1 Introduction

Agile software development consists of a multitude of iterations in between which the decision what requirements to implement is re-evaluated by a process of requirements re-prioritization. In 2010, Racheva, Daneva, Herrmann, and Wieringa (2010b) examined 22 different Requirements Prioritization (RP) methods and extracted an abstracted, conceptual model of the RP process from a generic perspective. Four months later, Racheva, Daneva, and Herrmann (2010a) present an improved version of this model based on a case study among 11 practitioners of companies that follow agile methodologies. The paper discussed in this method description (Bakalova, Daneva, Herrmann, & Wieringa, 2011) presents the same improved version of this model. In fact, a majority of this paper is identical to Racheva et al. (2010a), with the addition of mapping the concepts of the aforementioned 22 RP methods to the aspects of the improved model.

The end-goal of the method is to inform decision-makers (clients) of the important concepts that are considered during a RP iteration by practitioners. In the end, this information helps the client in three ways: "(i) to navigate through the agile process of delivering business value; (ii) to make explicit the tacit assumptions in different RP methods; (iii) to identify those possible pieces/sources of information important to the outcome of the prioritization and, consequently, to the project." (Bakalova et al., 2011, page 183).

Agile Requirements Prioritization consists of three subsequent activities: the team starts with Value Estimation of all requirements on the project backlog. Next, the team takes the value of each requirement into account when creating the prioritized project backlog during Requirements Prioritization. Finally, the project manager starts Iteration Planning, he selects a number of requirements to be completed in the next iteration and puts them on the final deliverable: the iteration backlog. During each of these activities, up to eight aspects are taken into account when making decision on requirements prioritization: (1) Project Context consisting of project settings such as ‘project size’ or ‘size of the client’s organization’, which also has a significant impact on (2) Prioritization Criteria together with estimated Business Value, (3) Estimated Size - broken down into functional size of a requirement and a size/effort estimation, (4) Input from the Developer providing advice and alternative solutions, (5) External Change represents events during the project that impact the company, business environment or product, (6) Learning Experiences comprises of new insights acquired by the client and/or the developers, (7) Project Constraints such as release date or budget and (8) Dependencies between requirements both architectural or chronological.

Figure 1 connects all aspects to the relevant activities in a RP process.

The authors of this paper are Zornitza Bakalova, Maya Daneva and Roel Wieringa of the Computer Science Department of the University of Twente together with Andrea Herrmann of Axivion GmbH. Bakalova is a PhD researching Agile Software Development with a focus on Requirements Engineering. To complicate things, she was formerly known as Racheva. As a result the main author of the two publications cited first in this article is identical to Bakalova. Daneva is an Assistant Professor mainly doing research into Requirements Engineering. Roel Wieringa is a Full Professor and head of the CS Department doing research on a diverse range of topics related to IS. Axivion GmbH, the company Andrea Herrman works for, is a German spin-off company of the University of Stuttgart that sells the commercial version of a software product to stop software erosion, developed in collaboration with the University of Bremen. However, according to her LinkedIn profile, Andrea Herrmann has been a private lecturer at the University of Heidelberg since 2004. As a consequence. She has accumulated 37 publications to date and actively contributes to the Requirements Engineering field.
2 Example

Due to the goal of the authors, the method discussed in this report is a meta-model of RP in general. Although applying the method to a real-world agile environment is possible, the authors refrain from discussing this specific use case. Instead, they map the concepts of their model to the 22 prioritization methods identified in Racheva et al. (2010b). The first and second paragraphs of this section duplicates a fraction of this work by discussing the process of one RP method and identifying the steps that are comparable to concepts of the discussed model. The third paragraph is an example of how a team prioritizes five requirements on a project backlog.

Wiegers Matrix Approach (Wiegers, 1999) calculates relative priority scores for a list of requirements by estimating four Prioritization Criteria. This RP method was chosen because it contains the largest absolute number of comparable concepts and the initial paper is available to anyone. Step 1 of Wiegers Matrix approach is the creation of a Project Backlog which lists all requirements, features or use cases that require prioritization. Second, the relative benefit to the customer is estimated on a 1 to 9 scale. Inversely, step 3 estimates the penalty if the feature is not included in the end product. Step 4 calculates the sum of step 2 and 3 and subsequently divides this figure by the total value of all requirements. The result is a representation of the Value of Requirement in percentage. In step 5 and 6, Input From Developers is leveraged to estimate the relative cost of implementing each feature and the relative degree of technical or other risk associated with each feature. Both are normalized in percentage of the total value, resulting in a relative Size/Effort Estimation. In step 7, depending on Project Context, the participants of the prioritization process assign weights to each of the former steps. Subsequently, the priority number is calculated as value %/(cost % * cost weight + risk % * risk weight) so the participants can list the features in descending order by priority in step 8. Depending on Project Constraints the customer and developer representatives review and adjust the final list.

