The Agile Requirements Refinery: Applying SCRUM Principles to Software Product Management

Kevin Vlaanderen, Sjaak Brinkkemper, Slinger Jansen
Department of Information and Computer Sciences
Utrecht University
Utrecht, the Netherlands
(k.vlaanderen,s.brinkkemper,s.jansen)@cs.uu.nl

Erik Jaspers
Planon B.V.
Nijmegen, the Netherlands
erik.jaspers@planon.nl

Abstract—Although agile software development methods such as SCRUM and DSDM are gaining popularity, the consequences of applying agile principles to software product management have received little attention until now. In this paper, this gap is filled by the introduction of a method for the application of SCRUM principles to software product management. For this purpose, the ‘agile requirements refinery’ is presented, an extension to the SCRUM process that enables product managers to cope with complex requirements in an agile development environment. A real-life case study is presented to illustrate how agile methods can be applied to software product management. The experiences of the case study company are provided as a set of lessons learned that will help in adapting the product management workflow to agile principles.

Keywords—software product management; requirements management; requirements refinery;

I. INTRODUCTION

One of the major innovations in software development methodology of the last few years has been the introduction of agile principles. Since the creation of the Agile Manifesto in 2001, including the years leading to its creation, several agile software development methods have come into practice [1]. Examples of such methods are DSDM [2], Extreme Programming [3], Feature Driven Development [4] and SCRUM [5]. The strong points of such methods are that by employing them, the development process becomes more responsive to a changing environment, working software is chosen over extensive documentation, individuals and interactions are considered more important than tools and processes, and customer collaboration is valued more than contract negotiation [6].

In the last few years, these agile methods have proven to be successful in a large number of cases. Companies that have put the agile method SCRUM [5] into practice range from small companies as described by Dingsøyr et al. [7] to large multinationals [8]. Research has shown that the use of SCRUM within a company can lead to significant benefits [9], and that its use is not limited to local projects [10].

As a consequence, demand for the extension of agile principles to other domains has risen. One such domain is software product management. Software product management (SPM) is the process of managing requirements, defining releases, and defining products in a context where many internal and external stakeholders are involved [11], [12]. The topic of SPM touches upon several other areas. In the related field of lifecycle management and release planning, several approaches have been proposed regarding market-driven requirements engineering [13], requirements interdependencies [14] and evolutionary and iterative release planning [15]. Another related field, requirements prioritization, has seen several publications in recent years, including work on requirements prioritizing for product software [16] and distributed prioritization [17].

Due to the complexity of software products, with a large variety of stakeholders, long lists of requirements and a rapidly changing environment, SPM is a complex task. However, relatively little scientific work has been performed in this area. An attempt to close this gap has been provided by Weerd et al. [11] in the form of a reference framework for SPM. Their work aims at providing a structure for the body of knowledge regarding SPM by identifying and defining the key process areas as well as the internal and external stakeholders, and their relations.

Currently, little work exists regarding agile SPM. A case study describing agile requirements engineering is described by Pichler et al. [18]. This research proposes an agile SPM method based on SCRUM, which improves the ability to handle complex requirements in an agile environment. Furthermore, a case study was performed at a product software company located in the Netherlands that has worked with the agile SPM method for nearly two years. By showing their experiences, a set of useful lessons learned is provided that aids in the implementation of SCRUM-inspired SPM alongside agile software development at other companies.

Section II continues with a description of the proposed SCRUM-inspired agile software product management process. In section III, an outline of the case study approach is given, including the research triggers, questions and meth-
II. AGILE SOFTWARE PRODUCT MANAGEMENT

This section describes a method for applying agile SPM in product software organizations that work according to agile software development practices. One should take into account that the proposed method is based on the default SCRUM process [5], developed initially for the purpose of (software) product development. Section II-A gives a short summary of the SCRUM development method, followed by the adaptations that have been applied to make the method applicable to SPM in section II-D.

A. SCRUM Development Method

The SCRUM development method was proposed in 1995 by Ken Schwaber [5], at a time when it became clear to most professionals that the development of software was not something that could be planned, estimated and completed successfully using the common ‘heavy’ methods. The SCRUM method is based on the work of Pittman [19] and Booch [20], and adheres to the principles of agile software development.

Central to SCRUM is the idea that many of the processes during development cannot be predicted. It therefore addresses software development in a flexible way. The only two parts that are fully defined during a software development project are the first and last phase (planning and closure). In between, the final product is developed by several teams in a series of flexible black boxes called ‘sprints’. No new requirements can be introduced during these sprints. This ensures that the final product is being developed successfully, even within a constantly changing environment. This environment, which includes factors such as competition, time and financial pressure, maintains its influence on development until the closure phase.

