependencies*:

(i) *Inter-domain* dependencies that are concerned with requirements cross-cutting multiple business areas (e.g. client's policy set-up and client's complaint management).

As P5 from the Alpha project said:

> "And... and the **interactions** here are not silo-ed. It is actually **across domains**. Ah... what happened was; as we went through the user stories, we would actually come across... we would see that there were many **dependencies** between each of these, you know, **processes**, **entities**." *(Alpha, P5)*

(ii) *Intra-domain* dependencies which refer to sequencing requirements that build upon each other within a specific business process (for example, before closure of a client insurance file, certain activities must happen beforehand). For example, Interview Quote 9 (from (i) above) also described the intra-domain dependencies – dependencies between processes and entities.

(iii) Dependencies *due to downstream activities*, i.e. sequencing requirements in a way that maximizes the use of the available human resources (e.g. testers, designers, architects). For example, P1 from project Alpha had the following to say:

> "So... and sometimes we pick up stories parallelly because I cannot put the stories on hold, there is a lot of **downstream dependency**, so we work on parallelly, agreeing with domain owner." *(Alpha, P1)*

(iv) *Team-based* dependencies concerned with avoiding multiple teams having to work on the same or on dependent artefacts. For example, P1 from project Alpha indicated:

> "[...] And having more scrum teams also would not help because of **dependencies** and components... the application is so... it is a single mono-source component application, so that gives a lot of challenges in terms of having more resources also, because at times four people have to work on the same component. Which might be easy, but still... merging at the end of the sprint, it's a nightmare." *(Alpha, P1)*

(v) Dependencies *among user stories*; these are imposed by the order of activities in a specific business process (e.g. client's identity data needs to be entered before client's file is altered).

> "We went through the **user stories**, we would actually come across... we would see that there were many **dependencies** between each of these, you know, processes, entities." *(Alpha, P5)*

(vi) Dependencies *among delivery stories*; these are dependencies between non-functional requirements (e.g. usability, maintainability) and architecture choices; for example, the first quote in this section (Alpha, P2) refers to such a dependency.

Furthermore, *Volatility* is a feature of group of requirements which the client anticipates to change. Our interviewees indicated two types of changes were permissible in the project: (i) changes from a business perspective, which means changes because of new/updated legislation and regulation rules, organizational priorities (e.g. mergers and acquisitions happening in the client organization), new/updated business rules and (ii) changes from a technical perspective, which refers to the cases when the vendor's tech leads find delivery stories too expensive or too inflexible (e.g. severely limit the architecture) or when team/downstream dependencies cause changes necessitating reprioritization of the requirements, so that the project gets finished on time and within budget. For example, as shared by participant P0 in project Alpha:

> "We completed the delivery story we were supposed to start, and then we came to know that those stories were not correct anymore because of the changes to the legislations." *(Alpha, P0)*

This statement by P0 clearly illustrates volatility due to changes in the business perspective. This is in line with the first quote in this section (Alpha, P2), also illustrating a typical example of volatility from the technical perspective.

Next, *Risk* is considered by our interviewees as anything that questions the project's ability to deliver the system according to the planned deadline. Risk might refer to any changes that call for a severe redesign of the architecture, or any changes that impact some essential non-functional requirements, e.g. maintainability. Since the vendor will be responsible for maintaining the system later on, the vendor's team is committed to a high level of maintainability. Hence, they treat any delivery story that is not compliant with the objective of making a system highly maintainable, as a risk. Apart from risk, *Effort* has to be taken into account. It implies the estimated amount of person hours to complete a delivery story. The vendor's organization uses empirical data about the productivity of their teams to provide accurate estimates (if feasible) as an input to the requirements reprioritization process.

Lastly, *Technical Debt* implies the amount of architecture-redesign related work that accumulates over a period of time, due to the short-term perspective on architecture. Newly arriving requirements may be technically un-implementable because of limitations in the current architecture. The need to accommodate these requirements would mean having to invest a significant amount of effort to redesign the architecture. Technical debt refers to those situations where it is considered safe to start on the development without having to focus much on architecture design. If agile projects are in such a situation, technical debt becomes a prioritization criterion. Table 6 (Appendix B) presents an explanation of this concept by a participant in project Beta.

According to our interviewees, the concepts *Risk* and *Effort* seem to be closely related, from the vendor's perspective. Our case study participants emphasized the fact that risks as perceived by clients and risks as perceived by vendors are different. They pointed out that in a fixed price/fixed schedule project, additional effort (e.g. adding new project staff, billing) pertains solely to the vendor. But the vendor would then have to 'spend' (elsewhere) employable people at no extra cost. The vendor would try to minimize this, if an effort-intensive user story is deemed high-priority. They explained that if a project is not fixed-price, it becomes a risk for the client and there would be intense deliberations as to whether the effort being proposed by the vendor is really justified. At this point, according to our interview data, the importance of trust and 'relationship' between vendor and client comes into play. The interviewees agreed that these aspects could well be subjective to some extent. However, they thought that if we explicated the domain knowledge being used to make the effort estimates and to assess risk – based on a systematic process – it might be less subjective. This relation between *Risk* and *Effort* can be seen in the following quote by P3 from project Alpha.

> "Today morning we had one discussion where we were saying 'if you implement it somewhere else, it will save our effort, and it will be a right thing'. But the way the business team, they explain, they said no, this is a scenario. But then we found there are other gaps also. So, what we say, we will implement this,

Table 3
The results (of RQ1 and RQ2) of the different projects in our embedded case study.

Outsourcing project	Project Alpha	Project Beta	Project Gamma
Roles (RQ1)	The 5 mentioned roles	The 5 mentioned roles	2 Roles (Business Analyst and Tech Lead)
Importance of Prioritization Criteria (RQ2)			
Dependencies	High	High	Low
Volatility	High	Medium	Low
Risk	High	Medium	Low
Effort	High	Medium	Low
Technical debt	High	Medium	Low

or we can keep a reason for this, once a change request comes for this, we will implement this portion. So it helps us to put the right thing instead in future we would again do a rework. So you give me additional requirements and I'll plug it. Instead of I'll rework something now I will take out the things now, and in future I will implement your solution". *(Alpha, P3)*

4.3. Characteristics of the project's settings that influence the prioritization process

RQ3: What is the relationship between project settings and requirements prioritization?

We found that the requirements prioritization processes were consistent throughout project Alpha regarding: (i) the shared understanding of the client's business value as the key driver, (ii) the way risk was treated and the attention that was paid to it, (iii) the use of delivery stories for effort estimation purposes, (iv) the change impact analysis procedures being used, and (v) the communication and escalation procedures that were followed to interact with the client's product owner. The perceptions on how the reprioritization process was carried out varied from participant to participant. Different actions were taken by different participants as a response to effort estimation information. Each participant represented a specific role and the variations are traceable to these roles. This implies that though the interview data provided by business analysts did not diverge, the experiences of the business analysts, scrum master, portfolio manager, as well as that of the testing scenario lead varied. For example, the testing lead was more concerned with requirements dependencies in terms of downstream activities and escalated them on a regular basis, while the business analysts were mostly focused on inter-domain and intra-domain dependencies. On the other hand, our interview data revealed that in projects Beta and Gamma the answers to our RQ1, RQ2 and as a consequence RQ3 were different, as shown in Table 3. Looking at RQ1, although project Beta had the same number of roles, project Gamma had significantly less. Also, most of the criteria we mentioned as important for project Alpha were reduced or of no importance for projects Beta and Gamma. We think the reason behind this could be the different degrees of outsourcing (Table 4) in projects Beta and Gamma, as compared to the more traditional outsourcing scenario in project Alpha.

In what follows, we discuss some context factors held to be responsible for the consistency of the requirements reprioritization process (and the types of variations we observed), as suggested by our interview data. These are: maturity of the organization, the way domain knowledge was shared, and the use of delivery stories as an artefact in the agile requirements prioritization process.

Table 4
Types of outsourcing arrangements applied to the embedded case studies after Dibbern et al. (2004).