This assessment facilitates the creation of an example deliverable. Wiegers Matrix Approach contains 7 out of 10 concepts of the discussed method, each of which has received a grey background in Table 1. Project Backlog and Value of Requirement are not part of the eight aspects that influence the RP process. However, the creators include them because they are deliverables of a multitude of RP methods. Furthermore, the authors consider two project backlog variations to be out of scope of their paper but include them in the comparison. Because Wiegers Matrix Approach does not include these variations, they have been omitted from Table 1.

<table>
<thead>
<tr>
<th>Project Context</th>
<th>Prioritization Criteria</th>
<th>Size/Effort Estimation</th>
<th>Input From Developers</th>
<th>External Change</th>
</tr>
</thead>
<tbody>
<tr>
<td>Learning Experiences</td>
<td>Project Constraints</td>
<td>Dependencies</td>
<td>Project Backlog</td>
<td>Value of Requirement</td>
</tr>
</tbody>
</table>

Table 1: Model Concepts Available in Wieger’s Matrix Approach
A development team working on a software product is ready to start the new iteration cycle. To begin, the Project Manager lists the five requirements on the project backlog:

1. Add trivia section to user profile
2. Automatically logout on all devices
3. Join trial and free users into regular users
4. Increase profile picture size
5. Change file upload with paperclip gem to carrierwave

Next, the team starts the value estimation process. Each requirement is assigned a numerical value based on the team’s judgement. This judgement takes into account the project’s context, prioritization criteria, learned experiences, external changes, a size/effort estimation and input from developers. In our example, each requirement will be assigned a numerical value in the Fibonacci sequence; a best practice in Agile requirements estimation. Because the underlying activities for arriving at these aspects is different for each development team in practice, the paper does not explicate what the activities to arrive at these aspects entails. When each requirement is assigned a value, the team assigns a priority to each requirement. During this process, the team first sorts the requirements by their value and then again takes into account prioritization criteria, learned experiences, external changes, a size/effort estimation and input from developers together with the project’s constraints and dependencies. For this example, we will assume that these considerations do not change the priority based on sorted value. Table 2 presents the prioritized backlog.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Requirement ID</th>
<th>Requirement Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>unify_users222</td>
<td>Join trial and free users into regular users</td>
<td>13</td>
</tr>
<tr>
<td>2</td>
<td>upload-36</td>
<td>Change file upload with paperclip gem to carrierwave</td>
<td>8</td>
</tr>
<tr>
<td>3</td>
<td>logout-8</td>
<td>Automatically logout on all devices</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>user_model-23</td>
<td>Add trivia section to user profile</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>profile-113</td>
<td>Increase profile picture size</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 2: The prioritized backlog

In the last step, the Project Manager reviews the prioritized priority backlog, selects the requirements to be implemented in the next development iteration and adjusts the priorities according to his judgement if necessary. During this process the Project Manager takes into account his new learned experiences, external change and the project’s constraints. In this case, the PM is of the opinion that it is useful to modify the user model first before unifying the trial and free users. Furthermore, increasing the profile picture is no longer necessary because upper management changed its mind. The final deliverable of this process or iteration backlog is presented in Table 3.

