Do we Know Enough about Requirements Prioritization in Agile Projects: Insights from a Case Study

Zornitza Racheva, Maya Daneva, Klaas Sikkel,
Roel Wieringa
University of Twente
Enschede, the Netherlands
e-mail: {z.racheva, m.daneva, k.sikkel}@utwente.nl,
roelw@cs.utwente.nl

Andrea Herrmann
University of Braunschweig
Braunschweig, Germany
e-mail: andrea.herrmann@tu-braunschweig.de

Abstract—Requirements prioritization is an essential mechanism of agile software development approaches. It maximizes the value delivered to the clients and accommodates changing requirements. This paper presents results of an exploratory cross-case study on agile prioritization and business value delivery processes in eight software organizations. We found that some explicit and fundamental assumptions of agile requirement prioritization approaches, as described in the agile literature on best practices, do not hold in all agile project contexts in our study. These are (i) the driving role of the client in the value creation process, (ii) the prevailing position of business value as a main prioritization criterion, (iii) the role of the prioritization process for project goal achievement. This implies that these assumptions have to be reframed and that the approaches to requirements prioritization for value creation need to be extended.

Keywords: requirements prioritization, agile development, value creation, exploratory case study;

I. INTRODUCTION

Agile software development methodologies become increasingly popular as the word spreads about the benefits they provide under certain project conditions. A key characteristic of any agile approach is its explicit focus on creating business value for the clients [1]. Essentially, in agile software projects, the development process is a value creation process that relies on active client participation. The value creation is ensured both through the final product as well as through the process itself. As previous studies show [10], the continuous prioritization of requirements during the project plays a central role in achieving value creation. Requirements (re)prioritization at inter-iteration time is the means to align technical decisions to the business strategy that drives the value. In accomplishing this purpose, the agile prioritization process assumes a number of conditions are fulfilled. Examples of these are the participation of the client at each decision-making step and the availability of knowledge about the value of single requirements. Yet, in the agile RE literature, very little is known about how agile (re)prioritization happens in practice and how the agile requirements mid-course decision process contributes to the client’s value creation.

In this paper, we present an empirical investigation of this phenomenon by means of an exploratory case study. Using a conceptual framework for agile requirements (re)prioritization that we developed earlier [29], we investigated real-world cases in eight companies. The overall research objective was to uncover how mid-course requirements prioritization takes place in industry and what notions of business value are included in it.

The case study is motivated by our previously published results [29, 30] from a systematic literature review on prioritization methods in agile projects. Our conclusion there was that the phenomenon of value creation through requirements prioritization was only partly explored. We had not found any study that clearly indicates how exactly individual agile practices or combinations of practices create value and keep accumulating value over time.

This paper is a step towards understanding how agile projects produce value to the clients or to the product owners through the requirements prioritization activity. We have set out to answer the following research questions (RQs):

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

RQ2: Which value-based criteria do companies use to perform value-driven decisions during agile prioritization?

RQ3: What are the characteristics of the project settings that are essential for the way a requirements prioritization process is carried out in a project?

RQ4: Are there any other ways (beyond the selection of requirements) through which the requirements prioritization process adds value to the project?

We answer these research questions by means of a multiple case study [37] in eight software developing organizations. The case study was exploratory because we had no pre-conceived ideas about the possible answers to these questions. Our purpose was to ‘look under the hood’, to observe and identify those mechanisms that drive the requirements prioritization as a value-creation process.

In the next section, we motivate this research in more detail and provide background on related work in the field of agile value-driven requirements engineering. Section III describes our case study research process and Section IV presents the results. Section V assesses our answers to the research questions and discusses implications for researchers and practitioners. Section VI analyses the possible validity threats, Section VII summarizes future research directions that we identified based on the case study and concludes the paper.
II. RELATED WORK

This related work section summarizes the state of literature with respect to the answers to our four research questions defined in the introduction. To the best of our knowledge, there is no systematic empirical research about how requirements prioritization is actually performed in agile projects and whether theory and practice always are consistent. Deviations of the real processes from the rules of agile development are rarely reported – what does not necessarily mean that they do not exist. Only in one paper [23] the authors observed that “Prioritization of the high-level goals was unclear and made it difficult to plan and organize development work based on business value.”

Agile approaches explicitly aim to deliver business value to the client early and regularly throughout the whole project [3, 7, 17]. In this way, the return on investment can be generated much earlier in the development process. A key practice of agile development which contributes to this early value delivery is the continuous and value-driven requirements reprioritization from customer’s perspective. A comparative study [8] of this process and the prioritization practices in ‘traditional RE’ indicates that, with respect to requirements (re)prioritization, agile RE is unique in two ways: (i) (re)prioritization happens at inter-iteration time, and (ii) (re)prioritization is mainly driven by business value, that is, the highest priority requirements get implemented early so that most business value gets realized, while exposing the project to as low a risk as possible.