One of the central documents in the SCRUM method is the product backlog (PB). The PB contains a prioritized list of all items relevant to a specific product. This list can consist of bugs, customer requested enhancements, competitive product functionality, competitive edge functionality and technology upgrades [5]. Each team that participates in the software development process maintains its own development sprint backlog (DSB). All requirements that are assigned to the development team are put on their DSB. Every requirement is decomposed into several tasks, which are then assigned to specific team-members.

B. Agile SPM

The agile SPM process and SCRUM are similar in the aspects that they both work in sprints, and both developers and product managers perform tasks according to a product backlog and a sprint backlog. The difference is that at the end of a sprint developers produce a working version of the software, whereas the product managers produce a product backlog that feeds directly into the SCRUM process.

Figure 1 displays a visualization of the agile SPM process. It shows the default SCRUM development process, supple-
mented with the SPM-specific adaptations that have been added to it. The input for the SPM process is in most cases an idea or a wish for new functionality. This idea enters the process in the form of a **vision**, shown by the cloud at the bottom left of the figure. During a number of sprints, this vision is then refined several times, going through the agile requirements refinery, which will be discussed in the next section. The SPM teams select a set of PB items and place them on their PSB, after which the tasks on the PSB are performed during the length of the sprint. The main deliverable of this process is an updated version of the PB that can be used by the development teams to develop the software product. As a result of sprint review meetings held at the end of each sprint, new (retrospective) knowledge is gained that can help to improve the process. Finally, the figure includes bugs from earlier versions. These form an alternative way of generating PB items and do not follow the usual path through the requirements refinery, introduced in section II-D. Instead, they are placed directly on the PB.

Each working day, also known as a **scrum**, starts with a SCRUM-meeting during which the previous day is discussed. As this session is primarily meant to improve the productivity and the effectiveness of the SPM team, a small set of possible improvements is discussed. This helps avoiding experienced problems in the future.

The end-result of an agile SPM sprint, lasting between two and six weeks, consists of the requirements definitions used by the development teams. The sprint length is equal to the length of development sprints, in order to synchronize the heartbeat of the product management and the development process. For each sprint a product management sprint backlog (PSB) is composed for each SPM team. The PSB consists of tasks that can be finished by the SPM team within the sprint.

C. The Requirements Refinery

The structuring of the workflow into sprints and scrums enables agile SPM dealing with customer wishes. Similar to the SCRUM development method, no new items can be added to the PSB, as it has been finalized at the beginning of the sprint. This means that the SPM team(s) can focus on the work at hand without disruptions. On the other hand, it also requires considerable thought about the structuring of specific tasks, since they need to be completed within the timeframe of one month. SPM tasks, however, are not easily restructured into fine grained tasks of up to one month. For this reason, the default SCRUM-approach to task management has been substituted by the more fine-grained approach that is described in this paper.

This approach, the **agile requirements refinery**, provides a solution for managing large requirements. The approach is suited to the characteristics of SPM tasks, and it resembles an industrial refinery in a way that during each sprint or iteration work is being performed on the requirement definitions that appear on the PB, to refine them from coarse-grained to fine-grained. Each refinement, from one stage to the next, can generally be performed within one month. When this is not possible, the item is placed back on the PB to be picked up again in the future. Structuring the SPM tasks in such a way results in backlog items with a smaller granularity, suited for the length of a sprint. By refining large requirements according to the abstraction levels of the requirements refinery, structure is added to the backlog that will help in completing the tasks in an effective manner.

Since SCRUM itself does not provide guidelines for effectively managing large amounts of requirements of different granularity, a set of stages is introduced. Within the agile requirements refinery, a product functionality vision will generally move through these stages, during which it is refined with details and specifications. The stages are:

- **Vision** – A vision is the starting point for each product lifecycle. It is an idea, brought up by the company board, a customer or any other stakeholder, and is defined in generic terms. Once the idea reaches a product manager, he or she then converts it into a (set of) theme(s). An example of a vision is the wish to target small enterprises as potential customers for an ERP software package.

