Case Study Report: Agile Product Management at Planon

K. Vlaanderen
S. Brinkkemper
T. Cheng
S. Jansen

Technical Report UU-CS-2009-005
March 2009

Department of Information and Computing Sciences
Utrecht University, Utrecht, The Netherlands
www.cs.uu.nl

ISSN: 0924-3275

Department of Information and Computing Sciences
Utrecht University
P.O. Box 80.089
3508 TB Utrecht
The Netherlands

Abstract

With the advent of agile development methods such as SCRUM, the unpredictable and dynamic process of software development has for the first time become manageable in an efficient manner. It is no surprise that these developments have resulted in an increased interest in the expansion of agile principles to other domains. One of these domains is product management, which involves mainly the elicitation, selection and prioritization of requirements as input for the development process. Up until now, sparse literature has been published concerning attempts to implement an agile product management process. In this work, we present the unique case of a company that successfully implemented an agile product management process. In this technical report we describe the relevant concepts, the product management and product development processes, and the lessons learned from implementing agile product management.

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 been developed (Abrahamsson et al., 2003). Examples of such methods are DSDM (Stapleton, 1999), Extreme Programming (Beck, 1999) and Feature Driven Development (Palmer & Felsing, 2002). 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[1].

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 (Schwaber, 1995) into practice range from small companies as described by Dingsoyr et al. (2006) to large multinationals (Fitzgerald et al., 2006). Research has shown that the use of SCRUM within a company can lead to significant benefits (Mann & Mauer, 2005), and that its use is not limited to local projects (Danait, 2005).

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 (Weerd et al., 2006; Gorchel, 2000). The topic of product management 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 (Carlshamre & Regnell, 2000), requirements interdependencies (Carlshamre et al., 2001) and evolutionary and iterative release planning (Greer & Ruhe, 2004). Another related field, requirements prioritization, has seen several publications in recent years, including work on requirements prioritizing for product software (Berander, 2007) and distributed prioritization (Regnell et al., 2001).

Due to the complexity of software products, with a large amount of shareholders, 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. (2006) in the form of a reference framework for SPM. This 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 stakeholders and their relations.

Currently, little work exists regarding agile SPM. A case study describing agile requirements engineering is described by Pichler et al. (2006). This research aims to further fill this gap by proposing an agile product management method based on the successful SCRUM development method. Furthermore, we present a case study performed at a product software company located in the Netherlands that has worked with agile SPM method for nearly two years. By showing their experiences, we are able to provide a set of useful lessons learned that aid in the implementation of SCRUM-inspired SPM alongside agile software development at other companies.

The remainder of this text is structured as follows. Section 2 describes the case study approach, including the research question and methods, and a description of the validity threats. Section 3

[1] http://agilemanifesto.org/

describes the conceptual model used in this study, with a definition of all relevant terms regarding SCRUM and agile product management. Section 4 contains a description of the application of SCRUM at the Dutch software vendor Planon, followed by a description of their implementation of agile product management in section 5. To conclude, section 7 contains a description of the lessons learned by Planon regarding agile product management and sections 8 and 9 summarize the study.

2 Case Study Research Approach

The main research question to this research is *How should SPM 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 facts have been collected to answer the research questions (Yin, 2003). The means through which we gathered these facts were:

- **Interviews** – The main research questions have been answered in part during the interviews with product stakeholders.
- **Document study** – The case company provided us with guideline documents such as the Volere Requirements Specification Template (Robertson & Robertson, 1998), the Product Backlog and Sprint Backlogs. These documents were studied and added to our case study database.
- **Direct observations** – Direct observations were made during our presence at the case company. These direct observations were later affirmed during the interviews.

A final analysis was performed on the obtained SPM sprint backlogs to obtain further insight in the practical consequences of agile product management.

2.1 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 (Yin, 2003; Jansen & Brinkkemper, 2008). 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. However, other similar sized software vendors have to continuously create new requirements as well, many of which have implemented SCRUM. We strongly believe that the practices described in this paper can be a successful way to manage teams of product managers for similar sized software vendors. 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 (Jansen & Brinkkemper, 2008).