Surprisingly, researchers in agile RE case studies found that the creation of software product value through requirements prioritization decision-making is only partly understood [6, 28].

**RQ1: Who are the decision-makers in the prioritization process? Which roles are involved and what are they responsible for?** The agile manifesto [2] deems the collaboration with the customer critical. XP [7], a prominent agile approach, recommends that the client is responsible for all business decisions, including requirements prioritization. Although decentralized decision-making involving all team members [10] is a guiding principle in agile development, it is the client who makes the final decisions. The customer is represented by a so-called ‘on-site customer’. In the decision-making process about requirements priorities, the development team takes the role of advisor by estimating cost and judging technical risk.

**RQ2: Which value-based criteria do companies use to perform value-driven decisions during agile prioritization?** Aurum and Wohlin [5] advocate a value-based approach, which in essence is about aligning clients’ requirements, business requirements and technological opportunities when making requirements prioritization decisions. For example, a recent study by Barney et al. [6] investigated the release-planning process to create software product value through requirements selection. These authors identified the factors that determine the decisions about inclusion of certain requirements for implementation. They are the client and market base of the software product, along with context factors such as maturity of the product, the marketplace in which it exists, and the development tools and methods available. We make the note that Barney et al. investigate market-driven development, while our study addresses custom-made development.

In our earlier systematic review [30], we found that agile literature sources agree that business value is considered the key requirements prioritization criterion in most agile projects. For example, the study of Cao et al. [10] concludes that “…agile RE practitioners uniformly reported that their prioritization is based predominantly on one factor – business value as the customer defines it.”

Some authors, instead of ‘value’ use the notion of ‘relative benefit’ [35], ‘contribution to user task or business goal’ [26] of a user story or of ‘high impact part of the business process’ [27]. Still, when reading about the application of the requirements prioritization method, it is our understanding that estimation of ‘value’ is the prerequisite for these prioritization methods.

There are two alternative ways of estimating the business value of a requirement – regardless the context of agile or of waterfall development: One can estimate either the value that the requirement provides to the system and its stakeholders if it is realized, or the loss it would cause if the requirement was not realized. For example, Wiegers [35] quantifies the “relative penalty which the customer or business would suffer, if the feature is not included”. Robertson and Robertson [32] in the VOLERE template estimate value and loss separately and anticipate both prioritization criteria: “customer satisfaction (which measures the desire to have the requirement implemented)” and “customer dissatisfaction (which refers to the degree of unhappiness if it is not implemented)”. Their project experience [32] shows that both values are different. This difference was investigated by one of the authors [18] by using mathematical representation and was explained by the dependencies among requirements. The principle of prioritizing requirements, based on what happens if it is not implemented is used mainly in the context of security (as by Arora et al. [4]). However, its use was successfully extended to cases of other non-functional requirements as well [15, 19] and even to functional requirements, as in Failure Mode and Effects Analysis (FMEA) [34]. We must note however that none of these authors who discuss the concept of loss by non-implementation of requirements stem from the agile community.

Next to business value, the agile literature suggests that when using some agile prioritization methods, the clients consider size/effort/cost, as well as risk caused by the implementation of a requirement [29]. Some (non-agile) authors emphasize that requirement dependencies, e.g. precedence constraints, influence both value and cost involved in the implementation of one requirement [14, 24], but are usually not taken into account explicitly by prioritization techniques [6]. We make the note that according to the agile literature [3, 7], size, effort, cost and risk are estimated by the developers and provided to clients for their decision-making. Size is based on the user stories (in XP and Scrum) or features (in Feature-Driven Development) and can, for example, be expressed in story points.
points [12]. Requirements are sometimes ordered according to a cost-benefit-ratio, or according to the extent to which they are ‘quick wins’ (meaning high benefit) and ‘low hanging fruits’ (meaning low effort).

**RQ3: What are the characteristics of the project settings that are essential for the way a requirements prioritization process is carried out in a project?**

The literature about agile development discusses two characteristics of the project setting which influence decision-making: change and project constraints. Change is explicitly expected and welcomed by agile development methods (“embrace change” [7]), or vice versa, agile approaches are chosen in contexts where change is high [26, 28] because they help to cope with this situation and apply strategies which reduce the cost of change [10]. Project constraints like limited resources and time pressure are typical for agile and non-agile projects. In larger projects, however, prioritization must be made on a higher abstraction level [26] than in small projects.

**RQ4: Are there any other ways (beyond the selection of requirements) through which the requirements prioritization process adds value to the project?**

The introduction of risk management in the development process and improving communication [26] are two of the motivations of agile and iterative development. The main types of risk which agile/iterative development intends to mitigate are change/ volatility and uncertainty [28]. Change means the introduction of new requirements or the change of existing ones, which can be caused by learning and by external change. Uncertainty can relate to instability of requirements or lack of technical experience, both of which lead to uncertain cost estimation.

