Manuscript Number:

Title: Agile Requirements Prioritization in Large-Scale Outsourced System Projects: an Empirical Study

Article Type: Special Issue: Agile Development

Keywords: Agile; Requirements prioritization; Outsourced software development; Qualitative research; Case study

Corresponding Author: Dr Chintan Amrit, Ph.D

Corresponding Author's Institution: University of Twente

First Author: Maya Daneva, Ph.D

Order of Authors: Maya Daneva, Ph.D; Egbert van der Veen; Chintan Amrit, Ph.D; Smita Ghaisas, Ph.D; Klaas Sikkel, Ph.D; Ramesh Kumar; Nirav Ajmeri; Uday Ramteerthkar; Roel Wieringa, Ph.D

Abstract: The application of agile practices for requirements prioritization 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. The study was carried out as a series of three embedded case studies in a large and mature company engaged in outsourced software development. Our grounded theory analysis of in-depth interviews yielded the following findings: (i) Understanding requirements dependencies is of paramount importance for the successful deployment of agile approaches in typical large outsourced projects. (ii) Next to business value, the most important prioritization criterion in the setting of typical outsourced large agile projects is risk. (iii) The software organization 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. These play a pivotal role in prioritization. (iv) The vendor's domain knowledge is key asset for setting up successful client-developer collaboration (v) The use of agile prioritization practices depends on the type of project outsourcing arrangement.

Dear Guest Editors,

Please accept our submission to the special issue on "Towards a unified view of agile software development". We would like to thank you for kindly allowing us an the extra week of time to improve the paper,

Yours Sincerely,

Chintan Amrit

(On behalf of the other authors)

Agile Requirements Prioritization in Large-Scale Outsourced System Projects: an Empirical Study

Maya Daneva[1], Egbert van der Veen[1], Chintan Amrit[1], Smita Ghaisas[2], Klaas Sikkel[1], Ramesh Kumar[2], Nirav Ajmeri[2], Uday Ramteerthkar[2], Roel Wieringa[1]

[1]University of Twente
Enschede, The Netherlands
{m.daneva, c.amrit, k.sikkel, r.j.wieringa}@utwente.nl
e.vanderveen@student.utwente.nl

[2]Tata Research, Development and Design Centre
54 Hadapsar Industrial Estate
Pune, 411013, India
{smita.ghaisas, ramesh.kumar, nirav.ajmeri, uday.ramteerthkar}@tcs.com

Abstract

The application of agile practices for requirements prioritization 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. The study was carried out as a series of three embedded case studies in a large and mature company engaged in outsourced software development. Our grounded theory analysis of in-depth interviews yielded the following findings: (i) Understanding requirements dependencies is of paramount importance for the successful deployment of agile approaches in typical large outsourced projects. (ii) Next to business value, the most important prioritization criterion in the setting of typical outsourced large agile projects is risk. (iii) The software organization 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. These play a pivotal role in prioritization. (iv) The vendor's domain knowledge is key asset for setting up successful client-developer collaboration (v) The use of agile prioritization practices depends on the type of project outsourcing arrangement.

Keywords: Agile; Requirements prioritization; Outsourced software development; Qualitative research; Case study;

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. A key pillar in any agile process of systems delivery is the close and continual collaboration between clients and developers [1], which culminates in making project decisions that optimize client's business value, while minimizing vendor's risk. 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 [2]. 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 [3]. In the area of agile RE, relatively fewer publications address the project realities of large and distributed projects. Even fewer focus on the complex RE settings unique to outsourced software development. For example, 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 [38] can impact agile RE practices. For example, the work practices related to agile RE for fixed price projects could differ from more flexible cost/ in-sourcing projects. More specifically, the question of how to balance value and risk in agile has received practitioners' and researchers' attention only recently. 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 bounded, poor trade-offs of value and risk might often have severe consequences [3].

In this paper we explore the value versus risk trade-off in agile RE prioritization with an embedded case study [15, 22]. We carry out this case study in a context in which three large, multi-site enterprise systems are delivered under a contract between a large and mature outsourcing vendor in Asia and three dispersed client organizations. Reusing a conceptual framework for agile requirements (re)prioritization that we developed earlier [2], we investigated these real-world cases. The overall research objective was to uncover how mid-course requirements prioritization takes place in large projects in industry, and what notions of value and risk are included in it from the vendor's perspective.

We set out to answer the following four research questions (RQs):

RQ1: Who are the decision makers in the prioritization process? Which roles are involved and what are they responsible for?

RQ2: What criteria do large project teams use to make risk-and-value driven decisions during agile prioritization 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?

Our answers to the four questions resulted from an embedded case study [22] that is exploratory in nature because we had no pre-conceived ideas about the possible answers to our questions. 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.

Our motivation for this embedded case study is traceable to our previously published results [2, 4] from a systematic literature review and a case study on prioritization methods in agile projects. Therein, our conclusion was that the phenomenon of agile requirements prioritization was only partly explored, as we had found no study which clearly indicates 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.

The research reported in this paper brings two contributions to the body of knowledge in agile RE. While most case studies dealing with agile RE have focused on small-medium enterprises (SMEs), our research explicitly states the RE knowledge while providing information on which practices (depending on the outsourcing arrangement) work in large projects associated with a large and mature outsourcing vendor. Second, our research contributes to the body of empirical software engineering (SE) studies in general, and to the body of the empirical studies in RE, in particular. This contribution is twofold: (1) we directly respond to the call [5] of the empirical SE community for more empirical research on which SE process to use in which specific context, (2) we also respond to the particular call [6] of the RE community for more empirical research in the sub-areas of RE.

