The agile requirements refinery: Applying SCRUM principles to software product management

Kevin Vlaanderen a,⇑, Slinger Jansen a, Sjaak Brinkkemper a, Erik Jaspers b

a Department of Information and Computer Sciences, Utrecht University, Utrecht, The Netherlands
b Planon B.V., P.O. Box 38074, 6503 AB, Nijmegen, The Netherlands

ARTICLE INFO

Article history:
Received 20 August 2010
Accepted 23 August 2010
Available online 31 August 2010

Keywords:
Software product management
Requirements management
Requirements refinery
Agile development
SCRUM

ABSTRACT

Context: 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.

Objective: In this paper, this gap is filled by the introduction of a method for the application of SCRUM principles to software product management.

Method: A case study research approach is employed to describe and evaluate this method.

Results: This has resulted in the ‘agile requirements refinery’, an extension to the SCRUM process that enables product managers to cope with complex requirements in an agile development environment. A case study is presented to illustrate how agile methods can be applied to software product management.

Conclusions: The experiences of the case study company are provided as a set of lessons learned that will help others to apply agile principles to their software product management process.

© 2010 Elsevier B.V. All rights reserved.

1. 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].

Due to its success, demand for the extension of agile principles to other domains has risen. One such domain is software product management. The original area of product management can be defined as “the discipline and role, which governs a product (or solution or service) from its inception to the market/customer delivery in order to generate biggest possible value to the business” [11]. Software product management (SPM) is then “the process of managing requirements, defining releases, and defining products in a context where many internal and external stakeholders are involved” [12,13]. The topic of SPM touches upon several other areas. In the related fields of lifecycle management and release planning, several approaches have been proposed, including market-driven requirements engineering [14] and requirements interdependencies [15]. A systematic review of release planning approaches has been given by Svaanberg et al. [16]. Another related field, requirements prioritization, has seen several publications in recent years, including work on requirements prioritizing for product software [17] and distributed prioritization [18].

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 van de Weerd et al. [12] 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. Some recent work related to the specific areas of the framework includes the use of feature modelling for handling variability throughout the product line lifecycle [19], and key success factors for pricing software products [20]. Another important addition to the field has been the QUPER model, developed for handling non-functional requirements [21,22].

Currently, little work exists regarding agile SPM. A case study describing the use of agile requirements engineering is described by Pichler et al. [23]. However, the paper does not provide details regarding the agile requirements engineering process. An attempt to link long-term product planning and agile development is provided by Vahathan and Rautiainen [24]. Greer and Ruhe elaborate on agile release planning by providing an iterative optimization method [25]. Collaboration between product managers and development teams in challenging environments, such as where no complete requirements are available, is investigated by Fricker et al. [26]. In a comparative case study by Fogelström et al. [27], a misalignment was identified between the agile principles and the needs of pre-project activities in market-driven development. They state that the differences between agile methods and the needs of market-driven software development may threaten product development by disabling effective product management.

In order to fill the remaining gap, we will describe in which way software product management can be performed in a SCRUM development context. The research described in this paper proposes an agile SPM method based on SCRUM, which improves the ability to handle large amounts of 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 2 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 2 continues with a description of the proposed SCRUM-inspired agile software product management process. In Section 3, an outline of the case study approach is given, including the research triggers, questions and methods, and a description of the validity threats. This is followed by an introduction to the product software company at which the proposed method has been used. In Section 4, the results of the case study are shown in the form of an analysis of the tasks within the process. The validity threats regarding this case study are shown in Section 3.3. To conclude, Section 5 contains a description of the lessons learned regarding agile SPM and Section 6 provides conclusions and suggestions for future research.

2. 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. The proposed method is based on the default SCRUM process [5], developed initially for the purpose of software development. Section 2.1 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 2.3.

2.1. 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 [28] and Booch [29], 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 with a high probability of success, 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.

The backlog is an important instrument in the SCRUM process. The following backlogs play a part in SCRUM development:

- **Product Backlog (PB):** The PB is central to the SCRUM method. 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]. Once a requirement has been fully specified, with the approval of a developer, the requirement can be copied from the PB onto the Development Sprint Backlog.
- **Development Sprint Backlog (DSB):** Each team that participates in the software development process maintains its own DSB. All requirements that are assigned to the development team at the beginning of a sprint are put on their DSB. Every requirement is decomposed into several tasks, which are then assigned to specific team-members. The Development Sprint Backlog is fed by the product backlog with items that have been fully specified.