The reprioritization of requirements before each iteration supports risk management by allowing decision-making to happen as late as possible, thus letting new information – through changes and learning – be considered in the decision. Volatile requirements are implemented later in order to reduce the cost of change [20], while technical uncertainty is reduced by implementing ‘worst things first’ (i.e. such which have a high effort estimation risk) like in XP’s Planning Game [7] and implementing first what supports learning [17].

All agile development techniques help to create opportunities for learning in the software process. These are – on different time scales – daily meetings like XP’s daily Stand Up Meeting or the Scrum Meeting/ Daily Scrum, continuous integration, daily build and regression testing, iteration planning – especially the reprioritization of requirements, sprint review meetings – including measurement of project velocity - and project retrospectives at a project’s end. Cohn [12] advises: “Incorporate new learning often, in order to decide what to do next” and “Incorporate new learning by prioritizing only as many features as can be completed in the coming iteration”. Furthermore, Larman [22] emphasizes that “Teams are organized by customer-centric features. A proper Scrum team is by definition a feature team, able to do all learning – individual and team learning increases because of broader responsibility and because of co-location with colleagues who are specialists in a variety of areas, critical for long-term improvement and acceleration”. For example, Harris and Cohn [17] present how they used strategic learning as the vehicle to minimize costs and maximize benefits and they provide guidelines on how to optimize business value.

External change, that is, a change in the project’s or company’s context, can influence the requirements’ value and priorities during the project. In the reviewed literature on agile RE however, we found only one explicit indication: competitive opportunities (i.e.: how do competitors perform with respect to a particular requirement) [13].

**III. THE RESEARCH METHOD**

We conducted a multiple-case study [37] to explore and explicate the decision-making process during a project in the context of agile projects and changing requirements. The case study consisted of semi-structured open-end in depth interviews with practitioners that work in organizations that develop software by using agile approaches.

#### 3.1 The case study process and participants

We executed an exploratory case study by performing the following steps:

- Compose a questionnaire;
- Validate the questionnaire through an experienced researcher;
- Implement changes in the questionnaire based on the feedback;
- Do a pilot interview to check the applicability of the questionnaire to real-life context;
- Carry out semi-structured interviews with practitioners according to the finalized questionnaire;
- Sample (follow-up with those participants that possess deeper knowledge or a more specific perspective).

The companies included in the study characterized themselves as organizations that follow agile methodologies. Some of them did strictly follow Scrum principles such as daily stand-up meetings and release retrospective. Most of them, though, applied a combination of agile practices without sticking precisely to a specific agile software development or project management approach.

Each interview was between 60 and 90 minutes long. Each interviewee was provided beforehand with information on the research purpose, the research process and the rights and responsibilities of the participating case study companies. At the meeting, the researcher and the interviewee walked through the questionnaire which served to guide the interviews. The questionnaire was composed of three parts: the first part discussed the prioritization practice that each interviewee experienced in one concrete project. The second part generalized the experience of the interviewees with respect to agile requirements prioritization processes across many projects, and the third part included questions related to the business value perception and value creation, as an essential part of the prioritization decisions. The rationale behind this structure was to focus the attention of the participants on a concrete example and then make a
transition to general observations drawn from participation in other agile projects. The third part aimed to clarify the considerations of value as a part of the prioritization decision-making process. We make two notes: (1) that no substantial changes in both the questionnaire and case study protocol took place after the pilot interview, so that the pilot interview could be considered part of the case study; and (2) that during the interviews there were cases when other questions arose next to those included in the questionnaire. These questions were not previously anticipated, however the researcher conducting the study considered them interesting and pursued the interview in that direction.

The study included 11 practitioners who described a total of 10 projects and who were working for eight different companies. The application domains for which these practitioners developed software solutions represent a rich mix of fields including banking, health care management, automotive industry, content management, online municipality services, and ERP for small businesses. In each organization we interviewed one or more representatives that were directly involved in the decision-making and the development process. Many of the participants performed multiple roles in the team and thus had a broad exposure to the whole process. The information about the participating companies and specialists is summarized below:

- 1 middle-sized company in the Netherlands (2 cases, 3 participants)
- 2 small companies in the Netherlands (3 cases, 3 participants)
- 1 small company in Bulgaria (1 participant)
- 1 middle-sized company in Bulgaria (1 participant)
- 1 German University (1 student project)
- 1 big consultancy in Italy (1 participant)
- 1 IT department in a big governmental organization in Turkey (1 participant)

Table 1 presents the participating companies with respect to the size of the client’s and developer’s organizations – we studied 8 companies and discussed the total of 10 projects, with 10 client organizations. Table 2 explains the primary role that the case-study participants had in the studied projects.