Degree of outsourcing	Ownership		
	Internal	Partial	External
Total	Project Gamma		Project Alpha
Selective		Project Beta	
None			

4.3.1. Maturity of the organization

Our data analysis indicates maturity of the organization is a vendor's quality that the interviewees deemed instrumental to the scaling up of agile practices, so that pieces of enterprise system functionality of high business value for the product owner can be delivered quickly and visibly. Being mature (e.g. CMM level 5) means that a vendor has a robust infrastructure for software process improvement. If such a vendor chooses the agile paradigm as their improvement strategy, the vendor can leverage the process improvement infrastructure and design working proposals on how to implement the agile principles and practices in their organizational environment, so that an agile process for system delivery is repeatable and predictably successful. An example of 'leveraging process improvement infrastructure' is the use of the enterprise modelling tool ARIS (Scheer, 2000) for mapping the business requirements onto the architecture design of the system to be delivered. (ARIS has been the leader in the tool market for the last 20 years of large enterprise systems projects). We note here, that our interviewees were not referring to organizations that just strive to obtain CMM level certifications (for the sake of being certified), but companies that create an environment in which teams and employees are genuinely committed to actual efforts in selecting relevant, applicable and feasible agile practices as well as 'customizing' these agile practices to suit different kinds of project execution scenarios (e.g. offshore or outsourced). We think it is realistic to assume that not all mature (e.g. CMM level 5) companies have this special focus. For this reason, although it is intuitive to believe that the maturity of CMM 5 organizations can help the performance of an agile project, we think the maturity level is not the only aspect determining the success of agile practices. Instead, we think the special focus on process excellence and the organization's urge for adopting and adapting agile values and practices influences the agile prioritization process, as described in this case study. In our case study site, all professionals were briefed on why and how agile approaches would be used in the organization and in the project. The plans for the use of agile approaches were elaborated in detail (by the vendor's program management office and process excellence team) and written down to make sure all processes were defined and followed. For the policy to be effective, it came with basic information and everyone working on the project had to know it.

4.3.2. Domain knowledge sharing and the complementary use of roles

Our data analysis indicated that domain knowledge sharing (which was broadly encouraged in the vendor's organization, see Oshri et al., 2007) seemed to be an important context characteristic affecting the way in which agile requirements prioritization was implemented. A feature of the agile RE process in our case studies was the complementary use of the *Domain Owner* role. This put an emphasis on the accumulation and sharing of knowledge and,

where needed, the transfer of domain knowledge from clients to developers. P1 from project Gamma had the following to say about the complementary use of the Domain Owner:

> "We have a kind of [specifically responsible for the domain knowledge in the team]... business analyst, they understand the domain. And then there are some set of people who work for... only for development, development activity. So there are business architects, technology architect... business architect works closely with the domain people. And then we have enterprise architect, they understand both." (*Gamma, P1*)

In our view, the fact that the vendor proactively considered the need for sharing knowledge (and in turn organized the agile RE process by paying a lot of attention to it) is not accidental, because domain knowledge sharing is related to the vendor's maturity and commitment to adopt/adapt agile practices (Oshri et al., 2007). As stated in the Agile Manifesto (Beck et al., 2001), the agile values and practices are primarily oriented to the team and project level and their premises rely heavily on the team of professionals immediately involved in a particular project effort. One can easily assume a highly mature vendor will want his/her processes propagated broadly and across different locations. A highly mature organization that has already established both the infrastructure and the mechanisms can be expected to preserve knowledge and information (and methodically apply approaches to making knowledge explicit across teams), and it is natural for it to maximize the benefits of the infrastructure and use knowledge sharing to strengthen the agile RE and project delivery.

4.3.3. Delivery stories as artefacts

General practice in agile methods is the specification of requirements as user stories. One novel concept was discovered with respect to the implementation of Agile in this setting, that of 'delivery stories'. This was introduced by the project Alpha team and was adopted later by project Beta as an internal best practice. Delivery stories are created by taking user stories and extending them with a functional specification, high level design and test scenarios. This provides the vendor with an assessment of what needs to be done to deliver the user story. As one participant remarked:

> "[...] it is not easy to comprehend from the business rules and directly develop. There is no proper flow sometimes, so, we had formulated delivery story idea." (*Alpha, P0*)

The Project Lead for project Alpha described delivery story as the following:

> "[...] delivery story is like a pre-development work for the Scrum" (*Alpha, P1*)

According to the experiences of our interviewees, the delivery story concept provides large projects with something more measurable and concrete than the concept of user stories. A number of participants from projects Alpha and Beta stressed the importance of delivery stories. However, though the participants in project Gamma did not explicitly use the delivery story concept, they did have an intermediate translation model between the user stories (the output of gathering requirements) and the final architectural design specification. A consultancy team translated the user stories into more concretely defined specifications for the developers. This represents an intermediate stage between the purely agile style and the formalized delivery process of Alpha and Gamma. With the help of the interviews as well as project documents made available to us, we have pieced together the steps that go into the 'user story – delivery story specification process'. Below we describe how delivery stories are created (according to the data collected from the interviews).

First, the vendor's business analysts acquire the user stories from the client. This user story is then assigned an owner, usually the responsible person on the client's side. Individual user stories are placed in context (based on common features) with each other in a process map, which is a way to explicate dependency and functional sequence. At the start of development iteration, the user stories are taken up from the project backlog and turned into delivery stories. Groups of user stories are associated with a generic 'feature'. The complexity of each feature is then estimated by means of a metric (high, medium, or low). The interviews indicate that this complexity metric is based on the experience and gut feeling of the business analyst, rather than on formal criteria. Participants did suggest they would adopt a more formal approach if one was available. Features are then mapped into sub-processes, which then evolve into the eventual delivery stories as they become further specified. For each of these sub-processes, the client defines business rules, together with the acceptance criteria for each business rule. The vendor's business analysts then take up these business rules and acceptance criteria, and translate them into functional specifications. At this point, the design team takes over and creates design specifications from the functional specifications, while the testing team uses them to create test scenarios. This process is summarized in Fig. 1.

It should be noted that all the different elements depicted in Fig. 1 are mapped onto one another through a traceability system that lets project team members navigate from the specific user story, to business rule and functional specifications. This enables project management to maintain a high-level overview.

At the end of this process, the following properties are associated with each delivery story:

- number of use cases
- required time for use case implementation in person days
- number of GUI Screens
- required time for GUI Screen implementation in person days
- business rules and validations
  - number of business rules
  - number of validations
  - number of field validations
- number of test scenarios
- required time for test case preparation in person days
- required time for data models in person days
- total time required for sub-process implementation

The interview data suggests that the delivery stories are created in close collaboration with the client's representatives. Once this process is over, requirements are concrete enough and they can be handed over to the actual development team while minimizing uncertainty. Our case study participants agreed, formalizing and concretizing user stories in this way helps bridge any gaps in understanding that may exist between clients and developers. While this process makes requirements more explicit, clarification requests can still be raised.

4.4. Balancing value creation for both the vendor and the client

RQ4: How does the vendor's team combine value creation for their own organization with value creation for their client?

Based on the interview data, we distilled ten practices which the case study participants considered 'to work well' in a large agile project, but there was a clear need to balance value for the vendor and value for the client. The participants perceived the vendor's and the client's perspectives as 'opposing sources' and indicated that they made a conscious effort towards 'combining value-creation for the vendor with value creation for the client'. Each interviewee was asked what he/she did personally with respect to this, hence we

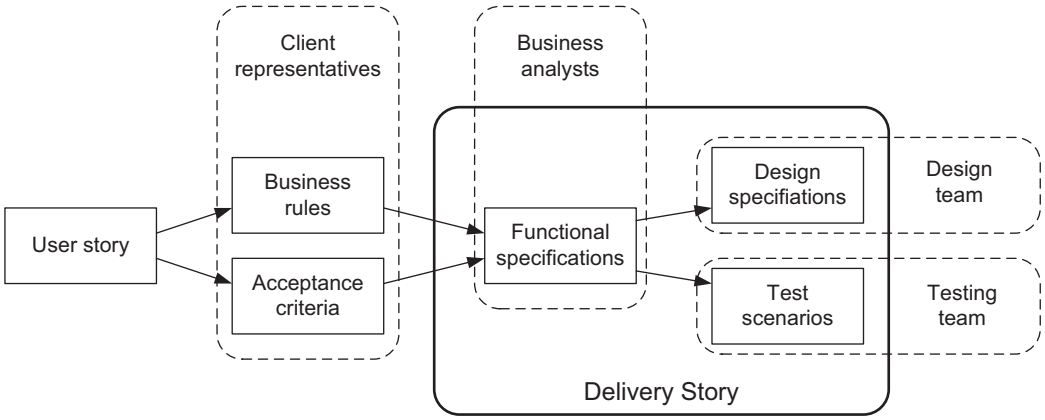

Fig. 1. The user story – delivery story translation process.

were able to distil clusters of actions belonging to an overarching theme. This in turn led us to formulate practices in an actionable format as follows:

1. Maintain traceability between user stories and delivery stories at all times. This allowed everyone on board to see how a high-level business process 'translates' into smaller chunks, namely user stories and the respective delivery stories.
2. Establish a dedicated Delivery Story Team. These are people directly responsible for the 'translation' of the functionality, as requested by the client for the architectural design.
3. Run series of workshops (user story workshops and delivery story workshops). This is the particular manner whereby the large vendor team created rapport and built trust with the clients. (The modus operandi of the workshops was: some initial face-to-face meetings followed by video-conferencing and teleconferencing, at regular intervals.)
4. Establish the role of Domain Owner. This role was deemed critical for acquiring knowledge about the core business processes and operational procedures specific to the insurance industry. The expected pay-off was that at the time of system maintenance, the domain owners would be instrumental in running the maintenance and operation processes effectively and efficiently.
5. Understand requirements dependencies **before** carrying out the design. This was critical to avoid the unnecessary implementation of complex functionality (only to subject it to changes a few days later).
6. Set up the Product Owner role at the client's site. Unlike the recommendations in agile books which refer to a product owner as someone on the vendor's site who 'represents' the client, in our case study project the client installed the Product Owner role.
7. Use the requirements change impact analysis process set up at the vendor's side to communicate with clients. The vendor's team had clearly defined policies about which requirements changes were permissible and which were not. They consistently used the change impact process whenever they found elevated risks associated with specific delivery stories.
8. Ask the Product Owner to confirm any changes directly and resist the temptation to second-guess the client.
9. Use status (every three weeks) to signal transparency. The case study participants deemed this to be instrumental in building a trustful interaction with the client.
10. Be prepared for the training on the domain to be gradual. In our case study, training activities were all supported by the client organization. For example, the client in Alpha prepared a 100-page book on how the core business processes work. Our interviewees deemed the training 'extensive'. In their view, the training meant investing resources (that were probably saved due to the adoption of agile practices!). They emphasized that the training was a worthwhile investment, as its returns would be realized in the next project with the same client organization.

We noted that the practices were derived directly from the interview responses to 14 Questions in the interview guide (Appendix C). We refer interested readers to Appendix C, questions Q.1-5–Q.1-9, Q.2-2, Q.2-7, Q.2-10, Q.2-11, Q.3-6–Q.3-8, Q.4-4 and Q.4-5). We also noted that not all practices were mentioned by all participants. However, because we wanted to describe the diversity of experiences of what made agile work, we included all the themes mentioned without counting the frequency of their occurrence.

5. Discussion

Our exploratory case studies were embedded in a CMM level 5 organization and covered different types of outsourcing engagements (see Table 2). Dibbern et al. (2004, p. 12), in their extensive review of outsourcing literature, provide a typology of outsourcing arrangements. In the typology they mention the degree of outsourcing (total, selective and none) as well as the amount of ownership (internal, partial and external). Furthermore, Dibbern et al. (2004) define total outsourcing as the transfer of IS assets, leases, staff, and management responsibility for the delivery of IS products and services from an internal IS function to a single third party vendor which represents more than 80% of the IS budget. Selective outsourcing refers to "the decision to source selected IS functions from external provider(s) while still providing between 20% and 80% of the IS budget internally". "None" implies keeping the management and provisioning of more than 80% of the IS budget internally.

As shown in Table 4, our embedded case studies cover the spectrum of outsourced project ownership. This gives us the unique position to also determine the impact of the different types of outsourcing arrangements on agile requirement prioritization. We observed that certain roles as well as prioritization criteria were given more importance as we moved from a typical Outsourcing scenario (project Alpha) to a more internal sourcing scenario (project Gamma) (Table 2). We also noticed that certain practices like the use of Delivery Stories that we observed in projects Alpha and to a lesser extent in project Beta were not so important in project Gamma. This was of course also due to the size of Gamma (Table 2).

The case studies yielded a few findings regarding the essential aspects of requirements (re)prioritization in large agile projects, which deviated from what agile literature says about these aspects.

Overall, these findings revised parts of our understanding of the following:

a) the roles on the client's and vendor's sides, such as the significance and utility of the Domain Owner role;
b) artefacts, like the Delivery Story, that helped in understanding and prioritizing the requirements;
c) the prioritization criteria used, and the implementation of agile practices were different based on the contexts of the embedded case study projects.

We conjecture that the introduction of both specific roles (e.g. Domain Owner) and specific artefacts (documentation) were due to (1) the high maturity of the organization the case studies were embedded in and (2) the organization's effort to leverage domain knowledge. We will now discuss some findings in more detail and will add some general observations.

First, an important role was played by the Program Management Office (PMO) and the Process Excellence (PE) team in the vendor's organization. Their involvement in scrutinizing scenarios of how to implement agile practices in a project was critical. In each of our different embedded case studies, we noticed the PMO and the PE teams let the project team try out a proposal for agile practice implementation and learn from it, and only then decide (based on reflections from learning) whether to continue using the practice as implemented, or to modify it in the next iteration. This led to lessons emerging as the project progressed, and also ensured that the new lessons were gradually incorporated into the company's ways of getting things done.

Second, this set up, allowing lessons learned to be incorporated, led to a new concept that merits further attention: the *Delivery Story*. According to our interviewees, although user stories describe system functionalities in terms of the client's business, they do not really describe the specific implementation details. In a small project, this is not problematic, as one gets to understand the design specifications, dependencies and functional requirements quite quickly in the short development iteration. However, in a large project, involving a lot of functionality, the requirements need to be more detailed and specific on the implementation details in order to facilitate better effort estimation. Delivery stories capture the same functionality as User Stories, yet they also add those details that are needed to plan their implementation. These details include *functional requirements*, *design specifications* (where appropriate), *security requirements* (if applicable). Delivery stories are created from user stories by turning them into functional specifications, high-level designs and test scenarios; see Section 4.3.3. Delivery stories do play an important role in (re)prioritization, because they have an associated effort and risk. Also, fleshing out the design details helps in understanding the requirements better. Hence, the information in the delivery stories helps in making sure that the business requirements are technically implementable. Lastly, delivery stories are a vehicle of communication between the vendor's teams and the client.

Third, vendor's value, volatility and effort are deemed key prioritization criteria, besides client's value. The concept of requirement's dependencies plays a critical role in balancing the vendor's and the client's value generation when making a decision on requirements priorities. Focusing on dependencies as early as possible in the project was deemed beneficial, because of its potential to save rework and redesign. Our interviewees pointed out that if uncovered late, complex inter-dependencies on other system components and environments could hamper productivity, as a small change in one system can lead to a cascade of rework in numerous different places. However, requirement dependencies only play a significant role if the architectural design is not very modular and/or if the dependencies are not universally well known to the development team members. As seen in project Gamma (Table 3), dependencies did not play such a significant role in managing risk as the project was very modular (Table 2) and the few dependencies were common knowledge to the team members.

Fourth, an interesting observation in our study is that agile RE was not treated as a tool-based approach that should be rolled-out across the project organization. Rather, it was considered a client's value-based approach. As a matter of fact, according to our interviewees, no special tool was used for agile RE. The vendor's team members were, however, well aware of the business needs of their company to make agile philosophy work in large projects and were highly committed to making it succeed. This observation converges with the observation of McDowell and Dourambeis (2007) at British Telecom, whereby in order to be successfully agile in large projects, "team members have to want to do it".

Fifth, a unique context factor in our case study organization was the determination to leverage domain knowledge. Domain knowledge is a reusable asset in follow-up projects (Oshri et al., 2007). In fact, delivery stories are a way to document knowledge. The reasoning being: in a small development team, such knowledge can be pooled between the team members; in a larger setting, this is impossible, and the knowledge flow needs to be organized. The essence of delivery stories is that requirements are immediately linked with essential pieces of implementation knowledge in a single artefact. Otherwise, these may get stored and known in different places. This information comprises functional specifications, related business rules, design specifications and test scenarios. The successful application in the case study projects shows the feasibility of this approach. We also observed that the commitment to leverage domain knowledge brought up the role of Domain Owner. This role is an expensive investment, as it requires extensive training. However, it is expected to pay off in any follow-up projects with the same client. We note that, indeed, the vendor's organization has taken up several initiatives to build reusable domain knowledge assets based on the lessons learned by delivery teams and domain owners when implementing large projects. There is evidence from other fields that the pay-offs and the value of the domain owner's roles are fully justified. For example in the field of RE for enterprise systems (e.g. ERP) similar roles have been documented. Earlier publications (Daneva, 2003) revealed that if RE is to succeed in enterprise system projects, the team must have three types of expertise: (1) in common business processes in the client's business sector, (2) in specific business processes of the client organization, and (3) in requirements models which define how the client's business processes will be supported by the system. In our case study project, the role of Domain Owner was supported by means of tool-and-standards-based processes. For example, process modelling was accomplished by means of the ARIS toolset (Scheer, 2000) (a leading product in the market of tools supporting model-driven enterprise system implementations). User stories and delivery stories were linked to the activities and events in the ARIS event-driven process chains (Scheer, 2000). In turn, this not only ensured traceability between requirements documents at different levels of detail (e.g. business processes in ARIS, user stories, and delivery stories), but also accumulated, packaged, and shared business process knowledge. The role of the Domain Owner and the environment settings that help it succeed and make an impact on the project, are under-researched in the agile RE literature. We think that one can expect agile projects to benefit in certain ways if experienced domain owners are on board. However, further research is needed to collect and analyse evidence to confirm or reject this hypothesis. The vendor's organization has taken up several initiatives to build reusable domain knowledge assets based on the lessons learned by delivery teams and domain owners while implementing large projects.