The DSB enables continuous monitoring of the progress of developers and development teams, while the PB enables monthly renegotiation about the priorities for each requirement.

2.2. Agile SPM

The development of software by large teams of developers requires a steady flow of elicited product requirements. Without this steady flow of requirements, software vendors run the risk of delaying new software releases and bad code due to badly specified requirements, all resulting in the waste of large amounts of resources. To avoid these problems, a functioning team of product managers is required, that can, cooperatively with the development team, supply approved and well-defined requirements. The agile SPM method applies SCRUM to maintain a steady flow of new requirements for the DSB. Furthermore, agile SPM enables a software vendor to flexibly define requirements according to a pre-defined procedure. The pre-defined procedure forces a software vendor to explicitly manage the lifecycle of a requirement, leading to better-defined requirements. Simultaneously, the process remains agile, i.e., some requirements can be defined and implemented quickly, while others move through their lifecycle at a regular pace.

Fig. 1 shows the flow of knowledge within the agile SPM process. The figure is based on the default SCRUM development process described in the previous section, and is supplemented with SPM-specific adaptations. In the figure the product management sprint backlog is introduced. The PMSB is an agile SPM concept. It provides product managers with a way of working similar to that
of developers in the SCRUM process, using PMSB items to establish division of work, and work planning.

- **Product Management Sprint Backlog (PMSB):** The PMSB contains all items that need to be completed within the sprint by each product manager. The PMSB is fed by with items from the product backlog, the full list of concepts, themes, and requirements for a product. The PB feeds the PMSB with items that need further specification before they can enter the DSB.

SCRUM and the agile SPM process are similar in the aspects that they both work in sprints, and that both developers and product managers perform tasks according to the shared PB and a team backlog. The main difference is that at the end of a sprint, developers produce a working version of the software, whereas product managers produce clearly specified requirements. Once these well-defined requirements are approved and prioritized by the product owners, the requirements are put onto the DSB. Table 1 lists the differences between the agile SPM process and SCRUM.

The input for the agile SPM process is in most cases an idea or a wish for new functionality, but also new technologies, bugs and upgrades. 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. Based on the amount of time available and the focus determined by the board, the SPM teams select a set of PB items, such as concepts and visions, and place them on their PMSB. During the length of a product management sprint, specification tasks are performed on these PB items. The main result of this process is a list of further specified themes, concepts, and requirements that can be placed back onto the PB. The requirements that have been fully specified and approved by a software developer are candidates for the next development sprint. 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.

The agile SPM process also 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. Instead, they are placed directly on the PB. If the bug can be fixed easily, it goes straight to the DSB. If the bug cannot be fixed easily, however, it will go onto the PMSB, for review and further detailing by the product management team.

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 consists of the requirements definitions, which can in turn be 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.

<table>
<thead>
<tr>
<th>Differences between SCRUM development and agile SPM.</th>
<th>PMSB</th>
<th>DSB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Takes work from..</td>
<td>Product backlog (PB)</td>
<td>Product backlog (PB)</td>
</tr>
<tr>
<td>Demands..</td>
<td>Vision (unspecified requirements), bugs</td>
<td>Specified requirements</td>
</tr>
<tr>
<td>Supplies..</td>
<td>Specified requirements</td>
<td>Functional software</td>
</tr>
<tr>
<td>Deals with..</td>
<td>Visions, concepts, themes, requirements</td>
<td>Bugs, product enhancements, functionality, technology, upgrades, etc.</td>
</tr>
<tr>
<td>Works in..</td>
<td>Sprints and daily scrums</td>
<td>Sprints and daily scrums</td>
</tr>
<tr>
<td>Worked on by..</td>
<td>Product managers</td>
<td>Developers</td>
</tr>
<tr>
<td>Puts back onto PB..</td>
<td>Requirements definitions</td>
<td>Finished PB items</td>
</tr>
</tbody>
</table>
There are three stakeholders in the agile SPM process. First and foremost, the product managers’ work process is the one determined by the agile SPM process. Secondly, the product board, consisting of the CEO, the support director, the business consultancy director, the development director, and several representatives from sales departments, determines requirements priority and product vision in a monthly meeting. The development teams increasingly monitor and approve requirements as they come closer to entering the DSB.