### TABLE 1. CASE STUDY SITES.

<table>
<thead>
<tr>
<th>Client’s organization</th>
<th>Small (less than 25 employees)</th>
<th>Middle-sized (25-200 employees)</th>
<th>Big (more than 200 employees)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Developer’s organization</th>
<th>Small (less than 25 employees)</th>
<th>Middle-sized (25-200 employees)</th>
<th>Big (more than 200 employees)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>4</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>

### TABLE 2. PARTICIPANTS IN THE INTERVIEWS.

<table>
<thead>
<tr>
<th>Interviewee’s primary role</th>
<th>Number of interviewees</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project Manager</td>
<td>5</td>
</tr>
<tr>
<td>Developer</td>
<td>3</td>
</tr>
<tr>
<td>Product Owner</td>
<td>1</td>
</tr>
<tr>
<td>Client</td>
<td>1</td>
</tr>
<tr>
<td>Scrum Master</td>
<td>1</td>
</tr>
<tr>
<td>Total Number of Interviewees</td>
<td>11</td>
</tr>
</tbody>
</table>

3.2 The data analysis

In our case study, the data analysis was guided by the grounded theory building method according to Kathy Charmaz [11], which is a qualitative method applied broadly in social sciences to construct general propositions (called a ‘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. It is driven by the desire to capture all facets of the collected qualitative data and to allow the theory to emerge from the data. In this method, the data is constantly compared to the previous data items, until a point of saturation is reached, where new sources of data don’t lead to a change in the emerging theory.

We first read the interview transcripts 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. ‘value’, ‘method’), or an activity (e.g. ‘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’. Other categories and, respectively, codes emerged during the coding process as a result of observations we didn’t anticipate and concerning concepts and aspects of the process we haven’t explicitly addressed in the questionnaire. Those are for example ‘problems that the developers encountered during the process’, ‘feeling about the way of work’, or ‘trade-off quality – schedule’. We clustered all pieces of text that relate to the same code in order to analyze it in a consistent and systematic way. The results of the data analysis are presented in the following section.

IV. 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?**

Our findings indicate that in our cases the developer plays a much more important role in practice than what is recommended in the literature. Overall, the practitioners agreed that the developers are active participants in the requirements decision-making processes. We observed the following situations: (i) the decisions were delegated fully to the developers; (ii) the clients required changes or faster implementation of certain functionality, without participating in other prioritization activities; (iii) the client participated during the project in a traditional way, e.g. by approving changes to budget. We provide some details about these cases in the following sub-sections. In only one of the investigated projects the client was the one driving the prioritization and making the decisions about the requirements that will be included for implementation in the next project iteration. In this project, the client had
significant previous experience in software development projects.

1) Strong developer’s participation

We observed that the participation of the developers in the decision-making processes is stronger in small projects, where the client is a small organization or company. First, such clients don’t possess knowledge in the IT domain and can’t afford paying extra for IT consulting services. They may even find it very expensive to allocate a resource to the role of ‘on-site client’. Often, it’s economically unfeasible for the client organization to pool away a full-time employee from their everyday business and task him/her to serve ‘on-site’ in an agile project. In such a context, it happens that the client delegates the decisions influencing the value creation, to the developing team. In one of the projects that we investigated the developers took over the decision-making because the client (car dealership) didn’t possess the time and abilities to analytically reason about the system he needed. This makes us think that there are certain constellations of contextual factors that will always lead to delegating the decisions to the developing organization.

Second, often the developers possess knowledge both in development and in the respective subject domain, as teams are specialized in developing a specific class of applications (e.g. banking, health-care, ERP, etc.). Some participants described situations where they had to ‘save the clients from themselves’, to prevent unwise decisions or suboptimal choices that will be harmful in long term. The experience from previous projects as well as the profound knowledge that the developers had of the client’s domain justified this course of action. “Also, when we know the goals, we can advise the client better, based on our past experience from other projects. For example, if they want something that we know from past experience that it didn’t work.” The developer was led by the desire to create maximal value for the client, to contribute to a successful project. In the experience of our interviewees “this leads to high client satisfaction and good relationship with the client, which will, eventually, lead to future mutual projects”.

Our case study participants indicated with certainty that a high level of trust is a prerequisite for such cooperation. One project manager reported: “Customer’s relations are essential; we want to make them happy but we don’t just do whatever they ask for. Instead, we try to understand what their problems are, and their domain, so that we (the application-builders) can better serve their needs, for example by smart architectural decisions. They gain as well from this approach…”

In our case study, the interviewees went further to explain why developers are so strongly participating in the decision-making. In their view, the developers’ company is the one to make sure that the project delivery process runs in a way that is profitable for the company. If developers accommodate all wishes which clients might come up with at inter-iteration time, the company may find it not sustainable in the long run. This observation raises the question about value considerations for the developers, discussed in detail in [31].