6. Evaluation of the validity threaths to the results

We evaluated the possible threats to the validity of the observations and conclusions in a case study (Yin, 2008). For this purpose, we used the checklist for case study researchers as recommended by Runeson and Höst (2009). When the exploratory qualitative research was planned and executed, the key question addressed when evaluating the validity of its results was (Yin, 2008): to what extent can the practitioners' experiences and the observed requirements prioritization mechanisms, be considered representative of a broader range of companies?

As mentioned in Section 3.2, our projects were not completely agile in nature. They incorporated a mix of agile and structured approaches to developing software because the vendor organization was CMM-5 certified and hence needed to follow structured practices (Banerjee et al., 2011). However, we think that the conclusions we draw will hold for other companies with similar contexts to our case study organization (company size, project size, presence of process excellence units, maturity level and fixed-price contractual agreement). In higher maturity companies, it is intuitive to assume there are teams (similar to our case study organization's Program Management and Process Excellence Team) who scrutinize the application of agile practices and assess the fit to the established systems delivery framework. We also think that there are many companies (especially in Europe and North America) which may not be CMM-certified, and yet may have established an effectively functioning project management office. The formation of these teams is, by and large, market-driven as (1) many clients increasingly prefer vendors to use agile practices and (2) vendors are expected to possess and demonstrate competence in "agile". While clients are well-versed in agile practices and the benefits thereof, they might not necessarily be involved first-hand in adapting the original Agile Manifesto (Beck et al., 2001) guidelines to large distributed (and possibly multicultural) project settings. More often than not, they expect the vendors to take care of these issues, while insisting the system should be delivered using agile practices. In light of this market development, we assume organizations which have already formed internal program management/process excellence units are most likely to find our conclusions relevant and useful.

We also acknowledge the inherent weakness of the interview techniques, as they were driven by the researcher, meaning there is always a residual threat to the accuracy of what interviewees say. An interviewee may have not understood a question, or was not honest in his/her answer. However, we think that in our study, this threat was reduced, because the interviewer (van der Veen) used follow-up questions, and asked questions about the same topic in a number of different ways. Also, we accounted for the possibility that the researcher might instil his/her bias in the data collection process. Moreover, each interview conversation was completely transcribed and every transcript was available for reference purposes, which reduced the threat of misunderstanding the data on the researchers' sides.

A further validity concern in interview-based studies is that the researcher influences the interviewee. To counter this threat, special attention was paid to conducting the interview in such a way that the interviewee felt comfortable when answering the question and did not avoid the question. This created a safer environment and increased the chances of getting a truthful answer rather than one aimed to please the interviewer.

To minimize the potential bias of the researcher, we also considered the construct validity of our study. We followed Yin's (2008) recommendations by establishing a chain of evidence. First, we included participants with diverse roles in the same project, and this allowed the same phenomenon to be evaluated from diverse perspectives (data triangulation, Patton, 1999). Second, the draft case study report was reviewed by researchers from the company which hosted our case study. These researchers read and re-read multiple versions of the case study results.

The choice of the projects in the study could represent a threat to the validity of the results as the authors relied on their professional and personal network to establish contacts with the project members. Although the projects were suitable for the study, they are not representative of all the possible ways in which prioritization is performed in large agile organizations. We, however, consider that our findings could be observed in companies and projects with similar contexts to those in our study, i.e. mature Asian companies engaged in outsourcing, with large distributed projects, and embracing agile as their systems delivery model. Seddon and Scheepers (2011) suggested "if the forces within an organization that drove observed behaviour are likely to exist in other organizations, it is likely that those other organizations, too, will exhibit similar behaviour" (p. 12). We acknowledge further research is needed to produce evidence to evaluate the extent to which our findings are observable in other organizations.

7. Implications for practice and research

This section presents our reflection on the empirical study regarding its implications for practicing agile SE professionals and for researchers engaged in empirical studies in agile SE and RE.

7.1. Practical implications

Our results have the following implications for agile practitioners: First, a key implication for agile project managers is the need to match the mode of project scope negotiation between the outsourcing partners, to the contextual needs of the project. We found that simply providing product owners and business analysts with the processes to develop user stories was not enough. Managers should match delivery stories with user stories. We found this allowed the case study organizations to strike a balance between the scope control efforts, the disciplined management of requirements interdependencies, and the changes in the environment faced by the development team and the client.

Second, delivery stories as artefacts can play a major role in large-scale agile software engineering. This role is threefold: delivery stories serve as (i) synchronization artefacts (that is, between the business analysts facing the client's representatives and the development team), (ii) an approach to manage scale, (iii) a control mechanism – it helps to tighten the scope as much as is feasible, thus mitigating the effect of changing requirements. In the role of a control mechanism, the delivery stories restrict the set of available solution options. As a result, this reduces the choices the vendor needs to make, which leads to fewer requirements for on-going feedback. Because of their multi-faceted role, delivery stories should be described more precisely and find their way into handbooks and textbooks.

Third, agile in a large, distributed and outsourcing context requires a carefully managed approach. Our findings agree with the results of Sarker and Sarker (2009) who found that agile methods need suitable adaptation for effective use in distributed software development projects. A managed approach means using controls as an important tool to respond to changing requirements. Organizational maturity provides advantages in effectively instilling control, while absorbing the advantage of the agile practices in dealing with the client (for example, using short iteration cycles, delivering business value and working functionality in each iteration, writing user stories instead of large upfront business requirements specifications). This is in line with Cohen et al. (2004) who acknowledge the goal of agile project management is "to

communicate effectively and documentation should be the last option" (p. 33).

Fourth, our study suggests the role of domain knowledge sharing might be more important to the success of agile practices than that suggested by agile literature. While the role of domain owner does not come cheaply, it is expected to pay off in the next large project for the same client organization or for other client companies in the same business sector. However, this raises an issue of scalability. How to manage and share an accumulating body of knowledge? Some kind of knowledge repository could be used, and the manner of approach could be an interesting topic for further research.

We consider these four implications important for practice as we assume more and more large organizations will embrace agile in the future. This assumption is grounded on a recent study (Ambler and Lines, 2012) indicating that 88% of companies with over 10,000 employees are evaluating or using agile practices on their projects. Assuming many of these companies would work with large vendors, we think that understanding the vendors' realities and the mechanics of agile RE, would help adopters gain insights and form realistic expectations based on their trust in the vendor's ability to leverage agile, while being in a mature CMM-compliant setting.

7.2. Implications for future research

Our study has several implications for future research in empirical RE and SE:

First, our study yields a rich analysis that explicates how agile RE works on the part of the vendor. Unlike other empirical reports which concentrated on the challenges, we present workable solutions for the vendor. We also explicate why these solutions worked. Our analysis went deeply into 'the salient attributes of phenomena' (as Gregor (2006) calls it). This is of particular benefit to other researchers in cases where very little knowledge is available about the phenomena (Gregor, 2006, p. 623). In line with Gregor's reasoning, despite the fact that we do not and cannot provide a definitive description of how vendors cope with agile RE in distributed and outsourcing contexts, we made an important first step towards making knowledge in this area explicit. This knowledge could be helpful in the future to work out which replication studies are desirable as well as to discover new and relevant research questions. We elaborate on this in Section 8 where we suggest lines for future research.

Second, when comparing the results of this research with those from empirical studies (Racheva et al., 2010) that focused on small and medium-size projects, we found that (1) large projects are much more concerned about balancing 'the opposed forces' of the client's and vendor's value and (2) small projects do not have any detailed requirements (architecturally significant requirements that are like the delivery stories in our case study), nor a dedicated team of delivery staff members who are employed full-time on a project. Both findings (1) and (2) confirm the observation of Ambler and Lines (2012) who posit that balancing client's and vendor's value generation ensures the congruence of the delivery process activities with the goals of the vendor and of the project. We identified 10 characteristic practices of large projects (Section 4.4). Although we do not claim these are the only practices that will work, we do consider them as a good starting point to base future research on the effectiveness and the efficiency of these practices and how they impact important project outcomes in contract-based software development.