<table>
<thead>
<tr>
<th>Priority</th>
<th>Requirement ID</th>
<th>Requirement Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>user_model-23</td>
<td>Add trivia section to user profile</td>
<td>3</td>
</tr>
<tr>
<td>2</td>
<td>unify_users222</td>
<td>Join trial and free users into regular users</td>
<td>13</td>
</tr>
<tr>
<td>3</td>
<td>upload-36</td>
<td>Change file upload with paperclip gem to carrierwave</td>
<td>8</td>
</tr>
<tr>
<td>4</td>
<td>logout-8</td>
<td>Automatically logout on all devices</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 3: The iteration backlog

3 Related Literature

Arguably, the earliest reference of Agile Requirements Prioritization in the form of incremental development and evolutionary advancement dates back to a report from Zurcher and Randell (1968), describing a design process consisting of a base-system which is sequentially expanded with an increasing amount of function and detail. Nevertheless, the first accepted formalization of prioritized development cycles by risk did not appear until Boehm (1986) introduced the "Spiral Model of Software Development and Enhancement" for risk-driven-iterations; a method to determine the priority of requirements in iterative and incremental development (IID). Since the Agile Manifesto was released (K. Beck et al., 2001), a variety of earlier development approaches are now referred to as agile methodologies (Larman, 2004). As a consequence, the new field of Requirements Prioritization encapsulates many accredited, recognized prioritization methods. For instance, Analytic Hierarchy Process (AHP) was developed by Saaty (1980) and the XP planning game
was first introduced in 1999 (K. Beck, 1999). In the years following the release of the Agile Manifesto, several authors conducted research into Agile Requirements Prioritization methods. For instance, Paetsch, Eberlein, and Maurer (2003) compare Agile Requirements Engineering to traditional RE and found that, in comparison, agile methods are “described vary[sic] vaguely and the actual implementation is left to the developers”. Berander and Andrews (2005, page 75-79) provide a general overview of RP techniques and compare five RP methods on scale, granularity and sophistication.¹ The authors proceed to recommend to use the simplest appropriate method in general and only use more sophisticated ones to resolve disagreements or supporting critical decisions. Cao and Ramesh (2008) finds that, in practice, organizations that practice Agile Development already apply multiple RP methods during a project. Nevertheless, Racheva et al. (2010b, page 287) noticed that “researchers in Agile RE case studies found that the creation of software product value through requirements prioritization decision making is only partly understood … a generic conceptual model describing requirements reprioritization in agile development is missing”, in order to proceed to fill that gap by proposing the first version of the method described in this paper.

None of the three publications that introduce different versions of the discussed method have accumulated citations that apply the method in a case study or another RP method. The latter is a consequence of the origin of the method, which is based on a review of all RP methods known in academic literature. Although practitioners might apply the method in their day to day operations, there is no way to confirm whether this happens due to the lack of a name for the method. Based on (Racheva et al., 2010a) identification of the importance and varying perception of business value in academia and business, Heidenberg, Weijola, Mikkonen, and Porres (2012) propose a method to model business value itself.

4 Process-deliverable diagram

The previous sections analyzed a requirements prioritization (RP) method to be used during inter-iteration (Bakalova et al., 2011). To augment the analysis, this section presents a meta-model of the RP method in the form of a process-deliverable diagram (PDD). The creation of a PDD is described by Weerd and Brinkkemper (2008) and is used for modeling activities and deliverables of processes. The method is based on the UML activity and class diagramming methods and its notation is therefore similar. Refer to Weerd and Brinkkemper (2008) to study the notational details. After creation of a PDD, it is accompanied by two tables that elaborate on each element in the diagram. With this elaborate analysis, requirements engineering researchers can conduct method comparisons or method adaptations.

To prevent clutter and unnecessary complexity, the PDD for this method is broken down into two layers of abstraction. The first layer models the main activities and deliverables of the method, shown in the grey blocks of Figure 1 of the Method Description. The second layers further details the main activities by explicating the sub-activities contained within each main activity.

The first layer detailed in Figure 2 begins with listing all requirements on the requirements backlog of an agile software project by the project manager. Next, the value of each requirement is estimated by the entire team in the second activity. The team subsequently assigns a priority to each requirement based on their value in order to create a prioritized project backlog. Finally, the project manager selects a subset of the requirements in the prioritized project backlog to create an iteration backlog. In practice, a typical business does not adhere to one strict Agile development method, instead a multitude of methods is combined into one unique method. Because of this, the authors are unable to specify the exact format of a requirement, requirements list and prioritized project backlog or define a concrete process of estimating value, assigning priority or choosing which requirements to select for the iteration layer. As a result, none of the concepts in this PDD have been assigned properties as it would be inappropriate to limit the method to a more narrow scope. The paper does specify what aspects are possible to take into account during the three main activities. Figures 3, 4 and 5 detail these aspects in the next layer by presenting the sub-activities and sub-concepts for each main activity.