In the following narrative, we first present related work on agile RE in large projects. We present the experiences reported in the literature with respect to our research questions. Then, we describe our embedded case study research design along with the justification for the decisions about the particular way in which we set up our research process. Next, we report on our case study execution and its outcomes. Finally, we discuss the results and the limitations of the study, reflect on the experiences, and present our future research activities.

2. Related Work: Agile RE in Large Projects

This section summarizes the state of agile RE literature with respect to the focus of our research questions. The results of our search for relevant published sources indicate that many large organizations were indeed preoccupied with the question of how agile scales up. We found a number of recently published examples of large companies that attempted to amplify agile practices in order to support large projects. In most reports, there is a discussion provided on what the respective company found to work in their settings. We note that all literature sources we reviewed (and included in this section) treated RE as part of agile SE for large projects. In spite of our best efforts, we could find no report that was specifically dedicated to agile RE in the large. In Table 1, we summarize these examples of using agile in large projects by indicating 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 on the use of specific new or modified agile SE practices. The last column indicates whether the authors contextualize the use of specific practices in the setting that they observed or were part of. Table 1 includes empirical reports that referred to projects with staff of more than 60 people and in which agile RE practices were described in detail. We note that the list of included sources is not claimed to be complete, but represents a sample that was relevant to our research questions. In Table 1, the `☑' or `–' marks signify whether or not the use of user stories and new/modified agile practices is clearly stated in the paper. The last column describes to what extent the requirements prioritization process is discussed.

Table 1: Selected Literature Sources on Agile at Large

Source	Business Sector	User stories	Modified Agile practices	Discussion of prioritization?
[7]	Mobile systems	☑	☑	hints
[8]	IT service delivery (a vendor)	☑	☑	hints
[3]	Telecom	☑	☑	explicitly
[9]	Telecom	☑	–	not discussed
[10]	National Post Services	☑	–	not discussed
[11]	IT service delivery (a vendor)	☑	–	not discussed
[12]	Enterprise IT infrastructure management solutions	☑	–	explicitly
[13]	Large government organizations	☑	–	explicitly
[31]	Safety-critical systems	-	☑	hints
[23]	Oil & gas (a vendor)	-	☑	hints
[24]	Digital library management (a vendor)	☑	☑	hints
[25]	ERP Implementation	☑	☑	hints

[26]	Software product business (a vendor)	☑	☑	hints
[27]	E-commerce	☑	☑	hints
[28]	Physics-based computational engineering	-	☑	not discussed
[29]	Banking	-	☑	not discussed
[30]	Defence system engineering (a vendor)	☑	☑	hints

Table 1 indicates that large project organizations delivering 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. Those who did not adopt the user stories as an approach for their business requirements redesigned their requirements specification practices to “become more agile” [23, 28, 29, 31]. We observe that while the literature sources discuss explicitly the user story elicitation and documentation practices, they provide mostly hints regarding the requirements (re)prioritization process at inter-iteration time. We note, however, that the number of publications on agile practices in large projects is significantly less than the overwhelming amount of ‘success stories’ in agile projects in SME. This is not a surprise, as Ambler indicates that most agile teams are less than 10 people and are collocated [14]. Moreover, Eckstein [3], 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 higher. While this sounds intuitive, it implies that the RE community is exposed to less empirical evidence on how large-scale agile RE happens and, in turn, little is known about whether theory and practice are always consistent. As we indicated in our earlier study [2], deviations of the real processes from the rules of agile development are rarely reported – which does not necessarily mean that they do not exist.

In the rest of the section we present what these sources say with respect to our four questions.

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

All published sources mentioned 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’ [3] (or ‘stream of streams’ [23]) approach. Examples of roles that companies introduce in large agile projects and that provide input to 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” [12]) and (ii) business area owners [13]. 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 in a timely manner. The scrum master is well aware of dependencies among teams and uses this knowledge to resolve them as early as they arise [27].

RQ2: What criteria do large companies use to make risk-and-value-driven decisions during agile prioritization in an outsourced mode?

Based on whether the clients’ or vendor’s perspective is considered, the studies suggest business value and risk, respectively, to be the key prioritization criteria. The agile literature sources implicitly assume that client’s value and vendor’s risk are both attributes of a large agile project [3, 6, 13, 23, 26, 28, 29, 30].

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 win-win for both parties, seems to be unknown. To the best of our knowledge, there is no empirical study that explains how risk-minimization takes place during agile RE. Moreover, our earlier research on the business value generation by means of agile prioritization [2] 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 documented.

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

The agile RE literature discusses two characteristics of the project setting which influence decision-making in requirements (re)prioritization at inter-iteration time: *need to embrace change* and *project constraints*. However, the agile authors point out that in large projects these characteristics are complemented with *scope* and the *number of project staff*. The last two are deemed 'first order dimensions' [3] that define the 'largeness' of a project. *Scope* characterizes the complexity of the requirements, while *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" [3], and because new problems may surface due to the largeness of the team (and those are easier to handle in small teams).

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

Three sources [3, 13, 23] 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. In [2], we have 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 what the level of trust between client and vendor is. Eckstein [3] and Larman [13] do 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.

3. Research Method

Our embedded case study design follows the guidelines by Yin [15] as well as Scholz and Tietje [22]. 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 [15] 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 [15]. We applied purposeful sampling [17] when looking for suitable projects. Our purpose was to find projects that had dispersed users and a project team that already earned experience in using an agile delivery model, or at least used a collection of agile practices. Hence, as we describe in the following subsection, our case study involves more than one unit, or object, of analysis [17]. We used in-depth interviews [16] as a data collection technique and grounded theory [17] as a data analysis technique. The in-depth interviews technique was selected because of two reasons: (1) it is a suitable technique for an inquiry like ours, and (2) the resulting data offers a robust alternative [16] 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. In our data collection, we triangulated data 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 provide a multidimensional image of composing activities in our particular project setting.