Third, our study confirmed the intuitive assumption that large scale agile RE demands new roles (like Domain Owners). However there is no reason to assume that all roles are suitable in all organizations. Uncovering which role fits better in what context could be potentially helpful when implementing agile, with more predictable outcomes and higher chances of success.

8. Conclusions and future research

This paper presents an empirical study in which a number of agile RE practices are introduced in the outsourced project development department of a large organization, in the context of real-life projects in different outsourcing arrangements. The contribution is a rich analysis of the studied cases, showing how the vendor implemented the agile principles and then adjusted the practices in order to fit the context of large projects. The key findings of our study are the following:

- The software vendor organization employed an original type of artefact, so-called *Delivery Stories*. These complement user stories with architectural design implications, test scenarios, effort estimation, and associated risk, which makes them pivotal objects for (re)prioritization. Using delivery stories emerged as a feasible way to lift agile practices to larger project contexts. To the best of our knowledge, delivery stories have not been documented before in the SE literature.
- The software vendor organization in which the case studies were conducted puts a lot of emphasis on domain knowledge. Knowledge transfer from the client is sought proactively, and the delivery model has been extended with the role of *Domain Owner*. This adds to the vendor's costs for projects in new domains, but it is believed that in the end it is a profitable investment as it captures business knowledge as well as architecture patterns of a domain.

There is no evidence from the data about how the application domain (e.g. the insurance application domain which is highly regulated) influences the prioritization process. We assume that a deep understanding of the domain helps to build trust with the client and to improve the quality of prioritization decisions, ultimately leading to better service and a longer engagement with the client.

Other findings about how prioritization takes place include: (i) depending on the modularity and knowledge of the architecture, an understanding of the requirement's dependencies is of paramount importance for the successful deployment of agile approaches in large projects and (ii) other than business value, the most important prioritization criterion in the setting of outsourced large agile projects is risk.

The cross (embedded) case comparison (Table 4) also demonstrates the impact of the outsourcing arrangement on the importance of certain roles, the criteria for prioritization (Table 3) as well as the importance of the use of artefacts such as delivery stories in the requirement prioritization process.

Reflecting on our experiences in this case study, we suggest the following lines for future research:

First, we consider it important to carry out a replication of the study in other organizations that have large, distributed and outsourcing projects. For example, we consider it worthwhile approaching other project organizations in different business units within the same company.

Second, we find it interesting to investigate the relationship between agile in the large and organization maturity (Cohen et al., 2004). Treating agile as a set of values, we have no reason to assume that a very mature organization would incorporate a new set of values from one day to the next. However, we found it can set up processes that make the interaction with the client almost perfectly agile. Explicating the organizational mechanisms responsible for this to happen will advance our knowledge and could potentially

help other organizations instantiate agile processes with more confidence and predictable outcomes.

Third, we acknowledge that the type of contract has a large impact: if it is fixed-price, there is a natural tendency on the vendor's side to reduce project-specific risk. We also acknowledge that the theme of risk which crystallized in our analysis, delves into how requirements prioritization took place, and relates closely to all those concepts that describe the balancing process of generating a client's and vendors' business value simultaneously. However, little is known about the types of project-specific risks on the vendor's side in agile at the large, in which way these risks are significant based on the pricing arrangement in the contract, what importance is associated with each of them, and what remedies exist to cope with them. We therefore think it is interesting to investigate risk as part of agile RE in large, distributed and outsourcing contexts, by conducting further case study research. This could include, for example, the use of existing risk analysis frameworks (Hossain et al., 2009) as the lenses to guide the collection and analysis of case study data in future research.

Acknowledgements

We thank the interviewees for participating in our research. We thank Ms. Zornitza Bakalova for serving as a reviewer of the questionnaire for our interviews. We also gratefully acknowledge the constructive and actionable feedback of the editors and the reviewers.

Appendix A.

Table 5.

Appendix B.

Table 6.

Appendix C. Interview Guide

"*First, let me introduce myself; my name is Egbert van der Veen, and I am a Masters student of the University of Twente, the Netherlands. As part of my graduation project I'm conducting research at [the vendor's] Research Centre. There, we are trying to better understand agile requirements prioritization from the client's perspective. This interview is part of a research project towards this goal.*"

<Offer consent form>

"*The questionnaire consists of some factual information up-front, followed by a semi-structured part. The whole interview should take no longer than 60 minutes. Please rest assured that all findings will be thoroughly anonymized, and no personal or confidential information will be used in any way. If you feel uncomfortable answering any of the questions, you can always choose not to answer. If you have any questions at any time, please feel free to ask.*"

Introduction

This case study represents an effort in discovering the way in which the agile requirements prioritization process produces value to the clients/product owners. We will study how requirements prioritization and decision-making on priorities happens, what the factors are that play a role in it, and how domain knowledge can support the process. The long-term goal is to improve the agile requirements prioritization process so that both clients and developers organizations profit: by ensuring a better understanding for the rationale behind the decisions and including the consideration of value-creation throughout the project.

In this study under re-prioritization we understand a re-arrangement of the initially prioritized Project Backlog (PB) during the project. This can happen after some of the following occurs: adding a new item on the PB; removing an item; changing the priority of an item; change in the scope of the project, change in the external conditions, new information during the project that require a change in the scope/schedule, etc.

"*Let's discuss a concrete project where you had to (re)prioritize requirements on multiple occasions. Do you have any questions at this stage?*"

"*We will first discuss some general characteristics of the project*"

Main characteristics and context of the project

- How long was the project?
- What was the type of contract? (e.g. fixed-price, per hour, etc.?)
- How many iterations/sprints did you perform?
- What was the nature of the distribution of the project?
  - How many sites were involved?
  - What was the distribution of people among these sites?
  - How was the work distributed between them?
  - How was the collaboration organized?
  - Which, if any, Agile distribution practices were implemented?
- Who were the different parties involved?
  - What were the different layers of corporate hierarchy that were involved?
  - Did the different parties represent different business units within the same organization?
- How did the communication with the client take place?
  - Modality
  - Frequency
  - Level of communication w.r.t. corporate hierarchy
  - Were there domain experts involved?
- What type of product was developed?
- How critical was the application?
- What was the size of the developing team?
  - How many other people were involved?
- What development method was used?
- What was the experience in all organizations with the methodologies that were applied? (i.e. project methodology (agile development))
  - Was there an existing preferred methodology?
- In what form were the requirements captured? (e.g. story cards, use cases, traditional requirements documents, etc.)
  - Was a specific method or template used?
  - Were the requirements written by the client, or by a representative of the developer's organization?
  - Is the business value of each requirement made explicit?

The following sections of the interview will proceed in a semi-structured fashion. The questions should be seen more as means to get a conversation started, and the line of questioning should not be considered to be too rigid. First we will talk about Section 1.

Section 1. Prioritization and re-prioritization process. (concrete case)

Q1-1. How often did re-Prioritization happen?

- Between each iteration/sprint?
- Triggered by outside change?

Q1-2. Can you remember concrete cases of (re)prioritization in the project?

Q1-3. Do you know what triggered them? How did you proceed in these cases?

Q1-4. Which factors played a role during the decision making?

Table 5
Case study participants.

Project	Participant	Role	Years of experience	Mode
Alpha	0	Business analyst	5	Face-to-face
Alpha	1	Delivery head	15	Face-to-face
Alpha	2	Portfolio manager	10	Face-to-face
Alpha	3	Scrum master	10	Face-to-face
Alpha	4	Business analyst lead	10	Face-to-face
Alpha	5	Business analyst lead (2×)	3, 10	Face-to-face
Alpha	6	Test scenario team lead	10	Face-to-face
Beta	0	Business analyst	2	Videocon
Beta	1	Project lead	10	Face-to-face
Beta	2	Tech lead	10	Face-to-face
Beta	3	Project lead	15	Face-to-face
Beta	4	Project manager	15	Videocon
Beta	5	Lead developer	10	Videocon
Gamma	0	Project lead	20	Face-to-face
Gamma	1	Architect	15	Face-to-face
Gamma	2	Consultancy lead	15	Face-to-face

Q1-5. Are there any reprioritization decisions that were hard to make, or took a lot of discussion?

- Why were they difficult?
- What were the pros and cons?
- What tipped the balance?
- Who, in the end, made the decisions? (client or vendor?)
- Does it always work like that?
- Was this an exceptional case; does it usually works differently?

Q1-6. Were there any issues arising specifically from the distributed nature of the project? Such as Language barrier, time zone difference, culture clash?