2.2 Case Study Company: Planon

The main contribution of this work lies in the description of a black sheep 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, has, as one of the first known companies, attempted to implement an agile product management process based on the agile principle (and the SCRUM development method specifically).

Planon International is an international software vendor that produces Facility Management and Real Estate management software for organisations. Planon, founded in 1984, currently has a customer base of over 1300, which is supported by more than 325 employees. Planon's products are marketed through six Planon 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 25 million in 2007.

Facility management is the discipline of ensuring functionality of the built environment by integrating people, place, process and technology. The complex process of facility management encompasses activities such as long-range and annual facility planning, facility financial forecasting, real estate acquisition and/or disposal, work specifications, installation and space management, maintenance and operations management, and telecommunications integration, security, and administrative services. Planon develops client-server software (two- and three-tier architectures) with which it attempts to support the processes of facility management. Finally, Planon has implemented a set of components that can be used to communicate with other applications through XML interfaces.

Planon observes the following principles for their products. To begin with, Planon uses the principle that common data can be used among different processes. The common data is therefore only entered once into the data model that is central to all Planon software. Secondly, because so many of the facility management processes are affected by Planon software products, Planon has developed its own implementation framework. With this framework a specific implementation path can be designed for a customer. Finally, Planon trains the application managers at customers on a regular basis, as to provide them with more competence with the Planon products.

3 Definitions and Conceptual Model

The SCRUM development method was proposed in 1995 by Ken Schwaber (Schwaber, 1995), 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 (1996) and Booch (1995), and adhers 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 projects in a very flexible way. The only two parts that are fully defined during a project are the first and last phase ('planning' and 'closure'). In between, the final product is developed in a series of nonlinear, flexible 'black boxes' called 'sprints'. During these sprints, the final product is being evolved, subject to a constantly changing environment. This environment, which includes factors such as competition, time and financial pressure, maintains its influence on the project until the closure phase.

In figure 1, the entire SCRUM development process can be seen. In all cases, this process start with the environment. Based on the vision of the company, which includes ideas about ROI (Return on Investment), releases and milestones, a 'product backlog' (PB) emerges (the product backlog will be discussed later on). Once the PB is filled with requirements for the first time, the sprint cycle starts.

The figure shows the sprint as a central, recurring part of the method with a fixed duration of approximately one or two months. This pattern is also called the 'heartbeat' of SCRUM. Each cycle starts with a planning meeting in which decisions are being made on the PB items that will be handled during the current sprint. Once this is done, the specific PB items are assigned to the teams. It should be noted that these meetings, being the 'bridge' between two sprints, form the only possibility during the project for introducing changed or new requirements. Once a sprint has begun, it becomes a 'black box' until the sprint ends.

The product management (PM) planning meeting is followed by the planning meetings of all