2.3. The requirements refinery

The structuring of the workflow into sprints and scrums enables agile SPM dealing with customer wishes. Similar to the SCiRM development method, no new items can be added to the PMSB, 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 sprint. SPM tasks, however, are not easily restructured into fine-grained tasks of up to 1 month. For this reason, the default SCiRM-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 complex 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 1 month. When this is not possible, the item is placed back on the PB to be picked up again in one of the future sprints. 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 complex 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 SCiRM 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 the lifecycle of most requirements. 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 with a light version.
- **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 briefly explain the necessity of the concept, while remaining clear and detailed enough to be useful for the definition of detailed requirements. The ‘small enterprise’ theme could for instance be converted to a set of concepts such as ‘productX Lite’, describing the high-level requirements of a product suited to the needs of small enterprises. Each concept definition should be checked with the software architects. Also, the developers help estimate whether the concept is sufficiently defined to further split up the concept into requirements.
- **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. In other words, the requirements refinery is not restricted to a top-down approach, but can also be used bottom-up. This is similar to the approach by Gorschek and Wohlin [30], who identified four abstraction levels on which a requirement can be placed, along with both a bottom-up and a top-down path along these levels. Also very similar, but with a chief aim of maintaining relationships between high-level and low-level goals during development, Vähäniitty and Rautiainen [24] identified the stages vision, business goal, iteration goal, backlog item and task.

2.4. Process description

Fig. 2 visualizes the agile SPM process, based on the process-deliverable diagramming technique by van de Weerd et al. [31]. 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
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 PMSB (1). 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 PMSB (2), or introducing new ideas (3) 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 (4) from it. Each theme is described according to the specification in Section 2.3. When the description is finished, the required investment needed to implement the team is estimated. Themes are then reviewed by the development teams, after which they are placed on the PB.

Concept specification (5) 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 requirement definitions (6). 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. They are also assigned a priority by the product board and the sales department, after which they are put on the PB. The highest rated requirements are to be developed first. 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 not being 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 (7). Each requirement is valued and prior-
It is necessary to decide on the order of the tasks based on the requirements. After prioritizing the requirements, the number of tasks to be completed is allocated to each requirement. This concept is applied to determine the order of tasks in a sprint planning. When requirements are clear and the details of the tasks are assigned to the development team, the tasks are placed in the DSB of the specific development team.

At the end of each completed SPM sprint, an evaluation (8) takes place, during which each team looks back at the last sprint, discussing what went right 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.

### 2.5. 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 2-6 weeks (varying per company). During the SPM sprint, the deliverables from one team are not always available in time for the other team’s new sprint.

Therefore, sprints should not be performed synchronously to the software development sprint. Instead, they should be 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. Also, information regarding implementation progress and the accuracy of requirements sizes and descriptions can flow back from the development teams to the SPM teams.

Fig. 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) delivers 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 PMSB is filled with items from the PB at the beginning of each sprint. The status of completed, canceled or ongoing tasks is continuously kept up-to-date on the PMSB. Each product manager is responsible for keeping the backlog up-to-date as the sprint progresses. An example of how this can be managed is an example excerpt of a

- **Interviews**: The main research questions have been answered in part during the unstructured interviews with product stakeholders. We interviewed the chief technology officer, two product managers and one requirements engineer. These interviews were recorded, and information regarding the SPM process and issues related to it were extracted later on.
- **Document study**: The company provided us with guideline documents such as the altered Volere requirements specification template [33] 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 PMSBs. The PMSBs itself were used for a qualitative analysis to obtain further insight in the practical consequences of agile SPM and to extract some examples.