The matter that developers strongly participate in the prioritization and decision-making gives us the hint that agile and traditional requirements engineering processes may not be that different regarding who prioritizes the requirements.

2) Client’s participation

Our interviewees indicated that the client’s judgment of importance regarding a specific requirement might not always be representative for the client’s organization as a whole (despite the matter that the client is supposed to represent the organization and speak on behalf of all users). In such situations, it might well be the case that the client consciously or unconsciously manipulates the developers to implement specific requirements. The developer has no possibility to collect more objective information about the situation and to judge the extent to which s/he could trust the client’s sense of priorities. One participant reported her experience in a case of a client who asked for a certain report. According to the client, this report was ‘very important’. It took significant efforts on the developers’ side to prepare it. Later it turned out, that this client’s representative was the only person in the whole company reading this report.

4.2. Value-based criteria

RQ2: Do companies use value-based criteria to perform value-driven decisions during agile prioritization?

While the concept of business value was deemed important to all participants, when it comes down to making requirements prioritization decisions at inter-iteration time, we found a surprising result: almost all participants stressed the importance of, what they called, a ‘negative value’. In their requirements prioritization experiences, the line of reasoning that prevailed was not concerned with how much value a certain feature would add to the product, but instead with the question of how much it would detract from the product’s value, if the developers would not implement this feature. (The amount value detracted was termed ‘negative value’). The negative value thus is equivalent to loss of value or damage to the business. If something important is missing, its value is not zero, but it has a negative value. This is linked to damage for the business or impossibility to support business needs or processes. As one participant put it “…the question was always asked: how big is the damage if we don’t have this or that requirement; how complicated would it be to use a work-around?…”

In the experience of one practitioner, this line of reasoning reflects a professional pragmatic behavior especially in project contexts where the resources are very limited and the client is concerned about whether or not s/he derives maximum benefit from the project. This phenomenon, though, seems to be observable not only in small projects. One of the study participants reported about a project where a big client (a bank) demanded urgent implementation of additional requirements and re-scheduling of the project, because the losses the bank suffered from the lack of this functionality were bigger than the extra costs
involved for development, re-scheduling and postponing the implementation of other functionality.  

First, the concept of negative value suggests that we may need to redefine the concept of business value in agile all together. Second, this concept sheds light on the type of value that serves as input into the decision-making processes in agile projects. We want to make a note that in Section II we found that negative value has been discussed by non-agile authors so far and exclusively by non-agile authors.  

Next, we observed that there is a link between the perception of value and the price for the implementation.  

The clients want to see a link between the perceived value of a feature and the cost of implementation. They don’t always understand the relation between the effort estimations of the developers and the perceived amount of work (from the viewpoint of the client). For example, a client wanted an additional button. Estimations and analysis of the system architecture showed that this one button will require significant changes and was estimated at 20 hours of work. The client could not understand the rationale behind this estimation and debated it. The developer decided to look for a work-around to implement this button in a more time-efficient manner and thus please the client. Similarly, in another project the developer prepared a demo to better understand the client’s needs. The client, on his turn, could not understand why he has to pay half a million for this one week of work, as the difference between the real system and the mock-up was not clear to him.  

Last, the value-creation process plays an important role for the developers’ organization, not only for the client’s company. The agile practitioners’ literature [3, 7] seems to share the opinion that the only value-creating considerations that drive the development decisions are those of creating value for the client. During this study we made the consistent observation that, more often then not, the value creation for the developers has been considered as well. (We note that the topic of the understanding about business value and the RQ2 are discussed in greater detail in [31]).  

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

RQ3: What are the characteristics of the project settings that are essential for the way a requirements prioritization process is carried out in an agile project?  

We found that the requirements prioritization processes vary regarding the forms of clients’ participation and collaboration in the process. In the following subsections we discuss two context factors, which seem to be responsible for this variation. Those are: size of the client’s organization, and size of the project in terms of resources (budget and time).  

1) Impact of the size of the clients’ organization  

The observations concerning the size of the client’s organization and the process are summarized in Table 3. First, in all cases where the client was a small company, except one, the decisions were delegated completely (in 2 cases), or mainly (1 case), to the developer. The special case was a small academic project where the professor served as a project manager and as a client. In this case the client considered herself experienced and led the whole process. In the other three cases, the developers took up the responsibility to oversee the work process and provide a solution. As one participant put it: “…we tried to look from the perspective of different users in order to understand what they need first…”  

<table>
<thead>
<tr>
<th>Size of clients’ organization</th>
<th>Process specifics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small client company</td>
<td>The client can’t allocate resources for participation and in the most cases does not possess the knowledge needed.</td>
</tr>
<tr>
<td>Middle-sized client company</td>
<td>Client’s cooperation is limited, don’t allocate resources, don’t agree to client on site or even to developer on site. The behavior changes only after few iterations where they see the benefits of the agile methodology.</td>
</tr>
<tr>
<td>Big client (e.g. banking sector)</td>
<td>The relationship becomes more strictly defined, changes require participation of higher-level management.</td>
</tr>
</tbody>
</table>