- **Theme** – A theme is the formal elaboration of a vision, describing it in more detail. The product manager defines the envisioned purpose of the new functionality, the business value of the theme and the involved stakeholders. A theme should briefly describe the business problem from which it originates and the main issues that fall within the theme scope. This can where possible be extended with a set of provisional requirements. In total, a theme description should not exceed one page of text, in order to maintain clarity. The previously described vision can for instance be translated to the theme ‘small enterprises’, describing its importance and what would be required to accomplish it. In reality, a vision is often so complex that it can be refined into multiple themes. To ensure the technical feasibility of a theme, it is reviewed by the development teams.

- **Concept** – Themes are broken down into smaller pieces called concepts. A concept is a high-level focal point within the theme, consisting of a set of solution stories that can later be used to deduct detailed requirements. The elaboration of each concept results in a document describing product drivers, product constraints and the concept scope. The description should on the one hand briefly explain the necessity of the concept, while on the other hand it should be clear and detailed enough to be able to use it for the definition of detailed requirements. The previous ‘small enterprise’ theme could for instance be converted to a set of concepts such as ‘productX Lite’, describing the high-level requirements...
of a software product suited to the needs of small enterprises. Each concept definition should be checked with the software architect(s).

- **Requirement definition** – The detailed definition of requirements is performed in three steps, of which only the first one is performed by the SPM team(s). SPM translates the concepts into a list of requirement definitions without going into a lot of detail. Requirement definitions consist up to this point of a description, a rationale and a fit criterion. The latter describes a constraint that must be met in order for this requirement to be successfully implemented. To ensure feasibility and compatibility with other requirements, each requirement definition should be checked with architects, functional designers or lead developers.

After the initial high-level requirement definitions have been determined based on the previously defined concepts, the software development teams then elaborate these into requirements containing a detailed description of some desired functionality, described in sufficient detail to work with. To accomplish this, each requirement definition is first processed during a development sprint by a development team, to ensure that they are feasible, consistent and understandable in a general manner. Then a second pass is made, where the development team ensures requirement clarity, so that each requirement is understood by all team members. This results in a list with all relevant requirements and their detailed descriptions, including any necessary diagrams, technical specifications or otherwise necessary information that is required for the implementation of the requirement.

With smaller topics, the definition of a vision and a theme might not be necessary, in which case the problem can be placed within an existing theme or concept. They are then elaborated without constructing a vision, theme and/or concept, or they are elaborated with the vision, theme and concept constructed afterwards.

### D. SCRUM SPM Process

Figure 2 shows a view of the SCRUM SPM process, based on the meta-modelling technique by van de Weerd [21]. In the figure, the deliverable side has been omitted in order to focus on the process aspect. Its notation is based on a UML activity diagram. Standard activities and sub-activities are depicted by white boxes, and open activities are shown by gray boxes. Arrows are used to show the control flow from one activity to the next. The top part of the activity diagram, indicated by a light gray box, will recur several times within each SPM sprint, once for each requirement.

At the start of each sprint, each SPM team has to prepare its PSB. The teams make a selection of PB items, of which they think that they can be completed within the upcoming sprint. This activity is similar to the sprint preparation as performed by the development teams.

The next step is to proceed with either refining the items that are on the PSB, or introducing new ideas obtained through customer support, meetings with business consultants, customer sessions, industry analysts and involvement at different types of forums in which market parties are active. During a sprint, each item is refined from its current stage to the next level of detail, i.e. from vision to theme or from concept to requirement definition.

When a vision enters the process, it is described globally, after which one or more themes are derived from it. Each theme is described according to the specification in II-D. When the description is finished, the required investment needed to implement the team is estimated. The theme is then reviewed by the development teams, after which it is placed on the PB.

Concept specification starts with breaking down the theme into a set of concepts. Every concept contains a set of solution stories which are used for defining detailed requirements. The concepts are defined by product managers and software architects. Again, an estimation is made regarding the required investment for implementation. After concepts
are defined, they are reviewed by software architects and domain experts.

If a concept is approved, the concept is broken down into a set of requirement definitions. A requirements engineer and a SCRUM development team are responsible for the definition of requirements. Requirements can be broken down into smaller pieces to fit into a sprint. Requirements are also assigned a priority, after which they are put on the PB. The highest rated requirements are to be developed first. This priority rating is assigned by the product board and the sales department, and the requirement definitions are reviewed by lead developers, architects, functional analysts and domain experts. In some cases, requirement definitions are rejected due to being unclear or because they are not described in sufficient detail. In such cases, the requirement definition needs to be further specified.

When requirement definitions are approved, the costs and business value are calculated. Each requirement is valued and prioritized, after which the ordered list is placed on the PB, where it is used for deciding when features will be developed. After prioritizing the requirements, time is allocated to each requirement or concept to allow the determination of a sprint planning. When requirements are clear and have enough detail they are assigned to the development teams. The requirements are then placed in the DSB of the specific development team.