The changes in the SPM approach at the case company were mainly investigated in retrospect. The interviews that formed an essential source for this research were performed at the start of 2009, at a point in time in which the case company already had several months of experience with agile software product management. Also, product management sprint backlogs up to that point were gathered. After this moment, the evolution of the approach was monitored by analyzing the product management sprint backlogs of the months after that. This has resulted in an overview of standard backlog items in Section 4.1 and an overview of relevant
roles and tasks in Section 4.2. The requirements refinery is then illustrated with two examples in Sections 4.3 and 4.4.

Based on the information from the backlogs and the interviews with the product stakeholders, we have derived the set of lessons learned that is presented in the final section. This list has been reviewed by the CTO at Planon to make sure that the most important items have been addressed accurately.

3.1. 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 Planon), has, as one of the first known companies, implemented an agile SPM process based on the agile principles in general (and the SCRUM development method specifically).

Planon is an international software vendor that produces Facility Management and Real Estate management software for organizations (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 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 in 2008. Planon develops client–server software (two- and three-tier architectures) with which it attained a revenue of 30 million in 2008. Planon is marketed through six subsidiaries, based in the Netherlands, Belgium, Germany, UK, India and the US, and a worldwide network of partners. 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.

3.2. Case study narrative

Until 2004, product development at Planon was based on the Prince2 method, after which a switch was made to SCRUM, thus following the ideas of the Manifesto for agile software development. This means that working software is delivered frequently, changing requirements are welcomed (even late in development) and teams reflect regularly on how to become more effective. Other agile ideas are that the course of a project unfolds in time, decisions are being made in a decentralized team-based way and there is a focus on early feedback.

Until 2007, as can be seen in Fig. 5, agile product development was accompanied by non-agile product management. Although several stages of elaboration were employed, no fixed cycles were used. This has as an effect that product managers did not manage to provide development with sufficiently detailed requirements before the start of each sprint. To improve this, the question had to be answered whether it was possible to base the management of the product backlog on SCRUM principles. This would imply a continuous adaptation of the product backlog to changing circumstances and a changing environment.

From there on, a new categorization emerged on the product backlog. Initially, the terms ‘concept’ and ‘theme’ began to be used to group items according to the specific topic that they belonged to. Further elaboration of product features was displayed under a products-list within the PMSB. After several months, this approach consolidated into a stable approach, in this paper defined as the requirements refinery.

To better understand the implications of the SPM adaptation of SCRUM, an analysis of the PMSBs was needed to gain a more detailed view of the results and implications of Planon’s adaptations of the SCRUM process in order to accommodate SPM. The dataset consisted of 28 PMSBs, describing an equal amount of months. The PMSB’s have been gathered from March 2007, when SCRUM was introduced into the SPM process, until July 2009.

The analysis focused mainly on general statistics about the task structure, including task duration and workload per person, as well as on pattern discovery. Table 2 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 PMSB 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 bottom two rows show statistics about the average score and standard deviation for all items.

Furthermore, we have checked the backlogs for any anomalies. Any anomalies we found were either removed from the dataset, or further analyzed based on the information received from the CTO at Planon. We then grouped the backlog items according to their characteristics. These groups have been checked with the chief technology officer, to make sure that they were correct.

Over the 2 years of experience, the PMSBs provide information regarding the number of tasks and their characteristics. The PMSBs provide insight into 2 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 complex requirements more manageable in an agile environment. To illustrate, two themes will be tracked through the entire SPM process, described in Sections 4.3 and 4.4.

The switch to the requirements refinery in month 15 had 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 PMSBs increased approximately 25%, whereas the average size of the tasks decreased with approximately 25%. Evidence on the PMSB’s suggests a relation with the introduction of themes and concepts on the PMSB, as larger tasks such as ‘describe requirements’ are now split into smaller tasks, specific to the current stage.

3.3. 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 [32,34]. Construct validity refers to the proper definition of the concepts used within the study. For this study, well established concepts from the area of agile methodologies and software product management were used to construct our theory on agile SPM. These theories were established in a discussion session at the beginning of the project. The terminology employed by the company was similar to that in standard SCRUM literature, and concepts from the area of software product management corresponded to those posed by Xu and Brinkkemper [35] and van de Weerd et al. [12]. 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 [36]. Planon 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 [34].

