A Conceptual Model and Process for Client-driven Agile Requirements Prioritization

Zornitza Racheva, Maya Daneva
University of Twente
Enschede, The Netherlands
{z.racheva, m.daneva}@utwente.nl

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

Roel J. Wieringa
University of Twente
Enschede, The Netherlands
roelw@cs.utwente.nl

Abstract—Continuous customer-centric requirements reprioritization is essential in successfully performing agile software development. Yet, in the agile RE literature, very little is known about how agile reprioritization happens in practice. Generic conceptual models about this process are missing, which in turn, makes it difficult for both practitioners and researchers to reason about requirements decision-making at inter-iteration time. This paper presents a Grounded Theory study on agile requirements prioritization methods to yield a conceptual model for understanding the inter-iteration prioritization process in terms of inputs and outcomes. The latter is derived by using qualitative empirical data, published earlier by other authors. Such a conceptual model makes explicit the concepts that are used tacitly in different agile requirements prioritization methods and can be used for structuring future empirical investigations about this topic.

Keywords - *Agile development, requirements prioritization, inter-iteration decision-making process, grounded theory*

I. INTRODUCTION

Continuous and value-driven requirements reprioritization from customer's perspective is key to the successful execution of agile software projects. A comparative study [10] of this process and the prioritization practices in "traditional RE" indicates that, with respect to requirements (re)prioritization, agile requirements engineering (RE) is unique in two ways: (i) (re)prioritization happens at inter-iteration time, and (ii) (re)prioritization is based mostly on business value, that is, the highest priority features (i.e. requirements in agile terminology) get implemented early so that most business value gets realized, while exposing the project to as low a risk as possible. Surprisingly, researchers [8,35] in agile RE case studies also found that the creation of software product value through requirements prioritization decision-making is only partly understood. In the current agile RE

literature, a generic conceptual model describing requirements reprioritization in agile development is missing so far.

Our paper makes a step to contribute to the understanding of agile requirements reprioritization at inter-iteration time. For doing so, we propose a generic model of agile techniques of requirements (re)prioritization from a client's perspective. This model is independent from any particular method, that is used, and it can describe on an abstract level the agile prioritization techniques. We obtained this model by literature research applying a grounded- theory-based approach. We do not provide a new prioritization technique, but instead present the state of the art described by concepts that we discerned from reading relevant publications. We expect that – after the validation of its completeness - the model can help practitioners in their decision-making process. We, as researchers, plan to use this model as a conceptual structure for planning and evaluating further empirical investigations on requirements (re)prioritization during agile development.

The paper is structured as follows: Section II presents our motivation for this research and the role of clients in agile RE. Section III introduces the Grounded Theory research method. Section IV describes our application of it and the results of this application. Section V and VI reflect on our results and discusses possible threats to their validity. Section VII concludes the paper.

II. BACKGROUND

A. Motivation

Our motivation for creating a model of agile requirements prioritization (RP) from a client perspective originates from that the practices of continuous requirements reprioritization, with strong client participation, are a relatively recent phenomenon and, consequently, are only partially understood. As the agile

literature [1,7,10] indicates, never before in the software engineering history, the client has been that actively and constantly involved in the requirements reprioritization as he/she is in agile. When the client is expected to actively participate in the process by performing, among other task, the key task of prioritizing requirements, he or she must be aware of the facets of his/her role and thus would profit from a clear model of the prioritization process available at his/her disposal. We think that a conceptual model can help the client in multiple ways: (i) to navigate through the agile process of delivering business value; (ii) to make explicit the assumptions that are used tacitly in different RP methods; (iii) to identify those possible pieces and sources of information important to the outcome of the prioritization and, consequently, to the project; (iv) to make the process more objective in the sense that having such a vehicle will allow also less experienced users to participate in the prioritization process and, they could do it with the confidence that they deliver a quality work. We also think that our model would help those RE researchers who may be interested in carrying out case studies to investigate how agile requirements decision-making happens in practice.

B. The client-centric agile RP process

The agile manifesto [36] deems the client's role critical in making decisions about "what to build". XP [9], a prominent agile approach, recommends the following for the client's role: (1) The client is an integral part of the team and should be on-site with the team. (2) The client writes user stories and then discusses each requirement directly with the programmers. (3) The client is responsible for all business decisions, including RP. (4) The small 2-3 week iterations allow the clients to evolve their requirements based on concrete working software. (5) The client regularly tests the software to confirm it works as expected. Although team work is a guiding principle in agile development, it is the client who makes the final decisions. In the decision-making process about requirements priorities, the development team takes the role of advisor by estimating cost and judging technical risk.

Our focus in this paper is on item 3 of the above list. Clearly, RP is a part of any project, independently from the development method. Yet, the purpose of RP, its place in the project life cycle and the role of the clients in it are essentially different when we distinguish between 'traditional' and agile development. In a 'traditional' (e.g. gated or waterfall-style life cycle), it is about which requirements (i) to implement earlier than others, or (ii) to include in an earlier iteration or release. The premise is that the whole functionality can not be implemented at the same release, but it will eventually be implemented. So it is a project-management activity from the developers' side. When asked about priorities in a 'traditional' project, the client tends to qualify the majority of the requirements as high priority.

In contrast, agile projects rest on the understanding, that the whole functionality as specified at the start will not be implemented and delivered at once with the first release, and part of it may never be implemented at all, because changes and learning are allowed. To illustrate how agile projects proceed, we describe below an example of how Scrum [29,39] treats requirements prioritization. Scrum is an iterative and agile incremental process model including values, artefacts, roles and meetings. The main roles in Scrum are:

- the "Scrum Master", who ensures that the Scrum process is used as intended and who enforces the project management rules;
- the "Product Owner", who represents the stakeholders;
- the "Team", a cross-functional group who perform the work activities as the actual analysis, design, implementation, testing.