After each completed SPM sprint, an evaluation takes place. During this evaluation, each team looks back at the last sprint, and discusses about the aspect that went good or wrong. The results are written down, and from the resulting list, two or three items are chosen to be put on the sprint backlog of the next SPM sprint. This enables the teams to gradually improve the process, learning not only from their own mistakes, but also from those of the other teams.

E. SPM Sprint

The agile aspect of the proposed SPM approach lies mainly in the fact that, like software development, the SPM task is performed according to sprints with a fixed length of two to six weeks (varying per company). These sprints are not performed synchronously to the software development sprint, but are shifted back half of the development sprint duration. This ensures that the DSB is always up-to-date and ready for use once the software development sprint starts, reducing the time between the inception of a requirement and its realization in the product.

Figure 3 illustrates this concept of alternating sprints. The horizontal timeline shows the synergy between software product management and software development, by switching from a focus on SPM to a focus on software development and back. The SPM team(s) deliver(s) an updated PB while the development teams are developing the next product release candidate (depicted by a floppy). Based on this release candidate, SPM will then redefine the PB, resulting in continuous double-loop feedback.

Similar to SCRUM software development, the status of completed, canceled or ongoing tasks is continuously kept up-to-date on the PSB. Also, a burn-down chart is created continuously to allow monitoring of the progress of the sprint. Another similarity is that the PSB is filled with items from the PB at the beginning of each sprint.

III. CASE STUDY RESEARCH APPROACH

The main research question to this research is How should software product management be performed in a SCRUM development context? The research question is answered by developing a method for agile SPM using design research. Furthermore, the method is tested in a case study of a production environment in a product software company in the Netherlands. During the case study data has been collected to answer the research questions [22]. The means through which these facts were gathered are:

- **Interviews** – The main research questions have been answered in part during the unstructured interviews with product stakeholders. We interviewed one requirements engineer, two product managers and the chief technology officer. These interviews were recorded, and information regarding the SPM process and issues related to it were extracted later on.

- **Document study** – The case company provided us with guideline documents such as the altered Volere requirements specification template [23] that is in use at the case company, the product backlog and the sprint backlogs for the SPM team. These documents were added to our case study database. Some of the filled-in Volere templates were used to gain understanding about the relation between the PSBs. The PSBs itself were used for a quantitative analysis. A final qualitative analysis was performed on the obtained PSBs to obtain further insight in the practical consequences of agile SPM and to extract some examples.

The information gathered from these sources has been combined in order to create the set of lessons learned that...
is presented in the final section.

A. Case Study Company: Planon

The main contribution of this work lies in the description of a unique case among Dutch product software companies, and potentially among product software companies in general. The company at which the case study has been performed, Planon International (from now on referred to as FacComp), has, as one of the first known companies, attempted to implement an agile SPM process based on the agile principles in general (and the SCRUM development method specifically). The decision to switch to an agile SPM approach stemmed from their positive experiences with agile development.

FacComp is an international software vendor that produces Facility Management and Real Estate management software for organizations (IWMS: Integrated Workplace Management Systems). Founded in 1984, it currently has a customer base of over 1300, which is supported by more than 325 employees. The company’s products are marketed through six subsidiaries, based in the Netherlands, Belgium, Germany, UK, India and the US, and a worldwide network of partners. The company made approximately 1.9 million profit with a revenue of 30 million in 2008. FacComp develops client-server software (two- and three-tier architectures) with which it attempts to support the processes of facility management.

B. Validity Threats

In order to ensure the quality of our work, we have tried to adhere to four validity criteria for empirical research. The validity threats are construct, internal, external, and reliability threats [22], [24]. Construct validity refers to the proper definition of the concepts used within the study. For this study, well-established concepts were used to construct theories. These theories were established in a discussion session at the beginning of the project. Since well-known concepts were used to describe novel phenomena, construct validity is guarded. Furthermore, peer review was used to check whether the constructs were used correctly. The internal validity, which concerns relations between concepts, was threatened by incorrect facts and incorrect results from the different sources of information. Interviews were held with several people in order to cross-check documentation found and to confirm facts stated in other interviews.

With respect to external validity, concerning the ability to generalize the results, a threat is that this case is not representative for other software producers working with SCRUM. FacComp is a standard product software supplier, which deals with a lot of new requirements. The practices described in this paper can be a successful way to manage teams of product managers for similar sized software vendors working with SCRUM. Finally, to defend reliability, similar results would be gathered if the case study was redone if the circumstances are at least similar (same interviewees, same documents, etc), due to the use of a case study protocol, structured interviews, and a peer-reviewed research process [24].