4. SPM sprint backlog analysis

4.1. Standard backlog items

From the PMSB, several standard recurring backlog items can be identified. The standard items, as opposed to incidental tasks, form a basic 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 3. On the left-hand side, all standard backlog items related to the SPM sprint are shown. On the right-hand 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 PMSBs were at first mainly structured in a product-focused manner. As a result, recurring backlog items were spread across the PMSB, 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 90 h. 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 15, in which the new PMSB structure is introduced. At that moment, the list of standard backlog items is reduced to two tasks with a total amount of 72 h. Remarkable is the steady growth of this list in the next 6 months, after which the list of standard backlog items consists of five different tasks, similar to the tasks of the earlier months, totaling only 80 h, which amounts to an average of 14.3% of the total amount of required hours each month. The final list of standard backlog items is shown on the left-hand side of Table 3.
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 PMSB 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 part of the tasks is recurring and thus added to the right-hand side of Table 3. The following items are recurring on the product backlog:

- Prepare and attend product board: The product board consists of several lead positions in the company, such as the CEO and the sales director, who have a major stake in the product itself. Once a month the product management team presents what has been developed and what the future will hold to the product board. The product board contributes in two ways. First, the product management team informs major product stakeholders of the progress of visions and plans that have an impact on the product. Secondly, the product management team is forced to report on their progress, which requires them to evaluate progress speed and SPM process quality.

- Sprint review: The sprint review consists of a full review of the PM sprint. Furthermore, the sprint review leads to an update of the internal and partner information portals of the product. These portals are used to report on the progress of the work and on the upcoming features for partners and sales teams.

- Team retro meeting: Once a month during the team retro meeting, the internal functioning of the PM team is discussed. The retro meeting does not specifically address practical problems, but tries to achieve better quality and use feedback to improve the Agile SPM process.

- Team allocation overview: Throughout the agile SPM process, themes are assigned to teams, consisting of a product manager and a development team. Generally, teams will remain active within that theme. However, when a certain set of requirements that originates from a certain theme can also be implemented by a team that has resources available, requirements sets might be transferred from one team to another during the team allocation overview.

- Problem and change management: The task of problem and change management deals with customer problems and large changes that require the interference of a product manager. Furthermore, product managers go through the list of reported problems from customers and respond within 1 month. The response to a reported problem generally consists of a decline, i.e., the problem will not be solved, or an accept, i.e., the problem will be inserted in the planning.
• **Sprint planning with development teams**: The sprint planning with development teams consists of an eight-hour meeting. During these meetings product managers and developers negotiate, accept, and approve PB items for the DSB. This is a typical part of the SCRUM process.

• **Sprint review with development teams**: During the sprint review the development teams present the functionality they have implemented to the other development teams. Developers also defend why the functionality is necessary.

• **How-to-demo stories**: Product managers create how-to-demo stories for the developers who are working within their theme. These how-to-demo stories are specified to indicate to the developers how they should demo the functionality they have implemented during the sprint review. The main reason for the creation of these stories is that developers frequently have a different view of the interesting parts of the functionality they have implemented.

These activities provide an overview of the different standard tasks that are executed monthly by the software product management team.

### 4.2. Roles and tasks

Although the identified recurring backlog items already form useful knowledge within a practical context, the link between the product management tasks and the actual execution of an agile SPM process is still missing. Therefore, we decided to further analyze the distribution of tasks over the product managers.

To do this, we analyzed the non-standard tasks per sprint, amounting to an average of 85.7% of the total allocated time per month. In doing this, we identified a set of roles within an agile Product Management team, each focusing on a specific set of tasks. Although no fully specialized roles were found, i.e. persons that handled only one or two kinds of tasks, each role has a characteristic combination of tasks assigned to it.

**Fig. 6** displays the amount of time spent by each member of the product management team on the types of backlog items. The horizontal axis shows the planned hours, not the hours that were actually spent on the tasks. On the vertical axis, all members are displayed.

In **Fig. 7** we have displayed another visualization of the task distribution, complementary to that of **Fig. 6**. First of all, the bar representing tasks related to ‘requirements elaboration’ is for a large share attributed to one person, PM#1. When compared to his bar in the previous figure, we can see that the majority of his time is actually spent on ‘requirements elaboration’, implying a position strongly focussed on this area.