Second, in the two cases which we observed to have middle-sized organizations as clients, there were serious problems with the client’s participation. In one of these two cases (namely, in a municipality), the client did not want to provide resources for either a ‘client on site’ or a ‘developer on site’. The project manager that we interviewed even went on writing to ask an advice (what to do in such case) from S. Ambler, a prominent agile practitioner who authored a large number of publications on agile practices. The answer was: “you have to have on-site client.” In the other case of a middle-sized organization, there was a tension between the product owner (PO) (that is, the client’s representative from the developing organization) and the developing team, as the PO persistently wanted the continuous implementation of new functionality without considering the technical feasibility.  

In the middle-sized organizations we observed that the client’s behavior eventually changes during the project, but no earlier than when a few iterations pass and the benefits of the agile approach become visible. A similar situation was observed in two of the big organizations. One project manager reported that: “…Now a ‘we’ feeling emerged, in contrast to the polarized ‘them-us’ attitude from the beginning of the project. Now we started doing the prioritization together and they learned how to prioritize…”  

Third, in all cases of big clients, we observed a more strictly defined decision-making process, including multiple levels of client’s involvement, reaching up to the higher management when it comes to decisions concerning the project development. The big clients were four in total: three were in the financial sector and one in the automotive sector. In essence, all four projects deployed a requirements prioritization process that was a blend between traditional
and agile. Our observations that a blended process is used in large organizations converges with reports in the literature on hybrid methods [16].

2) Impact of the factor resources

When it comes to resources, we observed that small projects with very limited resources aim at creating a product with a certain minimum of functionality that is absolutely necessary in order to satisfy the client’s needs. From the 10 projects we investigated, five were with strictly limited resources: in terms of time (one academic project), schedule (two projects), and money (two projects). Out of the five projects, four projects had small organizations as project clients, and one project had a middle-sized company as its client. In four of the projects the order of implementation of the features was almost irrelevant to the value creation because the projects were relatively short (2–5 months). In these four cases, of paramount importance was a minimum of functionality that needed to be implemented within the available resources. Without this functionality the product would be of no value (or very low value) for the clients. Table 4 captures our findings in this respect.

<table>
<thead>
<tr>
<th>Project resources</th>
<th>Process specifics</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very (and strictly) limited resources in a small project</td>
<td>Necessary minimum of functionality is absolutely required by the end of the project. Prioritization serves to choose those requirements that are crucial for supporting the main goal of the client. (This is often coupled with choices about implementation solutions).</td>
</tr>
<tr>
<td>Bigger project where additional resources can be considered</td>
<td>The prioritization serves to choose the highest value requirements for the next iteration.</td>
</tr>
</tbody>
</table>

The observations in subsections 1 and 2 suggest that the interplay between both factors represents a specific project constellation that requires (or pre-determines) the instantiation of the prioritization process.

4.4. Role of the prioritization process

RQ4: Are there any other ways (beyond the selection of requirements) through which the requirements prioritization process adds value to the project?

The use of the prioritization in agile context is not limited to selecting the most important/valuable requirements for the upcoming iteration. Our study revealed two other aspects that are very important for the project’s outcome: building the right product and incorporating new information and learning on-the-fly. Our participants indicated that in a context of volatile or unclearly defined requirements, the prioritization process ensures value by the change management mechanisms and by incorporating learning loops in the process. As a participant observed, “… if we would not have followed this approach, we could have made a completely different system from what they want. Especially in this case where the requirements were not SMART… ” Furthermore, the participants agreed that the agile prioritization process provides a natural way to react to new information and knowledge that become available in the course of the project. This might be a deeper insight about the requirements, or new knowledge about technical feasibility. For instance, in one of the projects in our study the initial planning was made with certain assumptions about functionality provided by open-source libraries. In the course of the project it became clear that these assumptions do not hold and the scope of the project was changed, and, respectively, the requirements were re-prioritized. Another participant reported that the requirements became more clearly defined at later stage in the project.

V. DISCUSSIONS OF THE RESULTS

We were surprised that our exploratory case study yielded a few findings regarding essential aspects of requirements (re)prioritization in agile projects, which deviated from what agile literature says about these aspects. Overall, these findings revised parts of our understanding about the roles of the client and developer, and the prioritization criteria used.

(i) With respect to the process aspects clients’ participation and collaboration, the agile literature [3, 7] suggest that the following assumptions are always true: (1) the client always participates and provides input about the value of each requirement: the client is on-site, is competent, is able to resolve issues rapidly and provide the developers with timely information about the client’s desires and especially about the value created by each requirement; (2) the client is the one taking the final business-related decisions and in particular the prioritization decisions, based on business value; (3) the developers proceed with the implementation based only on whatever the client desires, i.e. the value-creation for the client is driving the process.