The project starts with a product backlog which is an initial requirements list and is prioritized by business value. It also contains rough estimations of development effort. Business value is set by the Product Owner. Development effort is set by the Team. The project is executed in time units of iterations (the so-called "sprint") which are two to four weeks long. Each iteration starts with a sprint backlog which contains only those requirements which are to be implemented during this sprint. We note that this sprint backlog is frozen and not modified until the sprint is over. This means that (i) re-prioritization takes place during the sprint planning meeting at the beginning of each sprint only, and (ii) after this point of time no re-prioritization takes place during the daily Scrum meeting. At this meeting, business values and development effort of the requirements are re-estimated and the sprint backlog for the next sprint is created. At the end of a sprint cycle, two meetings are held: the "Sprint Review Meeting" (where the completed work is presented to the stakeholders) and the "Sprint Retrospective" (which serves the objective to make continuous process improvements).

In agile development, each iteration is treated as a timebox with a fixed duration. The iteration planning, then, means: (i) how to decide on what to implement in each (next) iteration, and (ii) which requirements will deliver the maximum value to the clients as early as possible while minimizing project risk and respecting the project constraints. The decision-making process implementing the principle "business value first" is as follows: One starts with the requirement deemed to have the highest business value, and then checks whether it is within the (iteration) budget. If yes, one continues with the second-highest value requirement. Is it also within the budget? If so, one continues. If no, one stops. In XP's prioritization procedure – the so-called Planning Game – one complementarily applies the "business value first" principle and the "worst things first" principle [9] which

means that requirements involving high technical risk should be implemented early. The rationale behind this is risk reduction: the earlier one implements the riskiest requirement, the earlier one gains more certain knowledge about this requirement's implementation effort.

One of the biggest assets of an agile approach is that business value is delivered to the client early and regularly throughout the whole project, and the return on investment is generated much earlier. Thus, any changes in the requirements can be accounted for and built into the product at an early stage. This highlights the paramount importance of the RP activities. Moreover, in contrast to traditional projects where the RP is usually performed once and before the implementation phase, in agile context, RP is an ongoing process happening at each iteration's start. This reflects the dynamics of the product backlog. The client is the key driver of this process, being supported by a project manager (called, for example, a 'scrum master'), while in the traditional development this task is performed by the developer, with the participation of a project manager and other stakeholders.

Building on the above discussion, we came up with the following research question (RQ): "What are the key concepts to consider when prioritizing the requirements from client's perspective in agile projects?" To answer it, we apply a grounded-theory-based research method [13].

III. THE GROUNDED THEORY APPROACH

The term Grounded Theory (GT) [13] refers to a set of systematic guidelines for data gathering, coding, synthesizing, categorizing, and integrating concepts to conduct a theoretical analysis of empirical data. The name 'Grounded Theory' points out the fundamental premise that a researcher can and should develop theory from rigorous analyses of empirical data. As a qualitative research method, GT is distinctive in (i) that it is inductive in nature, which means that we as researchers have no preconceived ideas to prove or disprove data, (ii) that collection and analysis proceed simultaneously and each informs the other, and (iii) that constant comparative techniques treat possible disagreements between the emerging theory and newly collected information. A GT exercise of a studied topic starts with a clear research question and with concrete qualitative data, e.g. observations, interviews, experiences and all kinds of documents. It ends with rendering them in a descriptive theory describing the mental model that people have with respect to the answer of the research question. From the very beginning, the researcher analyzes the data and identifies analytic leads, tentative categories, and causal dependencies to develop through further data collection. It is essential to note that, in this iterative process, whenever the emerging theory disagrees with newly collected information - be it from observations and experiences collected by the researcher or from literature, the researcher seeks to extend the theory so that it makes sense of both the new and former data. The key concern

throughout a GT process is the fit of the theory to the data and its ability to make sense of actual experience.

Research methodologists [13,20,40] suggest that a theory derived from data is more likely to resemble what's happening in reality, than a theory which is derived by putting together a set of concepts based on experience, existing theories and solely through ungrounded assumptions about how things in real life would work. The philosophical foundation of GT and how it affects the researcher's choices in carrying out his/her work have been discussed in [13] and are beyond the scope of this paper. Here, we focus on the application of the GT process and the results we obtained.

IV. OUR APPLICATION OF GROUNDED THEORY

For the purpose of our research, the first three authors used the GT guidelines by Kathy Charmaz [13] in order to generate a conceptual model which describes the answer to our RQ: "*What are the key concepts to consider when prioritizing the requirements from client's perspective in agile projects?*" Our model is based on published descriptions of agile prioritization techniques and of case studies. We executed a research process which included the following steps: (1) identification and review of data sources from published literature, (2) initial and focused coding of the concepts that play a role in agile prioritization, (3) clustering of those concepts, (4) conceptual modelling, and (5) theoretical sampling of empirical data, using the concepts from our resulting conceptual model. The goal of steps 1-3 is the discovery of as many relevant categories as possible, including their properties. Step 4 is about the visual representation of the categories and their relationships, and Step 5 is about 'saturating the categories'. Categories are considered 'saturated' when collecting fresh data no longer brings new theoretical insights nor reveals new properties of the categories in the conceptual model.

We traversed the steps 1-5 multiple times, as methodologists recommend, because: "Constant interplay between proposing and checking [...] is what makes our theory grounded!" [40]. That means, the analysis of the data collected in one step helps to check the interpretations from the previous step.