As mentioned above, we carried out the data analysis by implementing the grounded theory (GT) building techniques according to Kathy Charmaz [17]. GT is a qualitative method applied broadly in social sciences to construct general propositions (called "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 [17] mean situations in which the research does not start with hypotheses or a predefined theory and then seek proof. Instead, the researcher is driven by the desire to capture all facets of the collected qualitative data and to allow "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 "unchartered land" (as Baskerville et al. refer to it [35]), or a phenomenon about which little is known. Recent examples of GT studies in SE include the publications [35, 36, 37]. In the RE field, examples of applications of GT as an empirical research approach are published by Urquhart [32], Martin [33] and Martin et al [34]. The GT process is presented in more detail in Section 3.3.

3.1. The case study process

As indicated in the Introduction, our research questions were motivated by our earlier research that took place in the context of SMEs. We acknowledge, however, that the context of this study is different (namely, large projects in a large organization) and as a result has its own issues. Therefore, we kept an open mind and did not have any preconceived ideas of what the answers would be.

Our exploratory case study process included 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 semi-structured interviews with practitioners according to the finalized questionnaire,

(6) Sample (follow-up with those participants that possess deeper knowledge or a more specific perspective).

We carried out 16 interviews in total. The interviews were 1 hour and 20 minutes long on an average. All interviews were conducted either face-to-face or facilitated through video conferencing. For 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 each 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 case study. At 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. The second part included questions related to the risk, the business value perception and how the 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 to clarify the consideration of risk and value as part of the prioritization decision-making process. Here, we make two notes: (1) that no substantial changes in both the questionnaire and case study protocol took place after the pilot interviews, so that these could be considered part of the case study; and (2) that during the interviews there were instances when questions, other than those included in the questionnaire arose. These questions were not 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 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, specifically a form of Scrum that was adapted to the specific context. All 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 who were working at various capacities on one of the three projects. This allowed for including a variety of perspectives on the studied phenomena, which in turn, ensured triangulation of data in our research process [15].

The case study projects are described in Table 2 in terms of (i) type of engagement, (ii) scope, (iii) number of team members (staff), and (iv) project duration. For confidentiality reasons, we refer to these projects as project Alpha, project Beta, and project Gamma (Table 2). 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. Project Gamma on the other hand, was a medium size project. With respect to the nature of contractual agreement between vendor and client, project Alpha was a fixed-price/fixed-duration project, while projects Beta and Gamma had a flexible timeline. From the three projects, specific case study participants were selected based on the following criteria: (i) they had professional characteristics in common, which pertained to the topic of our study and (ii) they had the potential to offer information-rich experiences.

The list of our case study participants is presented in Appendix A. Therein; we provide the details of the participants' roles in the company as well as the mode in which the respective interviews were conducted.

Table 2: The details of the different projects we considered as part of our embedded case study

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	124	35-40 + Client Team of about 100	35-40
Cost and Duration	Fixed	Flexible	Flexible
Modularity of Product Architecture	Low	Low	High

In what follows, we describe the contextual setting of each case study project in more detail.

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 that had the objective to deliver a large enterprise system for 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 analysts, two business analyst leads, one delivery head (responsible for transforming the clients' user stories into 'delivery stories' that include architecture design decisions and non-functional requirements), one portfolio manager (who was responsible for the group of client's 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).

Project Beta

The second project consisted of a team of around 35 to 40 people, whose function was to complement an existing development team on the client's side, leading to an overall project consisting of 150 people. The team's engagement was not for a specific project, but was in place in order to enhance the client's existing development effort. This engagement started three years earlier to the time of the interviews, with Agile having been adopted as a development methodology for the previous year. Various products were under active development, and these products were divided into 'releases', with each release consisting of a number of sprints of either two or four weeks. Requirements here were in constant flux, with the mind-set to accommodate requests from the business whenever possible, although, requirements taken up during a particular sprint were kept as consistent as possible for the duration of that sprint.

Project Gamma

In project Gamma, the client was an internal, but not co-located business customer. This project differed from the other two projects in that the product was developed 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 former team as the 'development team' and the latter 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 26 people, with 21 people on the developer's side and 5 on the client's side (again, we refer here to the internal client). The product was a model-driven development environment. In contrast to the other cases, the development team was the actual owner of 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 the other cases, the requirements for the product was 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, that it worked towards. 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 switched to Agile only four months before the interviews were conducted. This transition was triggered by the advantages the client team achieved through their own transition to Agile. 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 cases, this happened concurrently). As with the second project, 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 of the researchers (Van der Veen). Because this researcher was in a country in Asia, while the analysis was supposed to be done in collaboration with another researcher (Daneva) located in Europe, the researcher responsible for data collection (Van der Veen) took special care of the transcribed information. For example, the first researcher added notes about the non-verbal language of his interviewees, and indicated whenever an interviewee paused and expressed doubts regarding the completeness of the information provided. This turned out to be critical information to the other researcher who was engaged in the coding. Once the transcripts were ready, we applied the data analysis guidelines of the GT approach [17]. Essentially, GT analysis includes 'coding' and 'constant comparison' of the interview data. Coding is the 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 the process through which we constantly compare instances of data that we have named as a specific category with other instances of data, to see if these categories fit and are workable. This process helps grouping the data into categories. The resulting codes and categories guide the writing-up of our results and aid in improving the accuracy of the claims [17].