A second observation that we can make is that a large share of the high-level theme-definition tasks is performed by PM#5. Also, he spends little time on low-level activities related to ‘requirements elaboration’. This indicates a position with more power and more influence on high-level decision making. Combined with the large share of ‘general items’ (referring to meta-activities such as process management) performed by PM#5, this indicates a management-oriented position.

Thirdly, when we look at the persons PM#6, PM#3, PM#4 and PM#2, we see a fairly regular pattern in the distribution of time spent on the activities ‘theme definition’, ‘concept definition’, ‘requirements elaboration’, ‘standard activities’ and ‘development sprint preparation’. This seems to imply that the position of these persons is fairly similar.

Based on the observations made above, we can identify three different roles. The first role is the *senior product manager*, personified by PM#5. This role is mainly management-oriented, reflected by the low amount of time spent on requirements elaboration. Instead, a large share of the senior PM’s time is spent on high-level tasks, i.e. on the concept- or theme-level. The remaining time is for a fair amount spent on ‘general tasks’, ‘supplier management’ and other management related activities. The ‘senior product manager’ generally has the biggest influence on issues related to high-level decisions.

The second role that we identified was the *general product manager* role. In this particular team, this role was personified by four people with similar task-profiles, namely PM#2, PM#3, PM#4 and PM#6. Each of these persons spent a similar, large amount of time on both requirements elaboration and development sprint elaboration. The ‘product manager’ role is responsible for the lion’s share of low-level activities related to requirements and concepts. Furthermore, due to the close relation between ‘product managers’ and the development teams, most of the activities related to the development sprint can be attributed to the ‘product manager’.

The final identified role is that of the *requirements engineer*, which mainly focuses on low-level work. Almost one half of its time is spent on requirements elaboration. The other half of the time is chiefly divided between ‘development sprint preparation’, ‘concept-level activities’ and ‘general activities’. The requirements engineer is not an active participant during the ‘standard activities’
such as sprint review and planning meetings. In this particular team, the role of ‘requirements engineer’ was personified by PM#1.

4.3. Illustration: maintenance planning

To illustrate the specific workings of themes, concepts and requirements within the PMSB, 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 14 of the analysis. The entire SPM lifecycle of the theme lasts 7 months.

Although the theme elaboration has not been explicitly documented in the PMSBs, due to the fact that a theme-section was not yet available, 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 the ‘vision, scope and requirements’ document. 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 14, right before the introduction of the new PMSB structure. 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’. 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 the ‘vision, scope and requirements’ document. 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 14, right before the introduction of the new PMSB structure. 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 PMSB of that month, it is clear from the task-descriptions that they are related to theme-level requirements. During the next 5 months, the tasks related to the theme should shift from theme-level towards requirements-level. However, the PMSB 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 PMSBs for the first time in month 15 of our analysis, at the same time as the introduction of the new PMSB structure, and for the last time in month 20. 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.

4.4. 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. As an example of this, we describe the lifecycle of a new product idea, coined within the company in the fifth month of the 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 vision. After this vision was created, it was discussed and revised. It then had to be reviewed by the CIO. However, as the priority of this task was not high, it remained on the PMSB for several months. Only in month 11 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.
5. 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.

- **Alternate sprint cycles for SPM and development:** One of the main lessons learned has been the importance of the alternating sprints. As discussed in Section 2.5, 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.

- **Complex requirements 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.

- **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 15-min 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.

- **Backlog administration requires discipline:** We have observed that strict documentation of all tasks on the PMSB is still difficult to achieve. Although the PMSB 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 PMSB and discuss requirements before they have been implemented, reuse 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.

The final three lessons are similar to key aspects of the original SCRUM development approach. Our research has shown that they also apply for agile software product management. The first two lessons apply specifically to agile SPM, and we consider them essential to a successful implementation of agile SPM.

6. Conclusions and outlook

This paper demonstrates an attempts to apply agile principles to the software product management process, based on the proven structures of a well-adopted agile development method in an established software company. 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 Planon 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. Part of this consists of further analysis of the tasks that are relevant to an agile SPM process. Besides this, more insight should be gained regarding the suitability of development methods for the application of agile SPM. 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