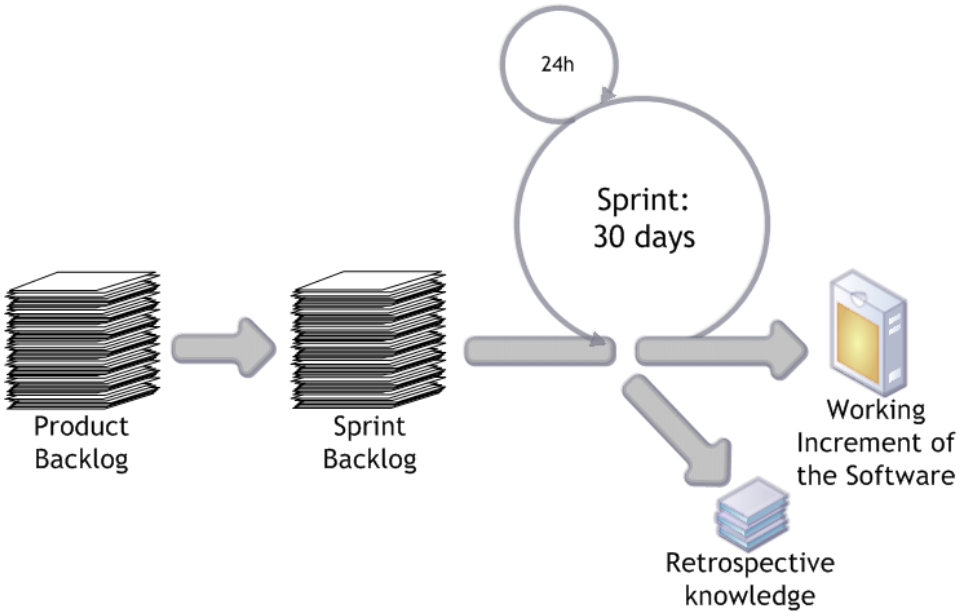

Figure 1: SCRUM development process

teams. During these sessions, knowledge about the project is being shared. Besides this, these sessions are used for the requirements-breakdown and the assignment of specific tasks to the different teams. Details about this breakdown and assignment are written down in the 'sprint backlog' (SB), which will be maintained during the sprint so that it reflects the current situation (more information about the SB will be provided later).

One can see that within each sprint, there is a daily recurring cycle. This cycle consists of two parts; the daily SCRUM meeting and actual development. During the daily SCRUM, the teams group together for 15 minutes to discuss the previous and current day. To guide the conversation, there are three questions that every member has to answer:

1. How did it go yesterday?
2. What will you do today?
3. Have you encountered problems?

Potential problems and improvements are identified and written down in a 'retrospective document'. Also, some small improvements to be made are placed on the sprint backlog.

All software that has been developed during a sprint has to be checked to see if it meets the criteria and if it is ready to be presented as a demo. Implemented requirements that do not meet one or more of the criteria are taken to the next sprint. The four criteria are the following:

1. All tests have been passed successfully
2. No urgent bugs exist
3. Documentation is made
4. Software is working

If the software does meet the criteria, it is taken to the activity "Deliver working software". At the end of every sprint, the sprint is evaluated by the team. The teams discuss what went well, what went wrong and what improvements can be made in the next sprint. These improvement

items are written down and a selection of one to three items is made. These items are then placed on the next sprint backlog.

When the sprint has ended and the software meets the criteria, an internal release is delivered. Also a demo is made to present the working software. Each team delivers working software, consisting of the software created in earlier sprints in addition to the requirements that have been implemented in the current sprint. The team presents the demo of the working software in the sprint review, which every stakeholder (employees, customers, product management) can attend. New requirements can be derived from this review, which are then inserted into the product backlog.

When the demo has been given and the sprint has been reviewed, an internal release of the working software is delivered. Each half year, internal releases are combined into an external release, which is then released to the customers. During this time, the sprint cycle is repeated until the entire product backlog has been implemented, or until the project terminates due to other reasons. These reasons include the absence of added business value or a lack of market demands.

3.1 Product & Sprint Backlog

One of the central documents in the SCRUM method is the product backlog. The PB contains a prioritized list of all items relevant to a specific product. This list can consist of bugs, defects, customer requested enhancements, competitive product functionality, competitive edge functionality and technology upgrades (Schwaber, 1995). An example of such a product backlog is shown in figures 2(a) and 2(b).

Each team that participates in the project maintains its own sprint backlog. On the SB are placed all the requirements that are assigned to the team. As stated before, every requirement is broken down into several tasks, which are then assigned to specific team-members (task-assignment will be detailed later on).

The sprint backlog contains, besides the requirements-breakdown and task-assignment, an estimation of work to be done per task and per employee. This makes it possible to keep track of the amount of work that has been done, thus allowing to see whether a team or a team-member is on track. An example sprint backlog can be seen in figure 3.

3.2 Effort Estimation

For the estimation of project, SCRUM does not offer any specific measurement system. The flexibility of the methods has as an effect that estimations can be rather complex and volatile. It is recognized however, that SCRUM projects tend to follow a certain pattern regarding acceleration and velocity. Initial velocity and acceleration are low as infrastructure is built or modified. As base functionality is put into objects, acceleration increases. Over time, the acceleration decreases and velocity remains sustainably high (Schwaber, 1995).