GT methodologists recommend that coding and constant comparison be done iteratively. 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 was discussed and peer-reviewed iteratively with two other authors (Amrit and Sikkel) of this paper to establish consistency and the emerging clustering into categories. Our analysis proceeded as follows: the 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', 'risk'), or an activity (e.g. 'risk 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 we duly noted. We then clustered all pieces of text that relate to the same code, in order to analyse it in a consistent and systematic way. We then compared the codes 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 respectively - codes, emerged during the coding process as a result of observations we had not anticipated. These primarily concerned concepts and aspects of the process we had not explicitly addressed in the questionnaire. Those were 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 5). This example uses transcribed text from the project Beta, where the code corresponds to the italicized interview text.

The emerging categories were analysed in relation to our research questions. The categories were clustered in headline themes that addressed our research questions (as described in Section 4) – i.e. similar categories were aggregated into a headline theme (each headline theme one of our four research questions). The results of the data analysis are presented in the following section.

4. Results

4.1. Roles of clients and of developers

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

From vendor's perspective, our findings indicate that people that 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 with the client's *Product Owner*. We note that unlike the product owner's role, as explained in agile books (e.g. [13]); 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's business decisions are informed 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. The vendor's organization is represented by five roles: (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 of the business domain of the client, 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, who design a system architecture that meets all the non-functional requirements, namely maintainability, changeability and usability, and (v) *Test Scenario Team Leads* who are responsible to ensure technology dependencies and constraints are considered as early as possible in the agile reprioritization process. We now illustrate some examples from our interview data that support the above findings.

> *"It is **product owners** [who are responsible for decisions on priorities]. Then [our] delivery story team, we have BAs; **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 role played by the *Product Owners*, who are then complemented by the *Business Analysts* and *Tech Leads*. P4 and P0, also from project Alpha, had the following to say about the business analyst, domain owner, product owner and the solution architect roles.

> *"The collection team has **business analysts** from the client, then domain owner from the client, then technical people; testers, everyone is involved in that."* (Alpha, P4)

> *"…for each SCRUM, for each collection, we have to have one **domain owner**, **product owner** from client-side , so they will be at off-shore, whereas all the solution architect team from there and risk compliance team, all these teams will be there in on-site."* (Alpha, P0)

On the other hand P1 in project Beta had the following to say 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 by the term *IS Analyst;* P1 refers to the *Project Owners* that we describe in the paragraph above. Hence we see that the importance of the *Business Analysts* and *Project Owners* in the development process.

P2 from the same project Beta had the following to say about roles and prioritization:

> *"[…] 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 role played by the *Tech Lead,* as described in our findings above. P2 (from project Beta) in above statement describes the crucial role played by the *Tech Lead* in prioritizing the requirements where the 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)

In the above quote participant P4 (project Alpha) describes how the *Domain Owner* role from the client 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 became moot at the time of implementation.

Furthermore, a designated delivery team comprising senior architects considered the pragmatic approach to include non-functional requirements early in the agile RE cycles, trace those to concrete architecture design choice, and analyse the impact of changed requirements on the architecture.

4.2. Risk-based criteria

RQ2: What criteria do large companies use to make risk-and-value-driven decisions during agile prioritization in an outsourced mode?

All our interviewees indicated that the concept of business value 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 what represents business value for their organization and that the client would communicate proactively any change that affects 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, each one (according to our participants), to 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 user stories translated into design and test cases. 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 subsection 4.3.3. Hence for project Alpha, risk was related to anything that endangered the project delivery according to the fixed-price/fixed-schedule contractual agreement. (We will see later in this section that risk is as important a prioritization criterion as business value). However, though less critical, even for the projects Beta and Gamma with flexible price/schedule agreements, there was risk associated with repetitive project delays and additional costs.

All participants deemed the delivery stories as the pillar in the requirements reprioritization process, because of the estimates being done 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 choices made earlier.

Most interviewees in project Alpha and Beta indicated that *Requirements Dependencies*, *Volatility, Risk*, *Effort*, *and Technical Depth* 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) had the following to say:

> *"[...] 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 (project Alpha) describes the prioritization difficulties faced due to the volatility (from the technical perspective, as we shall explain later), dependence among requirements and specifically due to dependent user stories.

P2 from project Beta had the following to say about requirement 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, in project Gamma *Requirements Dependencies* played a lesser role in the prioritization of requirements. The reason for this was because 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). As evidenced 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 that refer to sequencing requirements that build upon each other within a specific business process (for example, the closure of a client insurance file depends on certain activities that must happen before the file is ready for closure); For example, Interview Quote 9 (from (i) above) also describes the intra-domain dependencies – dependencies between processes and entities

(iii) Dependencies *due to downstream activities*, which means 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 said:

> *"[...] 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 a group of requirements that the client anticipates to change. Our interviewees suggested that 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 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 that call for reprioritizing the requirements, so that the project gets finished on time and within budget. For example, as participant P0 in project Alpha shared:

> *"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 (project Alpha), clearly illustrates volatility due to changes in the business perspective. On the other hand, the first quote in this section (Alpha, P2), also illustrates 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 change that calls for severe redesign of the architecture, or any change that impacts 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 highly maintainable system as a risk. Apart from risk, *Effort* 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 (as feasible) as an input to the requirements reprioritization process.

Lastly, *Technical Depth* implies the amount of architecture-redesign related work that accumulates over a period of time, due to having a 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 significant effort spent on redesigning the architecture. Technical depth refers to those situations in which development is considered safe to start without much focus on architecture design. For agile projects in such a context, technical depth becomes a prioritization criterion. Table 3 presents how one participant from project Beta explained this concept.