The sub-sections below indicate the execution of the steps along with the results we obtained from our application of GT.

A. The sources

In this study, the data used and constantly compared to the emerging theory is literature on agile RP available in digital libraries and prominent agile practitioners' journals. To search for relevant papers, we used five bibliographic databases: IEEEExplore, ACM Digital Library, Google Scholar, InterScience and CiteSeer. We additionally included three periodicals: the Agile Journal [1], and the platforms dedicated to software development and agile

methods: Dr. Dobb's [17] and InfoQ [26]. Our strategy for searching literature sources in these databases and periodicals included (1) the use of subject keywords and (2) the use of citations. To define the search strategy we implemented the recommendations of Webster et al [42] for searching in literature databases.

First, the key words we used for our search were: *agile* plus two out of the following: *requirements*, *backlog*, *prioritization*, *inter-iteration*, *decision-making*, *business value*, *risk*, *cost*, *features*. We deliberately did not include any literature sources on non-agile iterative development methodologies, as RUP, for example. Second, we traced the references in the identified papers to get access to other relevant sources.

To determine whether to include or not these sources to our GT research, for each one, we reviewed the abstracts and the conclusions and we checked this information against the following four quality criteria for inclusion in the review: (1) the paper is on a agile RP, (2) the paper is credible, i.e. the method described is meaningful and intuitive to follow; (3) relevance for practice: the RP method potentially offers support for practical RP; (4) the paper adequately describes the context, in which the method is expected to be applicable; 'adequately' means that the reader can judge about the use of the requirements prioritization method in his/her own context. The publications were written in English only and included qualitative research papers as well as experience papers, from scientists and practitioners. We carried out the quality check by using these criteria, which yielded 42 papers eligible for inclusion and review in the GT process. These papers refer to 19 RP techniques, as indicated in Table I.

For the reasons discussed in Section VI, we analyzed additional books about agile development. [11,24,25,32] since the GT methodologists [13,40] suggest that the GT approach can be applied to data obtained by different means, and even to heterogeneous set of data, we refer to the sources used here as the primary sources for our study.

Before proceeding to the next section (which is about the results of the analysis of the information in the relevant sources), we would like to make the note on our combination of the GT approach and the strategy for searching literature sources in bibliographic databases.

While the search strategy helps us find the body of knowledge on a certain topic (namely agile RP in this case), the GT serves to build a theory [13,40]. Here, the data, retrieved in the source identification step, serves as an input to the actual GT process – i.e. these data has been used to uncover, understand and explain the phenomenon of agile RP, and to order the findings in a conceptual model.

Last, we also make the note that our approach is not about carrying out a systematic literature review [28]. While our approach does include the definition of a search strategy, which is also common in the approach of

systematic literature review, there is an important difference regarding the way we treat the data analysis part of our research process. In contrast to systematic reviews which deploy specific data aggregation and data synthesis techniques [28], our approach uses the coding and constant data-comparison techniques which are characterizing features of the GT method.

TABLE I. THE RP APPROACHES PUBLISHED IN THE SOURCES USED FOR OUR APPLICATION OF THE GT METHOD

RP method	References
Round-the-group prioritization	[10]
Ping Pong Balls	[39]
\$100 allocation (cumulative voting)	[30]
Multi-voting system	[41]
MoSCoW	[19]
Pair-wise analysis	[22] [27]
Weighted criteria analysis	[22]
Analytic Hierarchy Process (AHP).	[38]
Dot voting	[22]
Binary Search Tree (BST)	[2]
Ranking based on product definition	[18]
Planning Game	[9] [27] [25]
Quality functional deployment (QFD)	[16][22]
Wieger's matrix approach	[43]
Mathematical programming techniques	[31]
Technique of bucketing requirements	[34]
Kano Model	[12] [15]
Eclipse Process Framework	[5]
Relative weighting	[15]
Larman	[29]
Theme Screening/Scoring	[15]
Planning Poker	[11]

B. The Conceptual Models

The multiple iterations of coding, constant comparing of information from literature, and conceptual modelling in our GT process delivered two models, Model A and Model B, which are presented in Fig 1 and Fig 2, respectively. Model A describes the artefacts of the agile RP process, while Model B elaborates on the conceptual categories related to making the RP decisions. We make the notes: (1) that Model B (on Fig 2) is not meant as a refinement of Model A (on Fig 1). Instead, the purpose of Model B is to explicate and bring insights into the decision-making step, which is the core of the RP process; and (2) that both models take the perspective of the client, unlike RP authors [7,8,23,43] who adopt the perspective of the developers.

To create these models, we used the initial and the focused coding practices described in [13]. This meant first to name the segments of data, and then to use the most frequent initial codes to "sort, synthesize, integrate and organize large amounts of data". We complemented our coding with diagramming, which enabled us to visualize the connections among the conceptual categories and to

see more clearly the relative strength or weakness of the relationships between the concepts. Our intensive diagramming activity was motivated by Clarke [14] who contends that conceptual mapping preserves empirical realities and complexities. Due to space limitation, we do not provide a mapping between the literature sources, that we used as inputs to build the model, and the parts of the model derived from each source. We, however, plan to publish this in a separate paper in near future. Below, we describe the two models in more detail. We claim that both models aggregate the consensus among authors of published papers about client-centric RP.

1) Model A

This model presents a generic agile prioritization process in terms of its artifacts, as viewed from the client's standpoint.

Figure 1. Model A: the prioritization artifacts from clients' perspective. Boxes represent the state in which a requirement can be. Arrows represent possible state transitions.