3.3 Task Assignment

SCRUM moves away from the idea of 'command and control'. Instead, it relies heavily on team empowerment. Effort estimation is one aspect of this empowerment, and it is supplemented by the way tasks are assigned to teams. In SCRUM, teams sign up for those tasks of which they think they will be able to complete them during the upcoming sprint. By using this set-up, more realistic estimations are produced.

Besides this, certain social aspects come into play. By letting the teams sign up instead of forcing tasks top-down, a higher feeling of commitment is created. Besides this, a sufficient (or even higher) velocity is enforced through social control.

ID #	Functionality	Remark(s)	Module	Release Theme
	Priority: 999			
	PM			
6866	The system shall be able to add or delete properties from an maintenan(	Requirement 9314...	PM	PM
	2008.11			
	Priority: 999			
	PM			
6864	The system shall be able to generate a new maintenance plan for a sel(	Requirement 9313...	PM	PM
	Report			
6830	Refactoring sub reports --> back end	2008-09: 0 CP (only test)	Report	Other
6831	Refactoring GUI --> front end	2008-09: 3 CP	Report	Other
6834	Refactor mail merge feature	2008-09: 1 CP	Report	Other
	Tech			
6961	Search for new Java profiler	Mail from Henk L on 28-8:..	Tech	Other
6400	Deploy the option in the installation process to choose for starting via W	DS issue !! (5 days)	Tech	Other
	Upgrade			
6890	Upgrade: Do not synchoronize TSI layout		Upgrade	Upgrade
	Webclient			
6897	Action in selection in Web	Reuse swing code (DIS 15	Webclien	PM

(a) Part one

PCP (remain)	Priority	isines: value	Sprint #	Team	State graph	State text	% done	PCP (real)	PCP (estimated)	TCP (estimated)	TCP (remain)
2 days	**999**						**0%**		**2 days**	**4 days**	**4 days**
2 days	**999**						**0%**		**2 days**	**4 days**	**4 days**
2 days	999	H		? Omega		1 red	0%	0,1 days	2 days	4 days	4 days
393 days							**47%**		**760 days**	**1.474 days**	**779 days**
27 days	**999**						**0%**		**27 days**	**52 days**	**52 days**
5 days	**999**						**0%**		**5 days**	**10 days**	**10 days**
5 days	999	H		? Omega		1 red	0%	0,1 days	5 days	10 days	10 days
13 days	**999**						**0%**		**13 days**	**26 days**	**26 days**
3 days	999	L	2008-09	Beta		1 red	0%	0,1 days	3 days	6 days	6 days
5 days	999	H	2008-09	Beta		1 red	0%	0,1 days	5 days	10 days	10 days
5 days	999	L	2008-09	Beta		1 red	0%	0,1 days	5 days	10 days	10 days
2 days	**999**						**0%**		**2 days**	**2 days**	**2 days**
2 days	999	H	2008-09	Galaxy		1 red	0%	2 days	2 days	2 days	2 days
0 days	999	H		? DS		1 red	0%	0 days	0 days	0 days	0 days
2 days	**999**						**0%**		**2 days**	**4 days**	**4 days**
2 days	999	H	2008-09	Alfa		1 red	0%	2 days	2 days	4 days	4 days
5 days	**999**						**0%**		**5 days**	**10 days**	**10 days**
5 days	999	H	2008-09	Galaxy		1 red	0%	0,1 days	5 days	10 days	10 days