We note that according to our interviewees the concepts *Risk* and *Effort* seem 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) is not a risk for the client. But for the vendor this would mean having to 'spend' (elsewhere) employable people at no extra price. 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 would become a client's risk 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 explicate 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, 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 re-work. 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?

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 roles mentioned	The 5 roles mentioned	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 Depth	High	Medium	Low

We found that the requirements prioritization processes was consistent throughout project alpha regarding: (i) the shared understanding of 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, (v) the communication and escalation procedures that were followed to interact with the client's product owner. The perceptions on how the reprioritization processes was carried out varied from participant to participant. Different actions were taken by different participants as a response to risk estimation information. Each participant represented a specific role and the variation is traceable to this role. This implies that though the interview data provided by business analysts did not diverge, yet 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 project Beta and project Gamma the answers to our research questions 1, 2 and as a consequence 3 were different, as seen in Table 3. For RQ1, though the number of roles was the same for project Beta, they were significantly less for project Gamma. Also, most of the criteria we mentioned as important for project Alpha had reduced or no importance for projects Beta and Gamma. We think the reason behind this could be the different degrees of outsourcing (Table 2) for projects Beta and Gamma, as compared to the more traditional outsourcing scenario in project Alpha.

Some factors that influence the prioritization process could be expected, based on literature. 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 risk.

In addition we discuss some context factors, which, based on the interview data, seem to be responsible for the consistency of the requirements reprioritization process (and the types of variations we observed). These are: maturity of the organization and the way domain knowledge was shared. An essential element in the prioritization, moreover, is the use of delivery stories as an artefact in the agile process in the organization in which the case studies were conducted.

4.3.1 Maturity of the organization

Apparently, being mature is a vendor's quality that is instrumental to the scaling up of agile practices, such that pieces of enterprise system functionality with 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 the 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.

We note here, that we do not mean any 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 may 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 that it's not the maturity level alone that determines the success of agile practices. Instead, we think that it is the special focus on process excellence and the organization's urge for adopting and adapting agile values and practices, which has influenced 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 will be used in the organization and in the project. The plans for using agile approaches were elaborated in detail (by the vendor's program management office and process excellence team) and written down to make sure that all processes are defined and followed. To be effective, the policy had basic information that everyone working on the project must know.

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) seems to be an important context characteristic that affects the way in which agile requirements prioritization was implemented. A feature of the agile RE process in our case studies is the complementary use of the *Domain Owner* role. This puts an emphasis on accumulation and sharing of knowledge and, where needed, transfers 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)

We think that the fact that the vendor proactively considered the need for sharing knowledge (and in turn organized the agile RE process by paying much attention to it) is not accidental, because domain knowledge sharing is related to vendor's maturity and commitment to adopt/adapt agile values. As stated in the Agile Manifesto [19], 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. For a highly mature vendor, one can easily assume that the vendor will want its processes propagated broadly and across different locations. As we can expect from a highly mature organization that has already established both the infrastructure and the mechanisms to preserve knowledge and information (and methodically apply approaches to making knowledge explicit across teams), it is natural for them 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 artifacts

General practice in agile methods is the translation of requirements into user stories. One novel concept that was discovered with regard to the implementation of Agile in this setting was that of 'delivery stories'. This was introduced by the team of project Alpha and later adopted by project Beta as an internal best practice. Delivery stories are created by taking user stories and turning them into a functional specification, high level design and test scenarios. This method provides the necessary translation step in a situation where developers do not directly receive requirements from the clients. 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)

The concept also provides large projects with something more measurable and concrete than the relatively less quantified concept of user stories. Multiple participants from projects Alpha and Beta stressed the importance of delivery stories. However, though the participants in project Gamma did not explicitly use the concept of *Delivery Story*, they did have an intermediate translation model between the gathering of requirements and the final architectural design specification[1]. From the interviews, as well as

[1] Project Gamma, although it did not use delivery stories specifically, did formalize its requirements process to a certain extent. Project Gamma had a consultancy team that received requirements from the client in the form user stories. The consultants then translated these short descriptions into more concretely defined specifications that could be handed over to developers. In this way, project Gamma represented, in some ways, an intermediate stage between the less explicit requirement gathering found in typical agile projects, and the highly formalized delivery story process we observed in Alpha and Beta.

with the help of project documents made available to us, we have pieced together the steps that go into the user story – delivery story translation process.

First, the vendor's business analysts acquire the user stories from the client. This user story is then assigned an owner, usually the person responsible 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 through 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. Although participants did indicate that they would like to adopt a more formal approach if one was available. Features are then mapped into sub-processes, which then become the eventual delivery stories as they are 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 cases. This process is summarized in figure 1.