The paper mentions that each sub-activity consists of multiple sub-sub-activities which result in sub-sub-concepts, but does not detail what these are or what methods by other authors should be used. Although a limited number of examples of what aspects to consider are given, they are always incomplete and non-mandatory. Therefore, all of the sub-activities and sub-concepts in Figures 3, 4 and 5 are closed because

¹There are many other publications that compare single RP methods to one another. Due to the numerical size of RP methods the discussed paper is based on, these have been omitted from this section.
their sub-activities are considered out scope. Secondly, the discussed method does not specify any order
nor concurrency of sub-activities. Instead, the authors state that practitioners are free to include any of the
sub-activities they wish in any order, but are not mandated to do so. The next two pages will introduce the
first two PDD layers, followed by the activity and concept tables on the next pages.

Figure 2: PDD - Layer 1
Value estimation based on the criteria

- Evaluate Project’s Context
- Evaluate Prioritization Criteria
- Estimate Size/Effort
- Evaluate External Change
- Evaluate New Learned Experiences
- Evaluate Input from Developers

Figure 3: PDD - Layer 2 - Value Estimation

Requirement prioritization

- Consider project constraints
- Evaluate Prioritization Criteria
- Estimate Size/Effort
- Evaluate External Change
- Evaluate New Learned Experiences
- Evaluate Input from Developers
- Consider dependencies

Figure 4: PDD - Layer 2 - Requirement Prioritization

Iteration planning

- Consider project constraints
- Evaluate External Change
- Evaluate New Learned Experiences

Figure 5: PDD - Layer 2 - Iteration Planning
<table>
<thead>
<tr>
<th>Activity</th>
<th>Sub-activity</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>List requirements</td>
<td></td>
<td>The Project Manager consults the project backlog to formulate a REQUIREMENTS LIST.</td>
</tr>
<tr>
<td>Value estimation based on the criteria</td>
<td>Evaluate project's context</td>
<td>The team considers the project size and size of the client organization during determination of the VALUE of a REQUIREMENT.</td>
</tr>
<tr>
<td></td>
<td>Evaluate prioritization criteria</td>
<td>The team estimates risk, business value and negative value of a requirement during determination of the VALUE of a REQUIREMENT.</td>
</tr>
<tr>
<td></td>
<td>Estimate size/effort</td>
<td>The team estimates size, effort, cost and risk associated with a REQUIREMENT to determine its VALUE.</td>
</tr>
<tr>
<td></td>
<td>Evaluate external change</td>
<td>The team evaluates the influence of events that happened during the project on the VALUE of a REQUIREMENT.</td>
</tr>
<tr>
<td></td>
<td>Consider learning experiences</td>
<td>The team takes into account the newly acquired technical knowledge and insights about desired software functions for the VALUE of a REQUIREMENT.</td>
</tr>
<tr>
<td></td>
<td>Evaluate input from developers</td>
<td>The team considers the alternative solutions and the organizational interests of the business brought up by the developers during determination of the VALUE of a REQUIREMENT.</td>
</tr>
<tr>
<td>Requirements prioritization</td>
<td>Evaluate prioritization criteria</td>
<td>The team estimates risk, business value and negative value of a requirement during creation of the PRIORITIZED PROJECT BACKLOG.</td>
</tr>
<tr>
<td></td>
<td>Estimate size/effort</td>
<td>The team estimates size, effort, cost and risk associated with a requirement during creation of the PRIORITIZED PROJECT BACKLOG.</td>
</tr>
<tr>
<td></td>
<td>Evaluate external change</td>
<td>The team evaluates the influence of events that happened during the project when creating the PRIORITIZED PROJECT BACKLOG.</td>
</tr>
<tr>
<td></td>
<td>Consider learning experiences</td>
<td>The team takes into account the newly acquired technical knowledge and insights about desired software functions during creation of the PRIORITIZED PROJECT BACKLOG.</td>
</tr>
<tr>
<td></td>
<td>Evaluate input from developers</td>
<td>The team considers the alternative solutions and the organizational interests of the business brought up by the developers during creation of the PRIORITIZED PROJECT BACKLOG.</td>
</tr>
<tr>
<td></td>
<td>Consider project constraints</td>
<td>The team takes into account the project duration, release date, budget, velocity and available resources during creation of the PRIORITIZED PROJECT BACKLOG.</td>
</tr>
<tr>
<td></td>
<td>Consider dependencies</td>
<td>The team takes into account chronological and architectural dependencies of the software project during creation of the PRIORITIZED PROJECT BACKLOG.</td>
</tr>
<tr>
<td>Iteration planning</td>
<td>Evaluate external change</td>
<td>The Project Manager evaluates the influence of events that happened during the project when selecting REQUIREMENTS for the ITERATION BACKLOG.</td>
</tr>
<tr>
<td></td>
<td>Consider learning experiences</td>
<td>The Project Manager takes into account the newly acquired technical knowledge and insights about desired software functions during selection of REQUIREMENTS for the ITERATION BACKLOG.</td>
</tr>
<tr>
<td></td>
<td>Consider project constraints</td>
<td>The Project Manager takes into account the project duration, release date, budget, velocity and available resources during selection of REQUIREMENTS for the ITERATION BACKLOG.</td>
</tr>
</tbody>
</table>