(b) Part two

Figure 2: Extract from a sample product backlog

Beta: Sprint 2008-08			Day	1	2	3	4
Sprint Task Description (I II III IV X H1 H2 Task)	**Status**	**Resp**	**Orig Est Hrs**	**4-aug**	**5-aug**	**6-aug**	**7-aug**
			total remaining by day	578	560	528	512
Specification	Cancelled	[illegible]	2	2	2	2	2
JUnit review (FC's and VR)	Cancelled	[illegible]	2	2	2	2	2
Various					0	0	0
Loadfile	Completed	[illegible]	16	16	14	12	12
External translation job	Completed	[illegible]	12	12	8	8	0
Robotmaintenance	Completed	[illegible]	6	6	6	0	0
Robotmaintenance	Completed	[illegible]	4	4	4	4	4
Documentation team Alpha	Completed	[illegible]	22	22	22	20	20

Figure 3: Extract of a development sprint backlog

3.4 Performance Indicators and Controls

Along with the introduction of agile development methods such as SCRUM, it has become more and more important for project managers to adapt to a rapidly changing environment. New requirements are added on a near daily basis, and chances are high that a large share of the requirements changes during the lifetime of the project. In order to be able to cope with these dynamics, it is no longer sufficient to think of software projects as being linear and plannable.

Augustine et al. (2005) state that modern project management, in order to be adaptive and to be able to communicate good practice, should adhere to a certain set of practices:

- The ability to manage and adapt to change;
- A view of organizations as fluid, adaptive systems composed of intelligent people;
- Recognition of the limits of external control in establishing order; and
- An overall humanistic problem-solving approach that:
  - Considers all members to be skilled and valuable stakeholders in team management;
  - Relies on the collective ability of autonomous teams as the basic problem-solving mechanism; and
  - Minimizes up-front planning, stressing instead adaptability to changing conditions.

The unpredictable and complex nature of software projects in general and SCRUM projects specifically requires a set of controls that allows the management team, in cooperation with the development management, to adjust the project when the environment dictates so.

The main control in SCRUM projects is risk. Continuous risk-assessment is the main tool to prevent projects from falling into chaos. Risk-assessment influences all other controls of a project, including releases, changes, problems and solutions. Ultimately, is also has its effects on the aforementioned backlogs, which form another main type of control.

Besides this, the two earlier mentioned concepts 'velocity' and 'acceleration' can be used to assess and control a project. Velocity indicates the speed with which requirements or backlog items are implemented. Acceleration describes the evolution of this speed. For an efficient project-flow, velocity should be sustained at a high level throughout the project.

These two concepts can be combined into a new control, called the 'burn-down'. This graph relates the dimension of time to the amount of backlog-items to be implemented. By updating this graph on a realtime basis, a view is obtained that demonstrates the flow of the project. This helps both the product management team as well as the software development teams to assess the current situation and to adjust in order to improve efficiency when needed.

3.5 Programming Infrastructure

Another requirement for a successfull implementation of SCRUM is the availability of a high quality programming infrastructure. Such a structure should contain at least aspects like continuous buils, nightly builds, test runs and reporting.

4 SCRUM at Planon

For the implementation of its integrated workplace management solution, Planon switched to the SCRUM development method in 2004. Before this time, work was being done according to the Prince2 method. However, management recognized that by working in this manner, several issues arose. Firstly, release cycles could take up to one or one and a half year. This was combined with the fact that release end-dates were difficult to predict. Another important issue was the fact that during a project, many changes were requested. The Prince2 method did not offer sufficient

support for this, resulting in a lot of calculations that caused a large share of the product managements time to be put into these tasks instead of in product value. These downfalls motivated the switch to the SCRUM. This has resulted in several improvements. Firstly, changes are now implementable against far lower costs. Also, software is developed in one-month sprint, resulting in two release per year, with marketing-versions delivered in between.

During the period that Planon has worked with SCRUM, it has been adapted to the company's own needs and wishes. These adaptions and customizations originated mainly from a set of lessons learned. The rest of this section describes the main lessons learned by Planon and the changes that were made since the adoption of SCRUM. The textual representation is illustrated by a process-deliverable diagram (PDD), which is shown in figure 4. The accompanying activity table is shown in table 3 in appendix A. Its concept table is shown in table 5 in appendix B.