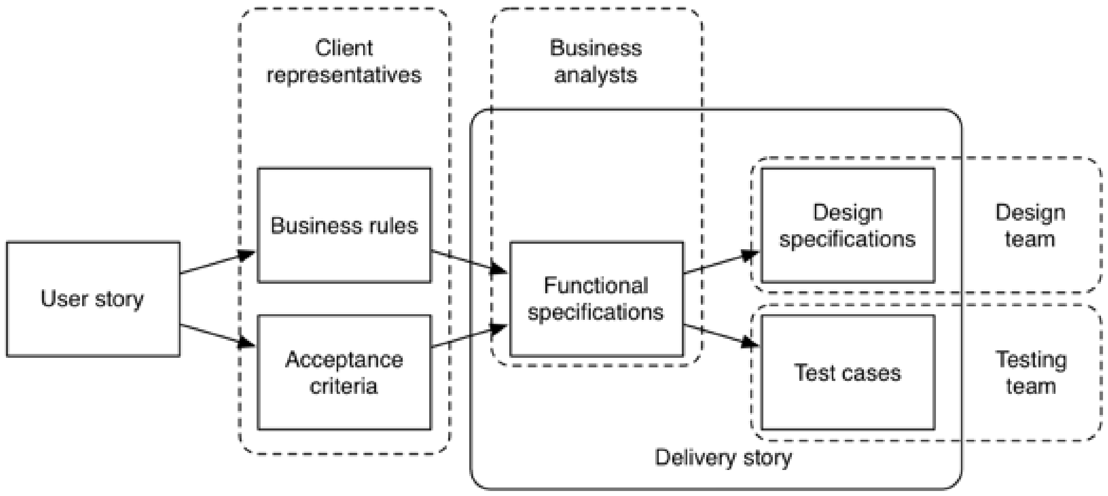

Figure 1: The User Story – Delivery Story translation process

It should be noted that all the different elements depicted in figure 1 are mapped onto one another through a system of numbering of the specific user story, business rule and functional specification. This ensures traceability throughout the process, and enables project management to maintain a high-level overview.
At the end of this process, the following properties are associated with each delivery story:

- Amount of use cases
- Required time for use case implementation in person days
- Amount of GUI Screens
- Required time for GUI Screen implementation in person days
- Business rules and validations
  - Amount of business rules
  - Amount of validations
  - Amount of field validations
- Amount of test cases
- Required time for test case preparation in person days
- Required time for data models in person days
- Total time required for sub-process implementation

These delivery stories are created in close collaboration with the client's representatives. After this process, requirements are concrete enough that they can be handed over to the actual development team while minimizing uncertainty. 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 the vendor and value creation for 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 that the case study participants considered 'to work well' in a large agile project where there was a clear need to balance risk and value. The participants perceived risk and value as 'opposing sources' and used this synonymously to 'combine value-creation for the vendor with value creation for the client'. The practices are formulated below in 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 that the client wanted into architectural design.

3. Run series of workshops (user story workshops and delivery story workshops). This is the particular way in which the large team of the vendor created rapport and built trust with the clients. (The mode 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 that call a product owner someone on the vendor's site who 'represents' the client, in the project of our case study it was the client who installed the role of Product Owner.

7. Use the requirements change impact analysis process set up at the vendor's side to communicate to clients. The vendor's team had clearly defined policies about what requirements changes were permissible and what were not. They consistently used the change impact process whenever they found that there was elevated risk associated with specific delivery stories.

8. Have changes confirmed directly with the Product Owner and resist the temptation to second-guess the client.

9. Use status (each 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 gradual training to gain knowledge of the domain. In the case study project, training activities were all supported by the client organization. For example, the client 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.

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. [38] in their extensive review of outsourcing literature, provide a typology of outsourcing arrangements (page 12, [38]). In the typology they mention the degree of outsourcing (total, selective and none) as well as the amount of ownership (internal, partial and external).

Table 4: Types of Outsourcing Arrangements applied to the Embedded Case studies [38]

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

As seen 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 observe that certain roles as well as prioritization criteria are given more importance as we move from an typical Outsourcing scenario (project Alpha) to a more internal sourcing scenario (project Gamma) (Table 3). We also noticed that certain practices like the use of Deliver stories, that we noticed in projects Alpha and to a lesser extent in project Beta were not so important in project Gamma.

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 about the following:

a) roles on the client's and vendor's sides, like the significance and utulity 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 context of the embedded case studies

We conjecture that the emphasis on specific roles as well as the need for special artefacts (documentation) was specifically due to the high maturity of the organization the case studies were embedded in. We now discuss some findings in more detail and add some general observations.

First, from vendor's perspective, risk, volatility and effort are deemed key prioritization criteria next to business value. The concept of requirements dependencies plays a critical role in managing risk while deciding 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 cascading re-work in numerous different places. However, requirement dependencies play a significant role only if the architectural design is not very modular and/or if the dependencies are not universally well known 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.

Second, an important role was played by the program management office and the process excellence team in the vendor's organization. Their involvement in scrutinizing scenarios of how to implement agile practices in a project was critical. In our different embedded case studies, we noticed these two teams let the project team try out a proposal for 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.

Third, this set up, allowing lessons learned to be incorporated, lead to a new concept that merits further attention: the *Delivery Story*. 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 can play an important role in (re)prioritization, because they have an associated effort and risk. Also, fleshing out the high-level design details helps in better understanding the requirements.

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 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 at British Telecom [9], that 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. The vendor referred to domain knowledge as a reusable asset in follow-up projects. This 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 the follow-up projects that would take place with the same client. We note that, indeed, the vendor's organization have taken up several initiatives to build reusable domain knowledge assets based on the lessons learnt by delivery teams and domain owners while implementing large projects. There is evidence that the pay-offs of the domain owner role are fully justified from other fields, where similar roles have been documented. In the field of RE for enterprise systems (e.g. ERP) there is enough evidence about the worth of the domain owner role. For example, earlier publications by one of the authors [20] revealed that for RE to succeed in enterprise system projects, the team must have three types of expertise: (1) in the business processes common for the business sector of the client, (2) in business processes of the specific client organization, and (3) in requirements models that 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 [21] (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 [21]. 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, much research is needed to collect and analyse evidence that would possibly confirm or disconfirm this hypothesis. The vendor's organization has taken up several initiatives to build reusable domain knowledge assets based on the lessons learnt 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 validity of the observations and conclusions in a case study [15]. For this purpose, we used the checklist for case study researchers recommended in [18]. Because we planned and executed exploratory qualitative research, the key question to address when evaluating the validity of its results, is [15]: to what extent can the practitioners' experiences and the requirements prioritization mechanisms we observed, be considered representative for a broader range of companies? We think that the conclusions we draw will hold for other companies in contexts similar to that of 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 that there are teams (similar to our case study organization's Program Management Office and Process Excellence