We deliberately used concepts that make clear how the status of requirements changes – from ‘Initial’ to ‘Prioritized’, to ‘Sprint’, to ‘Implemented’. The input is the *Initial Project Backlog*, that is the total set of requirements upon the start of the project. Before the very first agile iteration, the client runs a RP technique, which produces a *Prioritized Project Backlog*. This is an ordered list of the requirements (originally written in the *Initial Project Backlog*) according to their priorities. In agile settings, only a small portion of the upper part of this ordered list goes for implementation in the first iteration. This small portion of prioritized requirements forms the so-called *Sprint Backlog*. Once the iteration is completed, the status of those requirements which are already implemented in the software product, changes to *Implemented requirements*. Those requirements which could not be implemented by the developers are fed back into the prioritized project backlog and are subjected to reprioritization before the new iteration starts. At that inter-iteration time, the client might also decide to request a change to the requirements and this would lead to a new reprioritization as well (this is the arrow from *Sprint backlog* to *Prioritized project backlog*). However, during an iteration, this iteration's ‘Sprint backlog’ is frozen and must not be changed. At inter-iteration time, the client reprioritizes the project backlog, so that she/he knows the next portion of requirements that will go to the next *Sprint Backlog*. The relationship between the concepts *Prioritized Project Backlog* and *Sprint Backlog* - from the

view point of the clients in agile projects, is elucidated in Model B (see Fig 2).

2) Model B

This model is to help clients ‘zoom-in’ and see those concepts important to consider in RP at inter-iteration time, including context. Model B describes what happens in all those RP processes about which we read in the sampled literature. As in Model A, in Model B we take a generic perspective of RP, that is, it abstracts from the use of a specific agile RP technique. Model B suggests that there is a consensus in the reviewed literature that there are five aspects that the client considers when making his/her decision on requirements priorities: *Business Value*, *Risk*, *Effort Estimation/ Size Measurement*, *Learning Experience*, and *External Change*. Iteration planning additionally considers *Project Constraints*. We observed these aspects reappear in the descriptions of applications of all RP techniques. What varies is their importance to the RP outcomes. For example, a RP process may or may not choose both business value and risk as prioritization criteria in all iterations. So, in some iterations, the clients may make decisions solely based on the business value of the features in the backlog, and in others – based on risk.

We observed that the agile RP literature sources converge on that the *Business Value* is the dominating RP criterion, whereby *Business Value* is estimated by the customer alone. We also observed that some RP methods used the notion of ‘importance’, or ‘relative importance’ (as they call it), or ‘contribution to user task or goal’ [34] of a feature, compared to other requirements, instead of ‘value’. And Wiegers [43] quantifies “relative penalty the customer or business would suffer if the feature is not included”. Still, when reading about the application of the RP method, it is our understanding that estimation of ‘value’ is the prerequisite for these prioritization methods. In XP’s Planning Game [9], the client quantifies value by sorting the story cards on one of three piles: (1) those without which the system will not function (“must have”), (2) those that are less essential but provide significant business value (“should have”), and (3) those that would be nice to have. In addition to Business Value, the client in some methods considers *Size/ Effort/ Cost*, and *Risk* caused by a requirement’s implementation. We also found that requirement dependencies, e.g. precedence constraints, seem to influence both value and cost involved in the implementation of one requirement [31] but are usually not taken into account explicitly by agile prioritization techniques. We make the note that size, effort, cost and risk are estimated by the developers and provided to clients for their decision-making. Accommodating highly-volatile requirements, which in turn, means accommodating uncertainties in the development process is an inherent aspect of the agile software process which serves risk management. In fact, the strong focus on business value, on early delivery and on continuous RP is the key to successfully coping with uncertainty and with volatile requirements.

Figure 2. Model B: a generic model of data dependencies: activities (rectangles with rounded corners), artefacts (rectangles) that are input and output of these activities and arrows (data or information flow)

In the Planning Game [9,25] the developer sorts the story cards according to (estimation) *risk* into three piles: (1) those that they can estimate precisely, (2) those that they can estimate reasonably well, and (3) those that they cannot estimate at all. We note that here we talk of the risk, i.e. uncertainty, associated with each specific requirement, not related to the project as a whole. The latter are the same as for waterfall projects, although countered in the agile way [21].

In addition, the client considers *Estimated Size* based on functional size when making decisions on priorities for the next iteration. 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 [15]. In larger projects, relative size - or relative effort, is estimated top-down on several levels of granularity: on the highest level, size or effort are estimated with respect to each (logical) subsystem, then with respect to features (or business stories) and finally with respect to the user stories [11][32]. The relative units used for quantifying size on each of these three levels are called 'syep' (which stands for 'system effort points'), 'feep' (meaning 'feature effort points') and 'step' (that are the so-called 'story effort points'), respectively [11]. For expressing that one could use any arbitrary unit, sometimes they are called "gummy bears" [32]. Size/effort estimation can be done by applying "estimation by analogy method" - this means, by comparing one user story to others.[15]. Size is estimated by the developers, so from the client's perspective, size is a given – but potentially uncertain – input.

Another aspect which can be a consideration during the decision-making are *Project Constraints* like *release date*, *iteration date*, *budget* and *velocity* [6,15,25]. As Li et al. [31] put it: "requirements are selected so that the total revenue is maximized against the available resources". The iteration date describes the end of a period of some weeks in which the requirements on the

sprint backlog are implemented, while at release date, software is delivered to the customer. Resources which they consider are: schedule (i.e. 'upper bound of project span'), number of developers/ working hours, and other resources. Project velocity in XP and in Scrum means the number of stories (i.e., requirements) or story points which can be implemented during a single iteration. Velocity is thus work divided by time. Project velocity is known to vary from iteration to iteration. The velocity or load factor [25,32] measures the elapsed time for implementation. Usually, it is a number greater than one, what means it takes more than one day to implement a requirement which causes one day of implementation effort (one IED = "ideal engineering day" [25]).