IV. SPM Sprint Backlog Analysis

To understand the actual workings of the SPM adaptation of SCRUM, an analysis of the PSBs was needed to gain a more detailed view of the results and implications of FacComp’s adaptations of the SCRUM-process in order to accommodate SPM. The analysis consists of a data-set of twenty-one PSBs, describing an equal amount of months. The PSB’s have been gathered from March 2007, when SCRUM was introduced into the SPM process, until November 2008.

The analysis focused both on general statistics about the task structure, including task duration and workload per person, as well as on pattern discovery. Table I displays statistics about the sprints included in our study. From left to right, the table first shows the number of tasks that were placed on the PSB in that month, the total amount of planned hours for those tasks and the amount of unfinished work at the end of the sprint. Subsequently, the table shows the average amount of hours per task, the average workload per person expressed in amount of tasks and the average workload per person expressed in hours. The final column shows an effectivity-score, obtained by calculating the reduction in hours assigned to all the tasks. The bottom three rows show statistics about the lowest, the highest and the average score for all items.

Over the two years of experience, the PSBs provide information regarding the number of tasks and their characteristics. The PSBs provide insight into two years’ evolution of the number of tasks and their characteristics. First, several recurring, standard backlog items can be identified. Second, the evolution and introduction of the requirements refinery can be followed from the first introduction of themes, concepts and requirement definitions. The abstraction levels of the refinery (i.e. themes, concepts and requirement definitions) make large requirements more manageable in an agile environment. To illustrate, two themes will be tracked through the entire SPM process.

A. Standard Backlog Items

From the PSB, several standard recurring backlog items can be identified. The standard items, as opposed to incidental tasks, form a base structure of recurring tasks, mostly with the same amount of hours allocated each sprint. These tasks can be used to create a form of rhythm within the team(s).

At the case company, the list of standard backlog items has evolved during the reported period from a disorganized list into a stable list of tasks, shown in table II. On the lefthand side, all standard backlog items related to the SPM
### Table I