Table 4: Activity Table of All PDDs
### Concept List

**REQUIREMENTS LIST**
List with requirements for the projects (Bakalova, et al., 2011 page 187).

**REQUIREMENT**
A requirement that specifies or constrains the coding or construction of a system or system component (IEEE, 1990).

**VALUE**
The regard that something is held to deserve; the importance, worth, or usefulness of something. (Oxford Dictionary, 2013) In this context, value is a numerical representation of a REQUIREMENT's worth based on the results of the value estimation activity.

**PRIORITIZED BACKLOG**
Prioritized Project Backlog is the ordered list of requirements (Bakalova, et al., 2011 page 187). In practice it is the REQUIREMENTS LIST sorted by VALUE of each REQUIREMENT.

**PRIORITY**
The position in the order of implementation of requirements (Bakalova, et al., 2011 page 188). In this context it is the position on the PRIORITIZED BACKLOG based on the assigned VALUE.

**ITERATION BACKLOG**
A sub-set of the REQUIREMENTS on the PRIORITIZED BACKLOG to be implemented in the next development iteration of a software product (Bakalova, et al., 2011 page 187).

**PROJECT CONTEXT**
Project settings such as ‘size of the project’ or ‘size of the client’s organization’, and is used to explicate the impact of these settings on the prioritization process (Bakalova, et al., 2011 page 186). Deliverable of evaluate project context.

**PRIORITIZATION CRITERIA**
Secondary project settings prioritization is based on. Three main criteria are Business Value, Negative Value and Risk (Bakalova, et al., 2011 page 186). Deliverable of evaluate prioritization criteria. Practitioners take into account during value estimation and requirement prioritization.

**SIZE/EFFORT ESTIMATION**
Estimated size/effort based on functional size of a requirement (Bakalova, et al., 2011 page 186). Taken into account when making decisions on REQUIREMENT VALUE and PRIORITY.

**EXTERNAL CHANGE EVALUATION**
Events that happen during the project and impact the company, the business environment or the product under development (Bakalova, et al., 2011 page 186). Considered during all three main activities.

**LEARNED EXPERIENCES**
New insights acquired by both the clients and the developers during the project, such as new knowledge about technical solutions, or new insights about the desired functionality of the product under development (Bakalova, et al., 2011 page 187). Taken into account during all three main activities.

**DEVELOPER INPUT**
Advice and alternative solutions provided by the developers of the software project (Bakalova, et al., 2011 page 187). Deliverable of evaluate input of developers. Taken into account during value estimation and requirement prioritization.

**PROJECT CONSTRAINTS**
Constraints are anything that either restricts the actions of the project team or dictates the actions of the project team (Heldman, 2005 page 568). Deliverable of consider project constraints. Considered during requirement prioritization and iteration planning.

**DEPENDENCIES**
When two system elements have a relationship due to a connecting property a dependency is created (Beck and Diehl, 2011 page 355). Deliverable of consider dependencies. Taken into account during requirement prioritization.

---

**Table 5: Concept Table of All PDDs**

5 Template

Additionally to this review, the authors have created an excel spreadsheet to assist readers in requirements prioritization. The template includes a sample case that is identical to the method example in section 2.

**Figure 6: Template - Requirements Input Sheet**

**Figure 7: Template - Iteration Planning sheet**

---

8
References