Velocity has value five, if it takes five days to implement a requirement which is only one day implementation effort. Project velocity is expected to depend on the number of developers and their number of working hours, developer experience and the technology used, but also on estimation bias and on external factors like vacations, problems or additional tasks. In Scrum, the burn-down rate describes which percentage of the requirements (effort) is implemented per day.

Learning is an in-built principle in agile development. M. Cohn [15] 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". As Larman [29] emphasizes, "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 [23] present how they used strategic learning as the vehicle to minimize costs and maximize benefits through and provide guidelines on

how to optimize business value. They advocate the necessity of adopting a dynamic approach to agile RP, in order to leverage the important aspect of learning in an agile project. Their focus is particularly on incorporating learning and cost of change in the decision-making process. Crow [16] recommends to “identify warranty, service, reliability, and customer complaint problems to identify areas of improvement”, i.e. of requirements. Harris and Cohn [23] recommend two guidelines be followed in order to profit from learning effects: (1) requirements which are likely to change should be deferred until more and better knowledge about how (or even whether) to develop them is gained, and (2) bring forward features that generate useful knowledge about desirability of a feature. All agile development techniques include one or several opportunities for learning in their process. These are – on different time scales – daily meetings like XP’s daily Stand Up Meeting or the Scrum Meeting/ Daily Scrum, daily build and regression testing, iteration planning - especially the reprioritization of requirements –, sprint review meeting – including measurement of project velocity - and project retrospectives at a project’s end. Highsmith [24] emphasizes the importance of ‘learning loops’, as he calls them: “We have to test our knowledge constantly—using practices like project retrospectives and customer focus groups. Furthermore, these review practices should be done after each iterative cycle rather than waiting until the end of the project. The quality of learning derived from practices like project retrospectives provides a key indicator about the true commitment to learning in an organization, and, therefore, a key to its adaptability”

For illustration purposes, we provide an example of the role of team’s learning in improving judgment-based estimation of size and effort within an iteration. Agile authors indicate [32], that in each iteration, one typically notes down real implementation effort for the requirements. While during the first iterations in a project, one often exclusively estimates size, later-on – because of the continuous learning by the team - it becomes possible to translate the size estimates into effort estimates. For example, the authors in [32] report on that one gathers experience data about how many ‘gummy bears’ can be implemented within one iteration.

External change, that is, a change in the project’s or company’s context, can influence requirements’ value and priorities during the project. In the reviewed literature on agile RE, we found only one explicit indication: competitive opportunities (i.e.: how do competitors perform in respect to a particular requirement) [16].

Last, the *Project Backlog* [15] means the same as in Model A and *Prioritized Project Backlog* is the ordered list of requirements to be implemented in the next iteration. ‘*Prioritized*’ means to attribute requirements a priority, which during *iteration planning* translates into

an order of implementation: Starting with the requirements with the highest priority, so many requirements are chosen for the sprint backlog as can be implemented within the iteration. The iteration is treated as a timebox, i.e. the available time span is assumed to be fixed. Requirements are sometimes ordered according to cost-benefit-ratio, or one looks for ‘quick wins’ (high benefit) and ‘low hanging fruits’ (low effort) [11]. In the Planning Game [9], the high value requirements are implemented first as well as the high risk requirements.

3) Discussion on Model B

Model B is compatible with any of the techniques listed in Table 1. This means that a client could use Model B to reason about his/her RP process when using any of these techniques. Clearly, not all of the elements in Model B are necessarily present in each RP activity – i.e. some of them depend on the project context. From the literature research, it did not become clear whether and how each method uses all of these concepts. We believe that agile development generates occasions for using all these concepts intuitively, but the informal character of agile development precludes explicit prescription.

Model B provides a generic *framework for describing* the client’s decision-making situation while prioritizing the requirements. As per Alenljung and Person [3], a decision-making situation is “a contextual whole of related aspects that concerns a decision-maker”, that is – in our case, the client in an agile project. For example, one can use the concepts of Model B to depict a concrete client’s RP situation in a concrete agile project, in a specific organization and, thus, take into account the topics important to be considered by the client when prioritizing requirements at inter-iteration time. However, due to the mixed quality of the literature which is the source of this model (which we discuss in Section VI), the model’s completeness should be validated empirically, e.g. by a case study. We should check whether the model’s concepts really cover everything that is important for designing a prioritization process.

Furthermore, Model B shows the complexity of the decision-making from client’s perspective in agile RP. We observe that the client, when prioritizing the *Project backlog*, explicitly or implicitly relies on tacit knowledge to estimate the *Business value* of each item (in the *Project backlog*). The estimation is qualitative (as it was already found in a previously published study on business value in agile [37]). Yet, the agile clients make a conscious effort to connect business value to “something that delivers profit to the organization paying for the software in the form of an increase in revenue, an avoidance of costs, or an improvement in service” [33].

The client assesses the *Business Value* of the requirements in the project backlog based on his/her current knowledge and *Learning Experiences* within the

agile project as well as any changes (see the box *External changes* in Fig 2) occurring in the business environment of the organization. An example for an external change can be a merger between the client's organization and another organization. Both the client's continuous learning throughout the project and the dynamic environment in which the client organization operates can make - from one iteration to another, some requirements more valuable than others. (It is possible that *External changes* can even render some requirements irrelevant.)