<table>
<thead>
<tr>
<th>#</th>
<th>Period</th>
<th>Tasks</th>
<th>Hours Start</th>
<th>Hours End</th>
<th>Avg Hours per Task</th>
<th>Avg Tasks per Person</th>
<th>Avg Hours per Person</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2007-03</td>
<td>45</td>
<td>449</td>
<td>154</td>
<td>10.0</td>
<td>9.0</td>
<td>89.8</td>
<td>65.7%</td>
</tr>
<tr>
<td>2</td>
<td>2007-04</td>
<td>57</td>
<td>513</td>
<td>177</td>
<td>9.0</td>
<td>11.4</td>
<td>102.6</td>
<td>65.5%</td>
</tr>
<tr>
<td>3</td>
<td>2007-05</td>
<td>56</td>
<td>527</td>
<td>147</td>
<td>9.4</td>
<td>11.2</td>
<td>105.4</td>
<td>72.1%</td>
</tr>
<tr>
<td>4</td>
<td>2007-06</td>
<td>57</td>
<td>539</td>
<td>102</td>
<td>9.5</td>
<td>11.4</td>
<td>107.8</td>
<td>81.1%</td>
</tr>
<tr>
<td>5</td>
<td>2007-07</td>
<td>112</td>
<td>409</td>
<td>179</td>
<td>3.7</td>
<td>22.4</td>
<td>81.8</td>
<td>56.2%</td>
</tr>
<tr>
<td>6</td>
<td>2007-08</td>
<td>95</td>
<td>574</td>
<td>301</td>
<td>6.0</td>
<td>19.0</td>
<td>114.8</td>
<td>47.6%</td>
</tr>
<tr>
<td>7</td>
<td>2007-09</td>
<td>81</td>
<td>550</td>
<td>215</td>
<td>6.8</td>
<td>16.2</td>
<td>110.0</td>
<td>60.9%</td>
</tr>
<tr>
<td>8</td>
<td>2007-10</td>
<td>58</td>
<td>750</td>
<td>274</td>
<td>12.9</td>
<td>11.6</td>
<td>150.0</td>
<td>63.5%</td>
</tr>
<tr>
<td>9</td>
<td>2007-11</td>
<td>53</td>
<td>544</td>
<td>188</td>
<td>10.3</td>
<td>10.6</td>
<td>108.8</td>
<td>65.4%</td>
</tr>
<tr>
<td>10</td>
<td>2007-12</td>
<td>43</td>
<td>324</td>
<td>142</td>
<td>7.5</td>
<td>7.2</td>
<td>54.0</td>
<td>56.2%</td>
</tr>
<tr>
<td>11</td>
<td>2008-01</td>
<td>60</td>
<td>648</td>
<td>193</td>
<td>10.8</td>
<td>10.0</td>
<td>108.0</td>
<td>70.2%</td>
</tr>
<tr>
<td>12</td>
<td>2008-02</td>
<td>67</td>
<td>497</td>
<td>226</td>
<td>7.4</td>
<td>11.2</td>
<td>82.8</td>
<td>54.5%</td>
</tr>
<tr>
<td>13</td>
<td>2008-03</td>
<td>65</td>
<td>604</td>
<td>250</td>
<td>9.3</td>
<td>10.8</td>
<td>100.7</td>
<td>58.6%</td>
</tr>
<tr>
<td>14</td>
<td>2008-04</td>
<td>84</td>
<td>669</td>
<td>320</td>
<td>8.0</td>
<td>14.0</td>
<td>111.5</td>
<td>52.2%</td>
</tr>
<tr>
<td>15</td>
<td>2008-05</td>
<td>100</td>
<td>619</td>
<td>248</td>
<td>6.2</td>
<td>16.7</td>
<td>103.2</td>
<td>59.9%</td>
</tr>
<tr>
<td>16</td>
<td>2008-06</td>
<td>121</td>
<td>586</td>
<td>198</td>
<td>4.8</td>
<td>20.2</td>
<td>97.7</td>
<td>66.2%</td>
</tr>
<tr>
<td>17</td>
<td>2008-07</td>
<td>85</td>
<td>530</td>
<td>208</td>
<td>6.2</td>
<td>14.2</td>
<td>88.3</td>
<td>60.8%</td>
</tr>
<tr>
<td>18</td>
<td>2008-08</td>
<td>91</td>
<td>443</td>
<td>197</td>
<td>4.9</td>
<td>15.2</td>
<td>73.8</td>
<td>55.5%</td>
</tr>
<tr>
<td>19</td>
<td>2008-09</td>
<td>82</td>
<td>523</td>
<td>176</td>
<td>6.4</td>
<td>13.7</td>
<td>87.2</td>
<td>66.3%</td>
</tr>
<tr>
<td>20</td>
<td>2008-10</td>
<td>81</td>
<td>543</td>
<td>116</td>
<td>6.7</td>
<td>13.5</td>
<td>90.5</td>
<td>78.6%</td>
</tr>
<tr>
<td>21</td>
<td>2008-11</td>
<td>91</td>
<td>455</td>
<td>129</td>
<td>5.0</td>
<td>15.2</td>
<td>75.8</td>
<td>71.6%</td>
</tr>
<tr>
<td></td>
<td>Avg</td>
<td></td>
<td>75.4</td>
<td></td>
<td>537.9</td>
<td>197.1</td>
<td>7.7</td>
<td>13.5</td>
</tr>
<tr>
<td></td>
<td>Lowest</td>
<td></td>
<td>43.0</td>
<td></td>
<td>324.0</td>
<td>102.0</td>
<td>3.7</td>
<td>7.2</td>
</tr>
<tr>
<td></td>
<td>Highest</td>
<td></td>
<td>121.0</td>
<td></td>
<td>750.0</td>
<td>320.0</td>
<td>12.9</td>
<td>22.4</td>
</tr>
</tbody>
</table>

Table I

**RESULTS OF THE PSB ANALYSIS**

sprint are shown. On the righthand side, all standard backlog items related to the development sprint are shown. All tasks are performed by the SPM team(s).

As described earlier, the PSBs were at first mainly structured in a product-focused manner. As a result, recurring backlog items were spread across the PSB, resulting in a disorganized list which had to be recreated from scratch every month. As of month five, a small list of recurring backlog items related to the product board is distinguished. However, this is comprised of only ninety hours. This list grows to a set of five different tasks (of which some occur multiple times, once for each product manager), with a total amount of 268 planned hours. This list stays relatively stable until month fifteen, in which the new PSB structure is introduced. At that moment, the list of standard backlog items is reduced to two tasks with a total amount of 72 hours. Remarkable is the steady growth of this list in the next six months, after which the list of standard backlog items consists of six different tasks, similar to the tasks of the earlier months, totaling only 80 hours. The final list of standard backlog items is shown on the lefthand side of table II.