4.1 Sprint Duration

In essence, the SCRUM method does not prescribe a specific duration for sprints. In practice, such cycles can take anywhere from a few weeks to a few months. Initially, the sprint duration at Planon was two months. However, with a sprint of that length, the process became too inflexible. For this reason, the length of a sprint has been reduced to one month. This amount of time provides a workable balance between flexibility and the idea of a 'black box' in which the developers can work efficiently without being distracted by new requirements.

4.2 Product & Sprint Backlog

All themes, concepts and requirements that are placed in the 'vision, scope & requirements-document' are also converted to the product backlog. Once the high-level concepts have been broken into several smaller requirements, these are appended to the product backlog. This creates a list of items that starts with the high-level themes at the bottom and the low-level requirements at the top. During the lifecycle of the project, items on the PB are continuously detailed further, resembling a refinery process.

From this point on, teams are able to select them in order to convert them into working software. As stated above however, in most cases further refinement of the requirements is needed. For the largest part, this is done by product management. An exception to this are the 'design items' introduced by Planon, referring to those backlog items that are not fully defined yet and need some additional 'brainstorming'. Development teams can select these design items, which allows them to spend a certain amount of time thinking about that specific problem.

Every time that a requirement has been changed, decomposed, added or completed, the product backlog is updated. This ensures that a complete list of requirements is available at all times, especially at the time of the sprint planning meetings.

Since not all requirements are of equal importance, the product backlog items have to be prioritized in a certain manner. Bug fixes and certain feature requests might be placed lower on the list than other requirements, depending on factors such as business value, customer importance and return on investment. As a tool to support this prioritization, Planon has developed a valuation document. The document consists of priority-data gathered from customers based in several different countries. Each country is assigned a weighing-factor, which allows the calculation of a priority-score for each item on the list.

Although the valuation document is not intended to be used directly as input for the prioritization of the product backlog, it can function as input for further discussion. Based on such discussion, combined with the 'common-sense' of the 'product owner', a prioritized product backlog is created.

Each team that participates in the project maintains its own sprint backlog. On the SB are placed all the requirements that are assigned to a specific team. As stated before, every requirement is broken down into several tasks, which are then assigned to specific team-members.

The sprint backlog contains, besides the requirements-breakdown and task-assignment, an estimation of work to be done per task and per employee, expressed in 'complexity points' (CP;