We make the note that here we propose conceptual models **only** that, as suggested by GT methodologists [13], are not supposed to be validated against the data that has been used for the development of the models. Our models come out of a literature study and are descriptive and not prescriptive, which means that we do not make causal claims about phenomena and variables in the domain we study, namely agile prioritization process. Instead, the purpose of the models is to capture the reality of the prioritization process as reflected in the literature sources which were used as data sources for this study. We can *only* evaluate the resulting models against the evaluation criteria of the GT methodologists, which we do in sections V and VI. The validation of the models (i.e. how well they describe the phenomenon in question) will be a subject of a follow-up empirical investigation, e.g. by the means of future case studies.

V. REFLECTION ON OUR USE OF GT

Research methodologists [20, 40] emphasize that when a researcher builds up a conceptual model by using a qualitative approach as GT, the researcher should evaluate its emerging model in terms of three key criteria: (i) adequacy, (ii) fitness (or relevance) and (iii) modifiability [20].

Adequacy of the result of the GT process is to be assured by applying the set of techniques and analytical procedures in the GT. Examples are to adhere as closely as possible to the GT principles and processes, to code the data independently by each researcher before re-coding them in joint work discussions (in order to ensure the highest possible degree of inter-coder reliability), to consult literature to evaluate similarities and dissimilarities of the resulting theory to extend literature, and to check for any category, property or property value that might have been overlooked. We made conscious effort to keep these GT principles, however, we must be clear on a validity concern arising from the fact that most of the time the first three authors, that applied the GT, worked away from each other at three different locations and could not do much joint coding. However, our individual coding processes lead to the same results.

The relevance of the conceptual model to researchers is to be judged regarding how it fits the situation, that is, whether it helps individuals familiar with the

phenomenon (in this study, agile RP) to make sense of their experience and to manage the situation better. We took two steps to make sure we preserve the meanings of the clients in agile projects: (1) we consistently engaged ourselves in diagramming activity, and (ii) we searched and included the so-called 'in-vivo' codes, as recommended in [13]. These are special terms from the world of the practitioners in the studied context, which are assumed that everyone "knows and shares" them, which flag condensed but essential meaning, and which reflect assumptions that frame some actions. In our case, examples of in-vivo codes, associated to clients in agile RE, were "backlog" (meaning those requirements in an agile project, which are subjected to the implementation – for the whole project, as well as for immediate future iteration, 'project backlog' and 'sprint backlog' respectively) and "sprint" (meaning an individual agile iteration in a project). We looked into the implicit meanings behind these terms, which brought us to Model A on Fig 1. Beyond these two steps, in our immediate future research, we plan to demonstrate the fit of the framework by using it in case studies.

Furthermore, modifiability of an emerging theory is concerned with the possibility to update it and extend it in the future. We made a conscious effort to maintain a balance between keeping the concepts abstract enough - so that the theory can serve as a general explanation, and making sure the concepts do not get too abstract as to lose their sensitizing characteristics. In our view, we should keep our framework open as it makes more sense to invite other researchers to use it and test it, only after this, to strive for all-inclusive and general results. We think that if industrial uptake of agile software development practices increases and more knowledge on the client's role and the client-developer interaction modes becomes available, our framework will need some refinement and extension so that it's kept useful.

Last, we point out like other qualitative research approaches, the GT approach implies the risk that the researchers assume that the conceptual categories are saturated, when they might not be. Following Charmaz [13], we remained open at all times to any new literature source and whenever we felt we were getting stuck, we stepped back and re-coded the earlier collected information and looked for new leads. We also looked at many cases of agile RP, while carrying out the theoretical sampling and this increased our understanding of the empirical world and helped us discern variations in the conceptual categories we use to describe the agile RP from client's perspective. For example, we consulted two practitioners (one from large IT-solution providing company, and one from a mid-sized agile software development company), when we were trying to figure out (i) how clients define, estimate and use size in agile RP context, and (ii) how clients define business value. These practitioners brought insights into the variation in the meanings of our conceptual categories *Size Measurement*, *Business Value*,

and *Risk*. Checking our concepts against the empirical realities of these practitioners was instrumental to understand how, when, and why the meanings of our categories vary. We stopped our theoretical sampling process when we noticed that further acquisition of data from real project experiences did not bring new ideas nor opened up new ways to think of the properties of our conceptual categories. In GT, this state is called ‘saturation’ of the resulting conceptual model [13]. As we acknowledge that the judgment about the point at which we stop the theoretical sampling might be subjective, in the immediate future, we are planning case studies on real projects with companies in which we will use Model B as our framework to describe the client’s context of RP at inter-iteration time.

VI. REFLECTION ON THE QUALITY OF LITERATURE SOURCES

We noticed that most of the sources, which we retrieved through our search strategy, are authored by practitioners. We admit that these papers may vary in terms of quality of evidence they offer to researchers. Clearly, experience reports by practitioners might lack scientific rigor and this matter can represent a treat to the validity of the results in our study. However, we have to acknowledge that there is not much theory in the field of our research yet, and that the researchers have to use, as first step for their theory-building process, such sources that might be, strictly speaking, suboptimal choices from research perspective. What we think is important in this case, is to consider the information in these sources as indicative with respect to the studies phenomena (namely, agile RP). And this is in line with the understanding in GT that heterogeneous material can be analyzed and that the resulting conceptual model should be modifiable and evolvable [20]. Moreover, one recurrent observation in our study is that much tacit knowledge is involved in the agile RP process, and prioritization is often done based on experience or gut feeling of the experts, and is often performed as an implicit part of a more complex decision-making step, and not necessarily as a conscious separate step. This partially explains the scarce amount of explicit sources being available and also calls for a different research approach to be used to uncover tacit and implicit knowledge, e.g. case studies or action research.