The low amount of planned hours can be explained by taking into account the introduction of SCRUM principles. At the same time as the introduction of the new PSB structure, a new group of tasks has been introduced on the list, containing all the tasks related to the management of the upcoming development sprint. Although the exact contents of the group differ every sprint, a large share of the tasks is recurring and thus added to the righthand side of table II.

**B. Themes, Concepts & Requirements**

The introduction of the requirements refinery, described in section II-D, was not performed in one step. Although the term concept is found on the PSB as of month 2, themes are introduced for the first time in month 15. Regarding the concepts, several interesting notes can be made. The PSBs show a clear evolution in the use of the term. In the first few months, the sub-list 'concepts' contains an aggregation of tasks related to concept-elaboration in general. As of month five, tasks within the concepts-list are grouped according to the specific concept that they belong to, thus providing an early indication of the corresponding theme. Further
elaboration of product features is displayed under a products-list within the PSB.

The switch to the requirements refinery in month 15 has clear effects on the backlog. Most notable is the immediate structure and clarity that is created by this change. By dividing the tasks related to the elaboration of requirements into lists named 'theme definition', 'concept definition' and 'requirement elaboration', a clearer overview of the workload is obtained. For every task it becomes instantly clear in what phase of elaboration the requirement currently is.

Another consequence of the approach can be seen in two trends in the evolution of task size and amount. On the one hand, the amount of tasks on the PSBs increased approximately twofold, whereas the average size of the tasks decreased with approximately 50%. Evidence on the PSBs suggests a relation with the introduction of themes and concepts on the PSB, as larger tasks such as 'describe requirements' are now split into smaller tasks, specific to the current stage.

C. Illustration: Maintenance Planning

To illustrate the specific workings of themes, concepts and requirements within the PSB, the maintenance planning theme is followed throughout its evolvement. The theme was introduced in 2008, when the case company chose to achieve a redefinition of its existing maintenance management solutions. The theme describes functionality related to the maintenance of facilities, and was initially introduced on the PB in month fourteen of the analysis. The entire SPM lifecycle of the theme lasts seven months.

Although the theme elaboration has not been documented in the PSBs, the theme is elaborated into several concepts. These concepts are 'planned maintenance (PM)', 'planned preventative maintenance (PPM)' and 'maintenance management (MM)'. Besides these, several other concepts exist that fall partially within this theme, such as 'work orders' and 'asset'. Each of these concepts is described in several documents, of which the 'vision, scope & requirements' document is the most detailed. Within the theme, a focal transition is visible from 'planned preventative maintenance' to 'planned maintenance'. Furthermore, 'maintenance management' is introduced in a later stage. For this example, the focus lies on the concepts of 'planned maintenance' and 'maintenance management'.

The 'planned maintenance' concept was introduced in month fourteen, right before the introduction of the new PSB structure. Analysis of the PB shows that the concept of 'planned maintenance' was elaborated into 152 requirements, subdivided over the groups 'no value', 'contract', 'generic' and 'maintenance planning'. Although a theme-section was not yet available in the PSB of that month, it is clear from the task-descriptions that they are related to theme-level requirements. During the next five months, the tasks related to the theme should shift from theme-level towards requirements-level. However, evidence from the PSB shows that 'planned maintenance' tasks are placed under the requirements section right away. These tasks are concerned with the detailed elaboration of requirements, and would thus be expected later in the process.

This is slightly different for the initial tasks related to 'maintenance management' (i.e. the other concept within the 'maintenance planning' theme). The tasks related to the concept can be found on the PSBs for the first time in month fifteen of our analysis, at the same time as the introduction of the new PSB structure, and for the last time in month twenty. These tasks are, similar to 'planned maintenance' tasks, initially placed on the theme-level. As the concept matures, task-focus moves towards the concept-level and finally towards requirements elaboration, analogous to the 'theme/concept/requirements' lifecycle.

D. Illustration: Planon Lite

As shown in the previous section, the introduction of themes, concepts and requirements on the PB does not necessarily mean that all ideas brought up within the company follow the same, complete track through all phases. Although it is recommended to do so with large, complicated themes, the previous section has shown that it is possible and perhaps more efficient to take a 'shortcut'.