Q1-7. Are there any reprioritization decisions that, in retrospect, you would have made differently?

Q1-8. And some that you feel particularly good about? (i.e. where increased insight or serendipity made you change the direction of the project for the better?)

Q1-9. Where did changing requirements initiate from? Were they initiated by the client or by the developers?

"*In Section 2 will talk about the Agile requirements process in general, so not in any specific, concrete case*"

Section 2. Prioritization process (general observations)

Q2-1: Who is responsible for the decisions on priorities within one project and for every iteration?

Q2-2: Are any processes used to help prioritize and select requirements for an iteration? For example, the Planning Game?

Q2-3: Are any software tools used to help prioritize and select requirements for an iteration?

Q2-4: Which methodologies/techniques are informally used to select and prioritize requirements, if any?

Q2-5: Do you use explicit criteria for the prioritization? If yes, which? Do these criteria change from project to project or in different products? If yes, why? For example; if in one product the client is more important than in another, or if one project is more beneficial to your organization than other.

Q2-6: Do you use any formal way to calculate the worth of a requirement, e.g. cost-benefit analysis, a value estimation formula, an effort estimation technique?

Q2-7: Are you happy with the way requirements are currently prioritized at iteration time? What do you like in the existing approach?

Q2-8: Do you have cases when a client imposes his preferred way of prioritizing the requirements and then you adopted this in the prioritization process for the project? And this let the team change the way you prioritize, which might be different from what you typically do in other projects? If so, which parts of the approach do you adapt? Why?

Q2-9: Do you prioritize all the necessary work like refactoring, testing, etc., together with the client's requirements? Do you need to explain to the client why this is necessary?

Q2-10: Do you consider dependencies between the requirements? If yes, how do you handle them at prioritization time? How did you keep track of these dependencies?

"*Now we'll talk about some of your own past experiences for a moment*. . .

Q2-12: In the past, have you experienced cases where. . .

- 2-12.1 . . . you had to educate the client's representatives why prioritization is needed?
- 2-12.2 . . . the decision-makers did not have the necessary information to perform prioritization?
- 2-12.3 . . . the decision-makers did not know why and/or how to prioritize? If yes, please explain why, in your opinion, this happened.
- 2-12.4 . . . the stakeholders were not satisfied with the priorities, and this had a negative effect on clients or your organization?
- 2-12.5 . . . where the client requested changes in the backlog, do you know why did it happen?
- 2-12.6 . . . the prioritization decision had impacted the system in a significant way (significant refactoring or rework needed).
- 2-12.7 . . . the clients representative did not always represent the desires of all the clients stakeholders?
- 2-12.8 . . . the client's representative was not authorized to make crucial decisions, stalling/delaying the project?

"*Next is the topic of value creation, specifically*"

Section 3. Value creation

Q3-1. What does business value of a requirement mean for you?

Q3-2: At your meetings with your clients/product owners, do they explicitly discuss the business value of the requirements, so that everyone at the meeting understands why some requirements are of higher priority than others?

Q3-3: Is value connected to the business goals which the clients want to achieve by deploying the software system? If so, in which way does value connect to client's business goals?

Q3-4: When judging the value of the requirements, do clients also consider any other factors (e.g. cost, size, risk)?

Q3-5: Has the desired value been quantified? If yes, how?

Q3-6: In which way, in your experience, does the agile process add value to the client? Can you give a specific example from your practice?

Table 6
An example of coding with explanation.

Interview text	Code	Explanation
Interviewer: 'Ok. So, re-prioritization, is that ever triggered by outside change, or. . .? What triggers re-prioritization, generally? Participant P5: 'Ah. . . most important thing is. . . ah. . . what I have seen from my experience here is, one is the *management decisions* and the other one is the *end-user requirements*. The. . . quite a lot of mechanisms built into agile which define how prioritization takes place. For example, we have these show and tell sessions, which we try and give to the business in any particular iteration as and when we complete certain things. Then the *show and tell might actually trigger a thought process among the end users, probably they might think that instead of a combo-box they might as well go for something else, because after the show and tell they do feel that it is easier to use the keyboard rather than to use the mouse, so. . . so, that is something that triggers, that's from the end-users, from the aspect of how they're going to use it.* The next one is the management decisions. *A couple of management decisions re-prioritizing at the program level might trigger a change in what objectives you have for that iteration. So, I've. . . I've seen both of them.'*	Types of changes Types of changes Client-triggered-change Management-decision-type-of-change	Participant P5 discussed how reprioritization happened in his experiences in project Beta. Participant P5 explains a particular mechanism of how reprioritization is triggered, namely by means of 'show-and-tell sessions'. Participant P5 explains another mechanism of how reprioritization is triggered.
Interviewer: 'Ok. So can you remember concrete cases of re-prioritization in the project that you can maybe elaborate a little bit on?' Participant P5: 'Ah. . . Ok, something from the end-user. . . couple of simple changes like what message you need to get, couple of warning message that pops out of your application that was ah. . . agreed at the beginning of the iteration. But as and when we were carrying out and we were carrying out and we were showing the show-and-tell and all that, probably they wanted to change in that phase, because they felt that a different message would mean more understanding for the end-user. So those kind of simple changes, that's from the user, and probably a couple of things like how to fit that information in a grid, a couple of more options that they can use based on the estimates, so. . . it really depends on how we take in the changes. . . *the driving force is basically the estimate, and the impact of that change on the iteration objectives. So based on these two, we take a call and say that "if you do this, you have to risk re-testing everything. You have to put an effort and re-test everything, regression and everything. So based on a combined decision, we make sure that only those changes that are really required are taken in.'* Interviewer: 'Ok. So ah. . . yeah, you've already mentioned some of this, I guess. So which factors played a role in the decision-making? When deciding on priorities?'	Prioritization criteria Effort-as-Prioritization-Criterion Risk-as-Prioritization-Criterion	
Participant P5: 'Ah. . . well it's basically we know *it's cost and time that it takes, because time to market is always very important. So time to market. . . the other thing we try to accommodate is what we call as the technical debt.* *We term it as technical debt. In a typical agile project what happens is; you start developing something and you deliver something at the end of the first iteration which typically is a short period of time, its one month or something. So when you're trying to do that, obviously your design concepts have a short-term focus. You only design for the immediate requirement. So you keep doing that over a period of year. You. . . come out with something called technical debt. Design focuses only on the short-term focus; probably you wouldn't have planned for 500 users when you have to deliver something for the first month. Technically, no? So that keeps building up. So as and when it keeps building up, after the end of one year, whenever you try to invite any changes, to make the change, it becomes a bit complex, because your design is not so good, it is not an open design that you plan. . .. you've not foreseen a lot of things, so. . . the cost to incorporate a change over a period of time becomes high. So you also have to plan for a technical debt cost, which means you have to spend an iteration trying to remodel things and make it in the right way.* The design has to look [perfect?], so. . . So that also plays a part.	Effort-as-Prioritization-Criterion Technical-Debt-as-Prioritization-Criterion	Participant P5 elaborates on effort related to cost and time. Participant P5 explains what the term 'technical debt' and how it is used in reprioritizing requirements.

Q3-7: You, as part of the developing side do you consider the value for your own organization, or is what the client wants more important?

Q3-8. Do you share knowledge about business value creation within the organization?

"*Back to a concrete scenario. This section revolves around the question; have there been situations where a better understanding of the client's business domain would have helped requirements prioritization?*"

Section 4: Domain knowledge assistance

Q4-1: Were you personally familiar with the client's business domain at the start of the project?

Q4-2: Did your company have previous experience with this domain?

Q4-3: Were there recurring issues that originated from unknown aspects of the clients business domain?

Q4-4: In your opinion; have there been situations where a better understanding of the client's business domain would have improved the requirements prioritization process?

– Customers business goals made explicit

Q4-5: Was there a conscious effort, at any phase of the project, to learn more about the client's business domain? If so:

– At what stage of the project did this effort take place?
– Was there any methodology or tool used?
– Who was responsible?
– How was the customer involved?

Q4-6: Were the customers policies formalized in a structured way and made available to you, or were they hidden in the processes, from which they had to be reverse engineered?

Conclude interview; thank participant for his/her time, ask if there are any questions/remarks.

Appendix D. List of codes

All unique codes applied to the interview texts in the data analysis phase. Codes are nested, represented as 'superconcept >subconcept'.