The above-mentioned observations converge with the explicit statements by the proponents of the agile paradigm on how the agile community accumulates a body of evidence and knowledge. For example, Scott Ambler, a prominent representative of the practitioners’ community, points out in [4], that there are two ways to show the merits of agile development – one is by collecting evidences from the practice, and the other is “to wait until there is a scientific ‘proof’”. We do observe that the existing body of knowledge in this domain consists mainly of anecdotic evidence. We also observe that the anecdotic body of knowledge is rapidly

growing. The pioneers in agile development have written the first books and papers which have served as sources to spread knowledge and experience to other practitioners. Some of their successors, in their turn, have published their own experiences in books or in articles in online journals or in the agile conferences. The publication output produced by the agile community represents a challenge for the research community when it comes to applying scientific research methods. To confront this challenge, we - the researchers, have to be creative and apply combinations of qualitative and quantitative research methods, so that we can ensure our research processes are of good quality.

Nevertheless, as was mentioned in Section IV A, we additionally analyzed books by practitioners because we expected them to be more complete and more detailed than the other, short publications. These books were chosen because they are authored by prominent practitioners in the field and discuss practical applications of the agile methodology.

In the light of the discussion above, we think that there is relatively limited threat to validity of our conceptual models as: (i) even if the sources are mainly of anecdotic character, they are primary sources that truly reflect the author’s experience and opinion about the prioritization process in his/her concrete project settings; moreover, they reflect real-life instances of the investigate phenomenon; (ii) we consistently observe that the data available in multiple primary sources report similar experiences and converge in their opinions.

VII. SUMMARY AND OUTLOOK

This paper investigated the question of what concepts are important to consider when reprioritizing agile requirements from clients’ perspective at inter-iteration time. We proposed two conceptual models that were derived by GT explicating the RP in agile projects. These conceptual models fill a gap in the current agile RE literature, which lacks comprehensive studies on agile RP. These models present the state of the practice described by concepts that we discerned from reading relevant publications. Our conceptual model is a first proposal only. Clearly, we can and should not expect that it can explain how the RP decision-making takes place in the different contexts in which agile approaches are practiced. In order to gain a deeper insight in this process, we intend to use the two conceptual models as a structure for further empirical investigations about agile software projects. Specifically, we are right now carrying out two follow-up studies: First, we used Model B to produce a case study protocol for research with practitioners on the topic of prioritizing requirements in agile context. In this case study we include in-depth interviews with 11 practitioners from 8 development and consulting companies to answer the question of how exactly agile RP decision-making happens in practice. Currently, we are processing the data of the interviews.

Second we started analyzing books that are dedicated to different aspects of the agile software development, where the prioritization is covered in some extent. Authors of the books are practitioners with long experience in agile development projects. The new findings from these sources will be used to verify and, if needed, to modify, or refine the models presented here. We also invite agile RE researchers to explore the area and to accumulate support for – or a challenge to – the proposed theory.

While discussing the models, some open questions arose which could not be answered by re-analyzing the published data and which might be investigated in future research by using empirical methods. Such questions are: (i) When a requirement which was in the sprint backlog finally is not implemented and is fed back into the prioritized requirements backlog – how is its priority determined? How is the learning about the requirements taken into account in this transition? (ii) Although agile techniques generate many occasions for learning and for adapting to external changes, our literature research identified no concepts for describing what is learned from project experience or about external changes and how this updated information is used during the RP process. It would be interesting to observe such learning processes in case study research or to propose concepts from knowledge management research. (iii) Could agile RP techniques be more usefully applied by combining them explicitly with decision-making concepts? Or does this rather complicate the RP and destroy its intuitiveness which is the explicit strength of agile techniques? (iv) How does one treat the situation when there are “multiple client voices”, i.e. several clients have different priorities? (v) What is the linkage between the concrete project settings and the respective RP process that holds for this project? This knowledge can be used twofold: (1) to help researchers and practitioners to identify best practices and provide guidelines for practitioners, and (2) to serve as a roadmap for further empirical research to investigate the fitness of different RP approaches to different contexts.

ACKNOWLEDGEMENT

This research has been funded by the Netherlands Research Foundation (NWO) under the QUADREAD project and under the CARES project. The authors would like to thank the practitioners Luigi Buglione, Thijs Munsterman, Erlend Engum, the colleagues Klaas Sikkels, Klaas van den Berg, Siv Hilde Houmb, and the participants of the APSO workshop at ICSE 2008 for sharing ideas on the topic of agile requirements prioritization.