Team) that 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) who 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 that vendors 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 guidelines [19] to large distributed (and possibly multicultural) project settings. More often than not, they expect the vendors to take care of these issues, while insisting that the system be delivered using agile practices. In the light of this market development, we assume that organizations that already have formed internal program management/process excellence units are most likely to find our conclusions relevant and useful.

We also acknowledge the inherent weakness of interview techniques that they are driven by the researcher, meaning that there is always a residual threat to the accuracy of what interviewees say. However, we believe that in our study, this threat was reduced, because each interview conversation was completely transcribed and every transcript available for reference purposes.

Moreover, a validity concern in interview-based studies is that the researcher influences the interviewee. To counter this threat, special attention was paid to conducting the in-person interview in such a way that the interviewee felt comfortable in answering the question. The researcher who interviewed the participants made sure the interviewee is not avoiding the question and feels at ease. This created a safer environment and increased the chances to get a truthful answer rather than one aimed to please the interviewer.

To minimize potential bias of the researcher, we considered also construct validity of our study. We followed Yin's [27] recommendations in this respect, 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). Second, we had the draft case study report reviewed by researchers from the company that 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 in a number of respects: (i) The choice of the projects was not motivated by any other criteria except that they all needed to be agile. The authors relied on their professional and personal network to establish contacts with the project members. (ii) The studied projects are not representative for all the possible ways in which prioritization is performed in large agile organizations. We, however, consider that our findings can be observed in companies and projects that have similar contexts to those included in our study only (as already discussed earlier in this section).

7. 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 case study shows 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 a novel type of artefact, so-called *Delivery Stories*. These complement user stories with architectural design implications, test cases, 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 costs for the vendor in projects in new

domains, but it is believed that in the end it is a profitable investment to capture business knowledge as well as architecture patterns for a domain.

From the data there is no evidence how the vendor's domain 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; (ii) next to 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.

The study has the following implications for research and practice: First, to researchers, risk in large agile projects is deemed important, and merits further investigation. We assume that not all risks to project success are equal for a vendor organization. The questions of what kind of risks are perceived most important, what risk analysis models and what risk mitigation strategies companies use, form a line for future research.

Second, Delivery Stories as artefacts have a role to play in large-scale agile software engineering. Eventually these should be described more precisely and find their way into handbooks and textbooks. Third, our study suggests that 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 repositories could be used, but how to go about this is an interesting topic for further research.

Acknowledgment

We thank the interviewees for participating in our research. We also thank Ms Zornitza Bakalova for serving as a reviewer of the questionnaire for our interviews.

References

[1] Maiden, N., Jones, S., Agile Requirements: Can We Have Our Cake and Eat It Too? IEEE Software, 2010.

[2] Racheva, Z. and Daneva, M. and Herrmann, A. and Sikkel, K. and Wieringa, R.J. (2010) Do we Know Enough about Requirements Prioritization in Agile Projects: Insights from a Case Study. In: 18th International IEEE Requirements Engineering Conference, pp. 147-156. IEEE CS Press.

[3] Eckstein, J., Agile Software Development in the Large: Diving Into the Deep, Dorset House Publishing Company, 2004.

[4] Racheva, Z., Daneva, M., Sikkel, K., "Value Creation by Agile Projects: Methodology or Mystery?", Proceedings of PROFES 2009, 10th International Conference on Product-Focused Software Process Improvement, Oulu (Finland), June 2009, pp. 141-155

[5] Sjøberg, D. I.K., Dybå, T., & Jørgensen, M. (2007). The future of empirical methods in software engineering research. In L. C. Briand & A.L. Wolf (Eds.), Int. Conf. on Software Engineering/Workshop on the Future of Software Engineering. (pp. 358-378) IEEE CS Press.

[6] Cheng, B.H.C. & Atlee, J. M. (2007). Research directions in requirements engineering. In L. C. Briand & A.L. Wolf (Eds.), International Conference on Software Engineering/Workshop on the Future of Software Engineering. (pp. 285-303) IEEE CS Press.

[7] Auvinen, J., Back, R., Heidenberg, J., Hirkman, P., Milovanov, L., Software Process Improvement with Agile Practices in a Large Telecom Company. PROFES 2006: 79-93

[8] Elshamy, A., Elssamadisy, A., Applying Agile to Large Projects: New Agile Software Development Practices for Large Projects. XP 2007: 46-53.

[9] McDowell, S., N. Dourambeis, British Telecom Experience Report: Agile Intervention – BT's Joining the Dots Events for Organizational Change, XP 2007: Agile Processes in Software Engineering and Extreme Programming, pp. 17-23

[10] Sulfaro, M., Marchesi M., Pinna, S., Agile Practices in a Large Organization: The Experience of Poste Italiane. XP 2007: 219-221

[11] Koehnemann, H. M. Coats, Experiences Applying Agile Practices to Large Systems, AGILE, 2009, pp. 295-300.

[12] Gat, I., How BMC is Scaling Agile Development, AGILE'06, 2006, pp.315-320

[13] Larman, C., B. Vodde, Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-Scale Scrum, Addison-Wesley Professional, 2010.