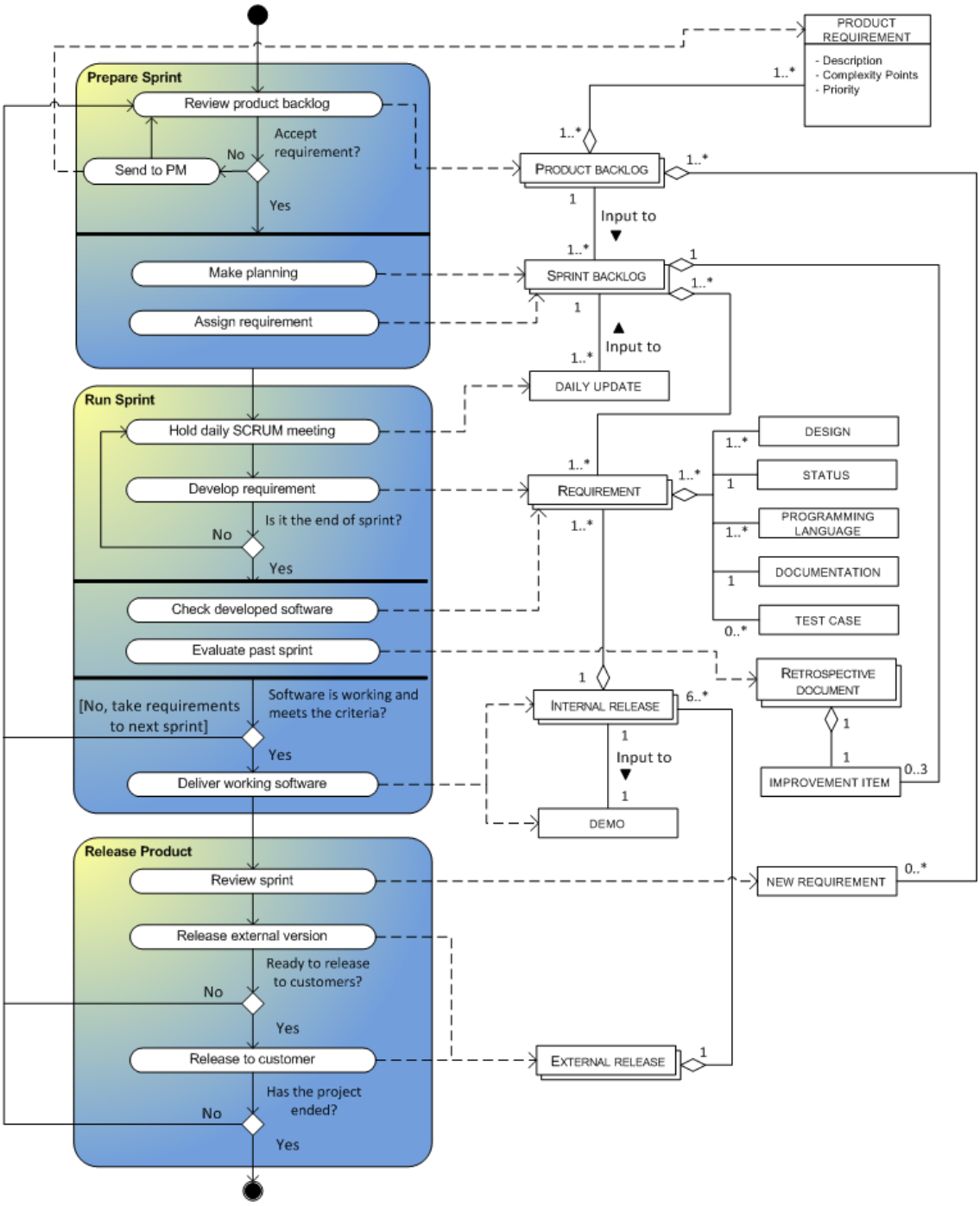

Figure 4: Process-deliverable diagram for the development process

complexity points will be defined later). This makes it possible to keep track of the amount of work that has been done, thus allowing to see whether a team or a team-member is on track.

4.3 Effort Estimation

In order to be able to make a sound planning, something should be known about the effort required for the implementation of a certain concept or requirement. Due to several reasons, one cannot suffice by simply taking the amount of hours required as a planning unit. Availability of testers, complexity of a task and skill of the teams are some of the factors that make it difficult to define an adequate unit.

As a solution to this, Planon currently uses 'complexity points' (CP) to indicate the estimated effort required. For each stage during the definition of the project, an estimation is made on the amount of CP's that a certain theme, concept or requirement will 'take'. However, what exactly constitutes a CP is not defined clearly. As a rule of thumb, 1 CP can be defined as 2 or 3 FTE's (Full-Time Equivalent), including testing and overhead time. More important though, is that CP's function as indication of the capacity of the teams and are to be seen relative to each other.

The assignment of CP's to specific requirements happens when new requirements become available. Their descriptions are handed over to the development teams who will then try to make a realistic estimation of the amount of effort required. Once these CP's are assigned to the requirements, they will be used throughout the project to keep track of the amount of work done and the amount of work that is remaining.

4.4 Task Assigment

Task assignment at Planon has moved away from the original idea as proposed by SCRUM. Although teams are still allowed to select those tasks of which they think that they can implement them within one sprint's time, more restrictions are being placed on the assignment of teams to projects. It is Planon's experience that teams become more effective and efficient once they have been working on a specific topic for a certain amount of time. This means that in practice, every team is assigned only to a few projects. As a result, team-members become specialists on this topic which increases their ability to implement related features.