REFERENCES

[1] Agile Journal <http://www.agilejournal.com/>

- [2] Ahl, V. An Experimental Comparison of Five Prioritization Methods, Master's Thesis, School of Engineering, Blekinge Institute of Technology, Ronneby, Sweden, 2005.
- [3] Alenljung, B., A. Person, Portraying the Practice of Decision-making in Requirements Engineering: a Case Study of large Scale Bespoke Development, *Requirements Engineering Journal*, 2008, 13, pp. 257-279.
- [4] Ambler, S., Answering the "Where is the Proof That Agile Methods Work" Question, <http://www.agilemodeling.com/essays/proof.htm>
- [5] Ambler, S., Complex requirements on an Agile Project, Dr Dobbs, Oct 31, 2008, <http://www.ddj.com/architect/211800534?cid=Ambysoft>
- [6] Ambler, S.W., *Agile Modeling - Effective Practices for eXtreme Programming and the Unified Process*, Wiley, New York, 2002
- [7] Augustine, S., *Managing Agile Projects*, Prentice-Hall, 2005.
- [8] Barney S., Aurum A., C. Wohlin, A Product Management Challenge: Creating Software Product Value through Requirements Selection, *Journal of Software Architecture*, 54 (2008), pp. 576-593.
- [9] Beck, K. *eXtreme Programming Explained: Embrace Change*, Addison Wesley, 2000.
- [10] Berteig, M., Methods of Prioritization, March 20, 2006 in Agile Advice Online Practitioners Forum, http://www.agileadvice.com/archives/2006/03/methods_of_prio.html
- [11] Bleek, W-G, H. Wolf, *Agile Softwareentwicklung: Werte, Konzepte und Methoden*. dpunkt, Heidelberg, 2008.
- [12] Center of Quality Management Journal (1993) Special issue on Kano's Methods for Understanding Customer-defined Quality , Vol 2, No 4, Fall 1993
- [13] Charmaz, K. *Constructing Grounded Theory: a Practical Guide through Qualitative Research*, Thousand Oaks CA, Sage, 2007.
- [14] Clarke, A. *Situational Analysis: Grounded Theory after the Postmodern Turn*, Thousand Oaks, CA, Sage, 2005.
- [15] Cohn, M. *Agile Estimating and Planning*, Prentice Hall, 2005.
- [16] Crow, K. Customer-focused Development with QFD, URL: <http://www.npd-solutions.com/qfd.html>, Accessed on August 26, 2009.
- [17] Dr Dobb's portal <http://www.ddj.com/>
- [18] Fraser, J., Setting Priorities, April 23, 2002, URL: <http://www.adaptivepath.com/ideas/essays/archives/000018.php>
- [19] Getting Started With Use Case Modeling, An Oracle White Paper, May 2007 <http://www.oracle.com/technology/products/jdev/collateral/papers/10g/gswUseCaseModeling.pdf>
- [20] Glaser, B. G., *Basics of Grounded Theory Analysis: Emergence vs Forcing*, Sociology Press Mill Valley, 1992.
- [21] Gottesdiener, E. How Agile Practices Reduce Requirements Risk, Better Software, July/Aug, 2009.
- [22] Gottesdiener, E., At a Glance: Other Prioritization Methods, EBG Consulting, Inc. www.ebgconsulting.com
- [23] Harris, R. S., M. Cohn, Incorporating Learning and Expected Cost of Change in Prioritizing Features on Agile Projects. *XP 2006*: pp. 175-180
- [24] Highsmith, J. *Agile Project Management: Creating Innovative Products*
- [25] Hunt, J., *Agile software construction*. Springer, London, 2006.
- [26] InfoQ software development community <http://www.infoq.com/agile>
- [27] Karlsson, L., Thelin, T., Regnell, B., Berander, P., Wohlin, C., Pair-wise Comparisons versus Planning Game Partitioning-experiments on Requirements Prioritisation Techniques, *Empirical Software Engineering*, 12(11), 2007

- [28] Kitchenham, B., Procedures for Undertaking Systematic Reviews, Joint Technical Report, Computer Science Department, Keele University (TR/SE-0401) and National ICT Australia Ltd. (0400011T.1), 2004.
- [29] Larman, Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-Scale Scrum, Addison-Wesley, 2008.
- [30] Leffingwell, D., Widrig, D., Managing Software Requirements, 2nd ed. Boston, Addison-Wesley, 2003
- [31] Li, C., Akker, J.M. van den, Brinkkemper, S., Diepen, G. Integrated Requirement Selection and Scheduling for the Release Planning of a Software Product, In Proc. of REFSQ '07, Spinger LNCS, pp. 93-108.
- [32] Lippert, M., S. Roock, H. Wolf, , eXtreme Programming in Action: Practical Experiences from Real World Projects, John Wiley & Sons , 2002
- [33] Patton, J. Ambiguous Business Value Harms Software Products, IEEE Software, 25(1), Jan/Feb 2008.
- [34] Patton, J., Finding the Forest in the Trees, In Proc of the IEEE Conf. on OOPSLA, 2005, pp: 266 – 274.
- [35] Petersen K., C. Wohlin, A Comparison of Issues and Advantages in Agile and Incremental development between State of the Art and an Industrial Case, Journal of Systems and Software, 82 (2009), pp. 1479-1490.
- [36] Principles behind the Agile Manifesto, 2001, URL: <http://agilemanifesto.org/principles.html>
- [37] Racheva, Z., M. Daneva, K. Sikkil, Value Creation by Agile Projects: Methodology or Mystery? In: Proc. of the Conf. on PROFES, Springer, LNCS, pp.141-155
- [38] Saaty, T.L., The Analytic Hierarchy Process, McGraw-Hill, New York, 1980.
- [39] Schwaber, K., Agile Project Management with SCRUM, Microsoft Press, 2004.
- [40] Strauss, A.L., J.M. Corbin, Basics of Qualitative Research - Grounded Theory Procedures and Techniques, Sage, Newbury Park, 1991.
- [41] Tabaka, J., Collaboration Explained: Facilitation Skills for Software Project Leaders. Addison Wesley 2006.
- [42] Webster J., R.T. Watson: Analyzing the Past to Prepare for the Future: Writing a Literature Review. MIS Quarterly 26(2): (2002)
- [43] Wiegers, K., First Things First: Prioritizing Requirements, Software Development, 7(9) 1999.