The results of this study suggest that these assumptions are not present in all projects. First, having an on-site client was not always feasible. Whenever it was feasible, we found that developers may have to work with a client that for various reasons may not be in a position to provide timely information about requirements priorities. The detrimental impact of involving such a client in the requirements prioritization process may vary. Second, the evidence from the study shows that, in contrast to the documented best practices in the agile literature [12], in most of the cases the developers did the inter-iteration decision-making. Our interviewees agreed that more often than not the involvement of the clients consisted mainly of approving the plan/giving comments. Only in few cases practitioners were able to provide evidence that the client is really capable/interested/aware of the agile way of defining priorities, and thus able to navigate the functionality by the mid-course decision-making process. Why is this so? One possible explanation is that the clients in general have little, if any, understanding of the agile process and their role in it. Another reason could be that the client is not aware of his/her needs or can’t express them in a way that can drive

...
the developing process. In some cases, notably with small companies, the client doesn’t have the resources to get deeply involved and has no alternative but to trust the developer to do a good job. Particularly poignant examples of this are cases where the client might want something that has proven not to work, but is not aware of that, and the developer gives advice based on his/her experience.

Last but not least, the client might have no experience whatsoever with IT solutions. All these reasons call for delegating the decision-making to a big extent to the developer. As one of the study participants explained, they invited a client’s representative to one of the iteration planning meetings and this turned out to be too technical, that’s why this experience was not repeated.

Our study suggests that in agile (as well in traditional) contexts, we can find examples of clients who essentially rely on developers to prioritize their project requirements; we, therefore, think that the difference between agile and traditional processes is not in who prioritizes the requirements, but: (1) in the competencies and (tacit) knowledge that those who prioritize are required to have of their client’s business, and (2) in the ability of the client to participate in the process. Our interviewees suggested that the developers who ‘saved the clients from themselves’ are experienced professionals (e.g. in the words of one interviewee, with 10 to 15 years of experience in IT systems delivery in a specific business sector) and this might indicate that for agile prioritization to be led by developers, it should include highly-competent and experienced people. And people willing to enter new knowledge domains in order to assist their clients and better support their process.

(ii) With respect to the value-based decision-making we observed that the consideration of value as a prioritization criterion is complex. The concept of negative value plays a significant role in some of the investigated projects, and to the best of our knowledge this has not been discussed previously in the agile literature (see section II). Further, there is a difference in the perception of value for the developers and clients, and the value creation for the developers impacts the input from the developers, and thus – the decision-making process. Clearly, developers and clients have some goals that ensure mutual benefits to incur e.g. “we want to make the client happy, so that he/she comes back”, while other goals on the developers’ side may not be related to one particular project or one particular client, and instead are related to issues like reuse, other concurrently running projects and distribution of resources for maximizing value for the organization.

We need to consider more carefully in which ways development teams balance the client’s business value with their own organizational bottom-line. We consider this an important topic for future research on its own right.

Last, the observation that clients have limited knowledge of requirements priorities when it comes to inter-iteration decision-making opens up a question to the researchers engaged in developing and evaluating requirements prioritization methods. The existing prioritization techniques rest on the assumption that clients are aware of the mechanics behind the application of requirements prioritization techniques and, as a minimum, they are conscious about their role of providers of the input that feeds into these techniques. Our case study findings undermine the realism of this assumption. Indeed, requirements prioritization methods take for granted that there are objective values to provide inputs into the methods. Our case study findings give rise to the suspicion that these objective values may not always exist and are sometimes very difficult to make. This presents a limitation to the requirements prioritization methods currently documented in the literature. Therefore we think that future research is warranted to understand those cases in which this assumption is not realistic.