1. business value
2. business value >business goals
3. business value >client vs developer
4. business value >decision hierarchy
5. business value >definition
6. business value >discussion
7. context
8. context >aims
9. context >aims >automation
10. context >aims >integration
11. context >aims >reusability
12. context >background
13. context >contract
14. context >contract >fixed price
15. context >contract >fixed price >consequence >fine
16. context >contract >fixed schedule
17. context >contract >per hour
18. context >criticality
19. context >issues >existing product
20. context >organizational structure
21. context >scope
22. distribution
23. distribution >challenges
24. distribution >challenges >solutions
25. distribution >communication
26. distribution >communication >hierarchy
27. distribution >facilitation
28. knowledge management >customer involvement
29. knowledge management >domain knowledge
30. knowledge management >domain knowledge >acquisition
31. knowledge management >domain knowledge >QA
32. knowledge management >knowledge sharing
33. knowledge management >lack of knowledge
34. knowledge management >learning process
35. knowledge management >tacit vs explicit
36. methodology >agile
37. methodology >agile >challenges
38. methodology >agile >collections
39. methodology >agile >collections >sprints
40. methodology >agile >feedback
41. methodology >agile >hybrid
42. methodology >agile >hybrid >streams
43. methodology >agile >perceived benefits
44. methodology >agile >products >releases >sprints
45. methodology >agile >reasons
46. methodology >agile >scrum
47. methodology >agile >scrum >teams
48. methodology >agile >surrogate customer
49. methodology >agile >test driven development
50. methodology >agile >transparency
51. methodology >concurrent development
52. methodology >experience
53. project management
54. project management >change process
55. project management >induction process
56. project management >metrics
57. project management >QA
58. project management >ramping up complexity
59. project management >refactoring
60. project management >testing
61. requirements >capturing method
62. requirements >capturing method >delivery story
63. requirements >capturing method >user story
64. requirements >change process
65. requirements >constraints
66. requirements >dependency
67. requirements >dependency >cross domain
68. requirements >dependency >difficulties
69. requirements >dependency >solutions
70. requirements >dependency >tracking
71. requirements >prioritization
72. requirements >prioritization >criteria
73. requirements >prioritization >discussion
74. requirements >prioritization >initial
75. requirements >prioritization >methodology
76. requirements >prioritization >methodology >tool support
77. requirements >reprioritization
78. requirements >reprioritization >decision process
79. requirements >reprioritization >external trigger
80. requirements >reprioritization >impact
81. requirements >reprioritization >internal trigger
82. requirements >reprioritization >inter-iteration
83. requirements >reprioritization >triggers

References

Beck, K., et al., 2001. Agile Manifesto. http://agilemanifesto.org/ (last accessed 5.5.12).

Alenjung, B., Persson, A., 2008. Portraying the practice of decision-making in requirements engineering: a case of large scale bespoke development. Requirements Engineering Journal 13 (4), 257–279.
Ambler, S., 2008. Agile and Large Teams. Dr Dobb's, June 17, 2008.
Ambler, S., Lines, M., 2012. Disciplined Agile Delivery. IBM Press, Boston, MA.
Auvinen, J., Back, R., Heidenberg, J., Hirkman, P., Milovanov, L., 2006. Software process improvement with agile practices in a large telecom company. In: PROFES, pp. 79–93.
Banerjee, U., Narasimhan, E., Kanakalata, N., 2011. Experience of executing fixed price off-shored agile project. In: Indian Software Engineering Conference 2011, pp. 69–75.
Barlow, J., Giboney, J.S., Keith, M.J., Wilson, D.W., Schutzler, R.M., 2011. Overview and guidance on agile development in large organizations. Communications of the Association for Information Systems 29 (1), 25–44.
Baskerville, R., Pries-Heje, J., Madsen, J., 2011. Post-agility: what follows a decade of agility? Information and Software Technology 53, 543–555.
Batra, D., 2009. Modified agile practices for outsourced software projects. Communications of the ACM 52 (9), 143–147.
Bosch, J., Bosch-Sijtsema, P.M., 2011. Introducing agile customer-centric development in a legacy software product line. Software – Practice and Experience 41, 871–882.
Bryant, A.K., Charmaz, K., 2007. The Sage Handbook of Grounded Theory. Sage, London.
Charmaz, K., 2007. Constructing Grounded Theory: a Practical Guide through Qualitative Research. Sage, Thousand Oaks, CA.
Cheng, B.H.C., Atlee, J.M., 2007. Research directions in requirements engineering. In: Briand, L.C., Wolf, A.L. (Eds.), International Conference on Software Engineering/Workshop on the Future of Software Engineering. IEEE CS Press, Washington, DC, USA, pp. 285–303.
Christou, I.T., Ponis, S.T., Palaiologou, E., 2010. Using the agile unified process in banking. IEEE Software, 72–79.
Cohen, D., Lindvall, M., Costa, P., 2004. An introduction to agile methods. In: Zelkovitz, M. (Ed.), Advances in Computers. Elsevier, Amsterdam.
Coleman, G., O'Connor, R., 2008. Investigating software process in practice: a grounded theory perspective. Journal of Systems and Software 81 (5), 772–784.
Daneva, M., 2003. Lessons from five years of experience in ERP requirements engineering. In: RE 2003, Toronto, Ont., Canada. IEEE CS, pp. 45–54.
Dibbern, J., Goles, T., Hirschheim, R., Jayatilaka, B., 2004. Information systems outsourcing: a survey and analysis of the literature. ACM SIGMIS Database 35 (4), 6–102.
Eckstein, J., 2004. Agile Software Development in the Large: Diving into the Deep. Dorset House Publishing Company, New York, USA.
Elshamy, A., Elssamadisy, A., 2007. Applying agile to large projects: new agile software development practices for large projects. In: XP 2007, pp. 46–53.
Gary, K., Enquobahrie, A., Ibanez, L., Cheng, P., Yaniv, Z., Cleary, K., Kokoori, S., Muffih, B., Heidenreich, J., 2011. Agile methods for open source safety-critical software. Software: Practice and Experience 41, 945–962.
Gat, I., 2006. How BMC is scaling agile development. In: AGILE'06, pp. 315–320.
Gregor, S., 2006. The nature of theory in information systems. MIS Quarterly 30 (3), 611–642.
Grewal, H., Maurer, F., 2007. Scaling agile methodologies for developing a production accounting systems for the oil and gas industry. In: AGILE 2007, Washington, DC. IEEE CS Press, pp. 309–315.
Hajjdiab, H., Taleb, A.S., Ali, J., 2012. An industrial case study for Scrum adoption. Journal of Software 7 (1), 237–242.
Hanssen, G.K., Smite, D., Moe, N.B., 2011. Signs of agile trends in global software engineering research: a tertiary study. In: Proc. of 2011 6th Int. Conference on Global Software Engineering Workshops, pp. 17–23.
Herrmann, A., Daneva, M., 2008. Requirements prioritization based on benefit and cost prediction: an agenda for future research. In: RE 2008, pp. 125–134.
Hong, N., Yoo, J., Cha, S., 2010. Customization of scrum methodology for outsourced E-commerce projects. In: Proc. of the 2010 Asia Pacific Software Engineering Conference (APSEC), Sydney, NSW. IEEE CS Press, pp. 310–315.
Hossain, E., Ali Babar, M., Paik, H.-Y., Verner, J., 2009. Risk identification and mitigation processes for using Scrum in global software development: a conceptual framework. In: 16th IEEE Asia-Pacific Software Engineering Conference, pp. 457–464.
Jalali, S., Wohlin, C., 2010. Agile practices in global software engineering – A systematic map. In: IEEE International Conference on Global Software Engineering (ICGSE 2010).
Kendall, R.P., Mark, A., Squires, S., Halverson, C., 2010. A case study of a large-scale, physics-based code development project. Computing in Science & Engineering, 22–27.
King, N., Horrock, C., 2010. Interviews in Qualitative Research. Sage, Thousands Oaks, CA.
Koehnemann, H., Coats, M., 2009. Experiences applying agile practices to large systems. In: AGILE, pp. 295–300.
Larman, C., Vodde, B., 2010. Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum. Addison-Wesley Professional, NY.
McDowell, S., Dourambeis, N., 2007. British Telecom experience report: agile intervention – BT's joining the dots events for organizational change. In: XP 2007, Agile Processes in Software Engineering and Extreme Programming, pp. 17–23.
Maiden, N., Jones, S., 2010. Agile requirements: can we have our cake and eat it too? IEEE Software 27 (3), 87–88.
Martin, A., Biddle, R., James, N., 2009. XP customer practices: a grounded theory. In: AGILE 2009, pp. 33–40.
Martin, J.L., Clark, D.J., Morgan, S., Crowe, J.A., Murphy, E., 2012. A user-centred approach to requirements elicitation in medical device development: a case study from an industry perspective. Applied Ergonomics 43 (1), 184–190.
Matavire, R., Brown, I., 2011. Profiling grounded theory in information systems research. European Journal of Information Systems, 1–11.
Oshri, I., Kotlarsky, J., Willcocks, L.P., 2007. Managing dispersed expertise in IT offshore outsourcing: lessons from Tata Consultancy Services. MIS Quarterly Executive 6, 53–65.
Patton, M.Q., 1999. Enhancing the quality and credibility of qualitative analysis. Health Services Research 34 (5 Pt 2), 1189.
Pfleeger, S.L., 2001. Software Engineering: Theory and Practice. Prentice Hall, London, UK.
Prikladnicki, R., Audy, J.L., 2010. Process models in the practice of distributed software development: a systematic review of literature. Information and Software Technology 52, 779–791.
Racheva, Z., Daneva, M., Herrmann, A., Sikkel, K., Wieringa, R.J., 2010. Do we know enough about requirements prioritization in agile projects: insights from a case study. In: RE 2010, pp. 147–156.
Racheva, Z., Daneva, M., Sikkel, K., 2009. Value creation by agile projects: methodology or mystery? In: PROFES 2009: 10th International Conference on Product-Focused Software Process Improvement, pp. 141–155.
Ramesh, B., Cao, L., Baskerville, R., 2010. Agile requirements engineering practices and challenges: an empirical study. Information Systems Journal 20, 449–480.
Rose, J., Pedersen, K., Hosbond, J.H., Kræmmergaard, P., 2007. Management competences, not tools and techniques: a grounded examination of software project management at WM-data. Information and Software Technology 49 (6), 605–624.
Runeson, P., Höst, M., 2009. Guidelines for conducting and reporting case study research in software engineering. Empirical Software Engineering 14 (2), 131–164.
Sarker, S., Sarker, S., 2009. Exploring agility in distributed information systems development teams: an interpretive study in an offshoring context. Information Systems Research 20 (3), 440–461.
Scheer, A.W., 2000. ARIS Business Process Modelling. Springer, Berlin.
Scholz, R.W., Tietje, O., 2002. Embedded Case Study Methods: Integrating Quantitative and Qualitative Knowledge. Sage, Thousands Oaks, CA.
Seddon, P., Scheepers, P., 2011. Towards the improved treatment of generalization of knowledge claims in IS research: drawing general conclusions from samples. European Journal of Information Systems, 1–16.
Shatil, A., Hazzan, O., Dubinsky, Y., 2010. Agility in a large-scale system engineering project: a case study of an advanced communication system project. In: IEEE Int. Conference on Software Science, Technology & Engineering, Herzlia, Israel. IEEE CS Press, pp. 47–54.
Sjøberg, D.I.K., Dybå, T., Jørgensen, M., 2007. The future of empirical methods in software engineering research. In: Briand, L.C., Wolf, A.L. (Eds.), Int. Conf. on Software Engineering/Workshop on the Future of Software Engineering. Minneapolis, MN. IEEE CS Press, pp. 358–378.
Sulfaro, M., Marchesi, M., Pinna, S., 2007. Agile practices in a large organization: the experience of poste italiane. In: XP 2007, pp. 219–221.
Sutherland, J., Viktorov, A., Blount, J., Puntikov, N., 2007. Distributed Scrum: agile project management with outsourced development teams. In: 40th Annual Hawaii International Conference on System Sciences HICSS, p. 274a.
Urquhart, C., 1997. Exploring analyst–client communication: using grounded theory techniques to investigate interaction in informal requirements gathering. In: Lee, A.S., Liebenau, J., DeGross, J.I. (Eds.), Information Systems and Qualitative Research. Chapman and Hall, London, pp. 149–181.
Valade, R., 2008. The Big projects always fail: taking enterprise agile. In: AGILE 2008, pp. 148–153.
Yin, R.K., 2008. Case Study Research: Design and Methods. Sage, Thousands Oaks, CA.