At the same time, introducing a more fine-grained notation also does not mean that every theme or concept will make it through all the steps of the requirements refinery. In fact, the added detail allows for an increased visibility of theme life cycles, which can result in the deletion of certain themes or concepts from the backlog. An example of this, the lifecycle of a new product idea is described, coined within the company in the fifth month of our analysis. This concept, called Planon Light, aimed at providing smaller companies with facility management services. The concept started out as an idea with a set of tasks related to the elaboration of the concept-vision. After this vision was created, it was discussed and revised, after which it had to be reviewed by the CIO. However, as the priority of this task was not high, it remained on the PSB for several months. Only in month eleven is the task completed, after which the Planon Light concept disappears from the backlog, indicating a rejection of the concept.

This example shows two important points. Firstly, it shows that not all features start out at the theme-level. As indicated before, only complex features are considered themes, whereas smaller features can be directly translated to concepts or requirements. Secondly, the fact that tasks keep recurring on the backlog indicates that basic SCRUM principles can be successfully translated to the product environment process, adding more clarity and structure.
V. LESSONS LEARNED

During its attempts to implement an agile SPM method, our case company has gained valuable experiences in this area. These experiences, which have mostly been mentioned in the previous sections, are listed in this section as a set of lessons that should be taken into account when implementing agile SPM alongside an agile software development method.

- **Alternating cycles for SPM and Development** – One of the main lessons learned has been the importance of the alternating sprints. As discussed in section II-E, the software development and the SPM sprint are both performed continuously, but with a difference in starting date of approximately half of the sprint duration. This implies that each SPM sprint ends halfway the software development sprint, ensuring that the PB is ready to be used when the development teams start their new sprint.

- **Daily SCRUM meetings are essential** – The daily stand-ups, or SCRUM meetings, that are essential within the SCRUM development method, are also valued highly within the agile SPM method. The fifteen-minute meeting at the start of each day is experienced as a positive, helpful aspect of the process. By providing constructive critique, potential problems can be avoided and existing problems can be solved.

- **Complex requirements are in need of structured detailing** – The essence lies in the division of requirements into themes, concepts and requirements. The structured agile requirements refinery approach has made it possible to effectively manage large sets of requirements of different granularity. Both high level and low level requirements are placed on the PB and handled in time by the appropriate person.

- **Backlog administration requires discipline** – We have seen that strict documentation of all tasks on the PSB is still difficult to achieve. Although the PSB can play a useful role in controlling the SPM process and keeping track of the progress of a sprint, the motivation to keep the current set of tasks and the amount of time spent on a specific task up-to-date is still lacking. However, it should be noted that one of the agile principles is a favoring of individuals and interactions over processes and tools. This means that, as long as the work gets done, project administration becomes less important.

- **Early collaboration promotes reuse and integration** – Since product managers in a SCRUM team cooperatively work on a PSB and discuss requirements before they have been implemented, re-use and integration opportunities can be spotted at an early stage. We suspect that higher quality software products are built using this approach than other approaches with less communication during the requirements specification process.

VI. CONCLUSIONS AND OUTLOOK

Up until now, no attempts to apply agile principles to the SPM process had been described in literature. In this situation we have proposed such a method based on the proven structures of a well-adopted agile development method. By providing the lessons that have been learned during this process, it is our hope that other companies can benefit from the experience of the case study company and that other researchers can apply and measure the effects of the requirements refinery.

In the description of the SCRUM development method we have shown that an agile development process implies an environment that is dynamic and to which it is constantly adapting, be it in a controlled, effective way. It is not hard to imagine that such a dynamic development environment requires an SPM process that is adequately adapted to this. The effect of this is an increased demand for agile SPM processes, of which one has been described in this work. The main contribution of this work has been the description of an innovative SPM process based on agile principles. The textual description along with process-deliverable diagrams both for the software development as well as the SPM processes allows effective reuse of the described method in other companies that find themselves in a comparable situation.

The experiences of the case study company have shown that, to ensure effective agile SPM, several factors should be taken into account, such as task size, backlog structure and willingness to keep the backlog up-to-date. By providing the specific lessons that FacComp has learned during its experience with agile SPM and SCRUM, we allow companies that wish to implement agile SPM to circumvent potential problems related to these items. However, as stated before, not a lot of research has been performed in the area of agile SPM. Future research should be aimed at further elaboration and formalization of the requirements of agile SPM processes. Besides this, more information should be gathered regarding current implementations of agile SPM processes, and their integration with agile development.

ACKNOWLEDGEMENTS

We would like to thank Planon for sharing its experiences, and for providing us with all required documents. Also, we would like to thank Jaap Kabbedijk for his input during the writing of this paper and Tjan-Hien Cheng for his contribution to the work.

REFERENCES