VI. Threats to Validity

We evaluated the possible threats to validity of the observations and conclusions in a case study [11, 33, 37]. For this purpose, we used the checklist for case study researchers recommended by Runeson and Höst [33]. First, we planned and conducted exploratory qualitative research. Therefore, to evaluate the validity of its results, a key question to address is: to what extent can the practitioners’ experiences and the requirements prioritization mechanisms we observed, be considered representative for a broader range of companies? [33, 37] In the study, we deliberately included representatives of companies of different sizes, business sectors and geographic location, as well as different stakeholders from projects with different sizes. Based on our observations, we think that the conclusions would hold for other companies in contexts similar to ours (company size, project size, and geographic location). This was confirmed by participants in a panel discussion on the first results of the study. Regarding company’s location, our case study participants deemed the geographic zone, where a company is located, indicative for the presence of a country- (or zone-) specific culture that pre-defines the relationships between the clients’ and developers’ organizations. For example, there are country-specific legal aspects that favour certain contractual terms to prevail in agile projects. Furthermore, the geographic location of the company predisposes it to certain ways of using the agile methodologies and principles (based on how broadly the agile philosophy has penetrated a specific part of the world). We expect that the results of our study are generalizable to companies in a similar context, e.g. located in Central and South-European countries. We make the note that the level of penetration of the agile paradigm differs in the different geographic zones. E.g. in North America, the increased awareness and usage of agile approaches has led to the formation of professional communities and networks as well as to specialized professional certifications (e.g. scrum master). Therefore, we expect differences regarding the application of agile software practices at companies in different cultural settings.
Second, we acknowledge the possible risks that we might have actually observed relationships in the cases that are not really there (e.g., are the projects we sampled indeed agile?). We took two steps to minimize the effect of this validity threat: (1) we confirmed with all case study participants that they did apply an agile methodology; (2) during the interviews, we consciously watched for confirmation of whether the interviewees indeed referred to examples of their experiences in agile projects (and not in projects that used other approaches). We acknowledge that the questions of where the line is between agile and other iterative and incremental approaches and which characteristics of the project should be observable in order to deem a team or a project agile, are philosophical in nature. These questions are, however, out of our paper’s scope. To the best of our knowledge, the projects we investigated in this study are agile projects in the sense of the Agile manifesto [2].

Third, our interviewees agreed that the agile process helped them create rapport with their clients easier than it could have been possible in a project that uses a ‘traditional’ delivery approach. As they put it “Making the clients happy was much easier than ever before.” We deem these observations biased, as they are provided by agile practitioners only. However, we make the note that more than 70% of the interviewees were seasoned practitioners with more than 15 years of software project experience and that they got a large part of this experience while working for organizations with traditional development approaches. So, they had enough practice to compare the both worlds. However, we admit that there is a possible threat to validity as we interviewed only those people that are currently engaged in agile development. Theoretically, there could have been professionals that left agile and got back to traditional approaches. However, searching for such practitioners to involve in the case study was not a feasible option within the resources we had.

VII. CONCLUSIONS AND FUTURE WORK

This work investigated the practical application of prioritization and business value delivery processes in eight software organizations. The study revealed an important gap between the realities of the practitioners and the assumptions made in agile requirements engineering literature. We found that three explicit and fundamental assumptions of agile requirement prioritization approaches, as described in the agile literature on best practices, do not hold in all agile project contexts in our study. Those are: (i) the driving role of the client in the value creation process, (ii) the predominant position of business value as a main prioritization criterion, and (iii) the role of the prioritization process for project goal achievement. We can conclude from our case study the following points: First, while an agile software company lets clients prioritize requirements, the requirements decision-making process can take place only when the client’s interest to make changes along the way is in balance with the developer’s interest for a sustainable business. Second, the existence of objective values to feed as input into the prioritization methods is questionable; instead, what is priority seems to be a combination of subjective value-based criteria. Third, the prioritization process instantiation varies across projects at different client companies and those variations seem to be linked to project characteristics such as size of project and size of clients’ organization.

An implication of the gap that we identified is that we have to reframe these assumptions and to extend the existing approaches to agile requirements prioritization for value creation. More in detail, our reflection on the gap brought us to the following research questions for the future:

1) What kind of developer’s assumptions do agile clients need to be aware of and what kind of clients’ assumptions do developers need to be aware of, in order to enhance the value-creation process?

2) How to reframe the prioritization methods to reflect the reality we observed?

3) In which project contexts are we likely to observe that the assumptions are not realistic? To identify and better understand those cases.

Our next step will be to identify lessons learnt that possibly suggest adaptations in the agile requirements prioritization process, so that both clients’ and developers’ organizations profit. For example, extensions that ensure a better understanding for the rationale behind the decisions and enhance the value-creation throughout the project. Further, understanding the mechanisms that condition the use of negative value in agile practice can bring us to restate the decision-making conceptual frameworks which we, the researchers, have been using to explain agile phenomena until now.

We think that if we collect and analyze examples of good and not-so-good ways to balance clients’ and developers’ value-creation perspectives, then we will be able to deduce patterns, principles, do’s and don’ts, and other general understandings that help practicing requirements engineers build up a body of knowledge that can assist them in value-creation. The mechanisms that are at play in contexts where this balance is feasible remain to be uncovered. We think, therefore, that understanding the variation in project contexts is needed, and also understanding the use of the different decision-making approaches, based on relevant context characteristics. We consider it work-in-progress that the community may want to adapt and expand.

ACKNOWLEDGMENT

We are indebted to all case study participants for sharing their experience and thoughts with us. Further, we want to thank the anonymous reviewers for the valuable comments and improvement suggestions.

This study was supported by the Netherlands Organization for Scientific Research (NWO) under the Quadread research project.

REFERENCES