Maya Daneva, PhD, is Assistant Professor in the Information Systems and Software Engineering group at the University of Twente, the Netherlands. Her key research interests are requirements engineering for large systems, requirements-based project estimation, and qualitative research methods, such as grounded theory and focus groups. Maya has a strong international exposure having spent two years of her career in Germany at the University of Saarbruecken and in the IDS Scheer, and 9 years as a business process analyst at TELUS Corporation, Canada's second largest telecommunication company. She has been a leading member of several industry-university research projects and serves as the liaison to the industry members of the Dutch Software Measurement Association (NESMA).

Egbert van der Veen is Information Analyst and application designer for Topicus, a mid-size software company in the Netherlands. His main areas of interest include software development methodologies, distributed development and knowledge management. He holds a MSc. in Business & IT From the University of Twente, and a BS degree in IT Administration from the Saxion school of applied sciences in Enschede, the Netherlands. For his Master's graduation project, he conducted research at the Tata Research Development and Design Centre (TRDDC) in Pune, India in the area of large scale, distributed Agile software development.

Chintan Amrit is Assistant Professor at the department of IEBIS department, University of Twente. He has completed his PhD from the same department in the area of Coordination in Software Development. He holds a master's degree in Computer Science from Indian Institute of Science, Bangalore. In the past he

has worked as a software engineer for a software company in Germany. He is a track co-chair at ECIS 2013 and is a guest editor for a special issue on "Human Factors in Software Development" in the *Information and Software Technology* journal.

Smita Ghaisas is a Principal Scientist at TCS's Innovation lab- Tata Research, Development and Design Center (TRDDC). She leads the Requirements Engineering research group in the Software Engineering Lab of TCS. Her current research interests include applying knowledge engineering to enable requirements knowledge reuse and using empirical research techniques to discover conceptual models in various RE practices. She has authored and patented a requirements method that helps identify unobvious requirements. In the last ten years, she has conceptualized, and developed collaborative knowledge-assisted method and toolset and deployed those successfully in large-scale projects in TCS.

Klaas Sikkel is assistant professor in the Information Systems and Software Engineering groupat the University of Twente. He has a M.Sc. in Software Engineering from the Free University in Amsterdam and a Ph. D. in Theoretical Computer Science from the University of Twente. He has worked in groupware implementation at the GMD Research Center for Applied Information Technology in Birlinghoven (Germany). His current research interests include Requirements Engineering and qualitative research methods in Software Engineering.

Ramesh Kumar is currently the Head of Delivery & Solutions, Insurance & Healthcare, Emerging Markets for TATA Consultancy Services. He started his career as Mainframes developer and has performed the roles of Designer, PM, Architect, Delivery & Client Relationship Manager and IT Strategy Consultant during his career across the business domains of Insurance, Financial Services, Semiconductor, Chemical & Manufacturing. He is the creator of the Arva methodology for large agile delivery of business transformation programs that is based on principles of Theory of Constraints, Scrum development and large program management. This methodology has successfully delivery program over 250 MUSD and 600 person/years. The Business Solutions team lead by Ramesh won some of the large and strategic deals within Insurance unit.

Nirav Ajmeri is a PhD student and a Graduate Research Assistant at the Department of Computer Science, North Carolina State University (NCSU). He received his B.E. in Computer Engineering from Gujarat University in 2007. Prior to joining NCSU, he worked as a Researcher with the Requirements Engineering Group at Tata Research Development and Design Centre (TRDDC). His primary research interests are knowledge engineering, requirements engineering, natural language processing, argumentation and multi-agent systems.

Uday Ramteerthkar is heading the MasterCraftAppMaker Suite under the Component Engineering Group in Tata Consultancy Services Ltd. This product suite is Model Driven Framework which is aimed to provide higher productivity, better software code quality. He has experience in development, deployment of tools, different software development methodologies & architectures. In his prior roles, he was involved in the implementation of Core Banking Solution for one of public sector banks. He has led Data Services portfolio for a for a large bank holding company, which involved database administration, data warehousing, modeling and data migration. He has done his Master of Technology from Indian Institute of Technology, Mumbai and is a certified TOGAF architect.

Roel Wieringa is Chair of Information and Software Systems Engineering at the University of Twente, the Netherlands. His research interests include modelling and design of e-business networks, requirements engineering, and research methodology for software engineering and the design sciences. He has written two books, *Requirements Engineering: Frameworks for Understanding* (Wiley, 1996) and *Design Methods for Reactive Systems: Yourdon, Statemate and the UML* (Morgan Kaufmann, 2003, and is currently preparing a book on Design Science Methodology. He has been Associate Editor in Chief of *IEEE Software* for the area of requirements engineering from 2004 to 2007. He serves on the board of editors of the *Requirements Engineering Journal* and of the *Journal of Software and Systems Modeling*.