[14] Ambler, S., Agile and Large Teams, Dr Dobb's, June 17, 2008.

[15] Yin, R.K., Case Study Research: Design and Methods (2008)

[16] King, N., Horrock, C. (2010). Interviews in qualitative research. Thousands Oaks, CA: Sage.

[17] Charmaz, K., Constructing Grounded Theory: a Practical Guide through Qualitative Research, Thousand Oaks CA, Sage, 2007.

[18] Runeson P., Höst, M., Guidelines for Conducting and Reporting Case Study Research in Software Engineering. Empirical Software Engineering 14(2): 131-164 (2009)Host and Runeson – add later.

[19] Agile Manifesto, http://agilemanifesto.org/ (last access: 10 Feb 2010)

[20] Daneva, M., Lessons Learnt from Five Years of Experience in ERP Requirements Engineering. RE 2003, IEEE CS, pp 45-54.

[21] Scheer, A.W., ARIS Business Process Modelling, Springer, 2000.

[22] Scholz, R. W. and Tietje, O. Embedded case study methods: Integrating quantitative and qualitative knowledge: Sage Publications, Inc, 2002.

[23] Grewal H.Maurer F., Scaling agile methodologies for developing a production accounting systemsfor the oil and gas industry. In: Proc.of AGILE 2007, IEEE CS Press, pp. 309-315.

[24] Sutherland, J., Viktorov, A., Blount,J., Puntikov, N., Distributed Scrum: Agile Project Management with Outsourced Development Teams, In: Proc of the 40th Annual Hawaii International Conference on System Sciences HICSS (2007), pp. 274a.

[25] Valade, R., The Big Projects Always Fail: Taking Enterprise Agile, In: Proc. of the AGILE 2008 Conference, IEEE CS Press, 2008, pp. 148-153.

[26] Bosch, J. Bosch-Sijtsema, P.M., Introducing agile customer-centric development in a legacy software product line, Software – Practice and Experience, 41, 2011, pp. 871-882.

[27] Hong, N, Yoo, J., Cha, S. Costomization of Scrum Methodology for Outsourced E-Commerce Projects, In: Proc. Of the 2010 Asia Pasific Software Engineering Conference (APSIC), IEEE CS Press, 2010, pp.310-315.

[28] Kendall, R.P., Mark, A. Squires S., Halverson, C. Condor: Case Study of a Large-Scale, Physics-based Code Development Project, Computing in Science & Engineering, May/June, 2010, pp. 22-27.

[29] Christou, I.T., Ponis, S.T., Palaiologou, E., Using the Agile Unified Process in Banking, IEEE Software May/June, 2010, pp.72-79.

[30] Shatil, A., Hazzan, O., Dubinsky, Y., Agility in a Large-Scale System Engineering Project: a Case Study of an Advanced Communication System Project, In: Proc. of 2010 IEEE Int Conference on Software Science, Technology & Engineeringg, IEEE CS Press, 2010, pp. 47-54.

[31] Gary, K., Enquobahrie, A., Ibanez, L., Cheng, P., Yaniv, Z., Cleary, K., Kokoori, S., Muffih, B. and Heidenreich, J. (2011), Agile methods for open source safety-critical software. Software: Practice and Experience, 41: 945–962. doi: 10.1002/spe.1075

[32] Urquhart, C. Exploring analyst–client communication: using grounded theory techniques to investigate interaction in informal requirements gathering. In: A.S. Lee, J. Liebenau and J.I. DeGross, Editors, Information Systems and Qualitative Research, Chapman and Hall, London (1997), pp. 149–181

[33] Martin, A., Biddle, R., James N., XP Customer Practices: A Grounded Theory. AGILE 2009: 33-40

[34] Martin, J.L., Clark,D.J., Morgan,S., Crowe,J.A., Murphy, E., A user-centred approach to requirements elicitation in medical device development: A case study from an industry perspective, Applied Ergonomics, June, 2010

[35] Baskerville, R., Pries-Heje, J., Madsen, J., Post-agility: What follows a decade of agility? Information and Software Technology, 53, 2011, pp. 543-555.

[36] Coleman, G., O'Connor, R., Investigating software process in practice: A grounded theory perspective, Journal of Systems and Software, 81(5), 2008, pp. 772-784.

[37] Rose, J., Pedersen, K., Hosbond J.H., Kræmmergaard, P., Management competences, not tools and techniques: A grounded examination of software project management at WM-data, Information and Software Technology, 49(6), 2007, pp. 605-624.

[38] 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.

.

Appendix A

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 manager	10	Face-to-face
Alpha	4	Business analyst lead	10	Face-to-face
Alpha	5	Business analyst lead (2x)	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

Appendix B

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	Types of changes	Participant P5 discussed how reprioritization happened in his experiences in project Beta.
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,*	Types of changes Client-triggered-change	Participant P5 explains a particular mechanism of how reprioritization is triggered, namely by means of ‘show-and-tell sessions’.
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.’*	Management-decision-type-of-change	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,*	Prioritization criteria	
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?’ 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 depth.* *We term it as technical depth. In a typical agile project what happens is; you start developing something and you deliver something at the end of*	Effort-as-Prioritization-Criterion Risk-as-Prioritization-Criterion	
the first iteration which typically is a short period of time, its one month	Effort-as-	Participant P5 elaborates on

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 depth. 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 depth 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.	Prioritization-Criterion Technical-Depth-as-Prioritization-Criterion	effort related to cost and time. Participant P5 explains what the term 'technical depth' and how depth is used in reprioritizing requirements.