4.5 Product Quality Assurance

One of the key-elements to a successfull SCRUM implementation identified by Planon is product quality assurance. In order to ensure optimal quality of the produced software, this should be done in the correct way as soon as possible, preferably from the beginning. Important in this respect is an effective composition of the development teams regarding the ratio of programmers over testers, which is three to one in the case of Planon.

Another crucial aspect is test automation along with unit testing by the whole team. Since it is not possible to test all features completely, test effort has to be determined per backlog item. To this extend, Planon uses a classification based on the dimensions 'potential business damage', determined by the product owner, and 'probability of failure (software complexity)', determined by the software architect. This results in four possible classifications according to which test effort is prioritized. Figure 5 shows the resulting diagram.

4.6 Development Team Composition

As stated before, team composition is important with relation to quality assurance, but also in relation to efficiency and effectiveness. Therefor, the composition of development teams requires considerable attention, since an incorrect set-up can have a large impact. At Planon, this was noted after changing team-formation prior to the sprints, which led to a decrease in the velocity. Only after several Sprints in unchanged composition teams become productive, so changing formations should be avoided where possible.

Currently, development teams at Planon are multi-disciplined, consisting of functional designers, testers, (lead) programmers and documenters. Besides this, two other functions exist outside of the teams, namely the lead architect and development infrastructure support.

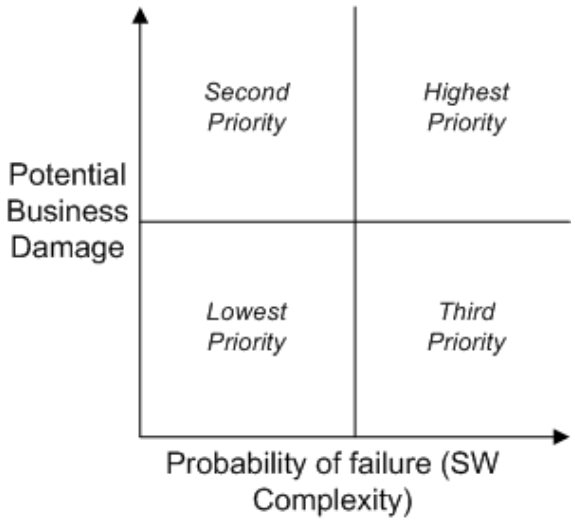

Figure 5: Determination of Testing Effort

5 Agile Product Management at Planon

Up until recently, little known work had been performed on the application of agile practices to every-day product management. To our knowledge, no publications have been made describing the successful, unsuccessful nor partial implementations of agile product management. As of 2007, Planon has made efforts to change this situation.

Planon has tried to apply SCRUM principles to the product management process. By making this adaptation, lean principles have been introduced to the requirements management process (Poppendieck & Poppendieck, 2003). Related to these principles are three main motivations that led to the implementation project. Synchronization of the product management heartbeat with the development heartbeat can be identified as the first reason. Secondly, the introduction of SCRUM principles inherently allows a greater flexibility in requirements management process. As a third reason, related to the previous, SCRUM principles allow a manageable implementation of continuous improvement.

5.1 The Process of Agile Product Management

This section describes the agile product management process as it is used at Planon, after which some aspects are described in more detail The textual representation is illustrated by a process-deliverable diagram, which is shown in figure 6. The accompanying activity table is shown in table 4 in appendix A. Its concept table is shown in table 6 in appendix B.

The inception of the process starts with the identification of a verified business problem. This can be a new theme, or a problem that fits within existing projects. The input for the themes can come from various sources, such as customers, employees or stakeholders. Once such a problem has been recognized, goals are being set for the upcoming project based on the vision of the company.

Stakeholders are involved in the decision making process and information is acquired from them. In order to be able to determine which themes and functionalities are going to be implemented, both the potential costs and opportunities are estimated. The cost is compared with the potential added business value, which is described in the release document.

Based on the business value comparison, involving stakeholders, identified opportunities and visions, profitable themes are selected. An estimation is made regarding the amount of time needed to realize the theme. After this estimation, themes are reviewed by product management. If they are approved, product management will further define the theme. If it is rejected, themes need to

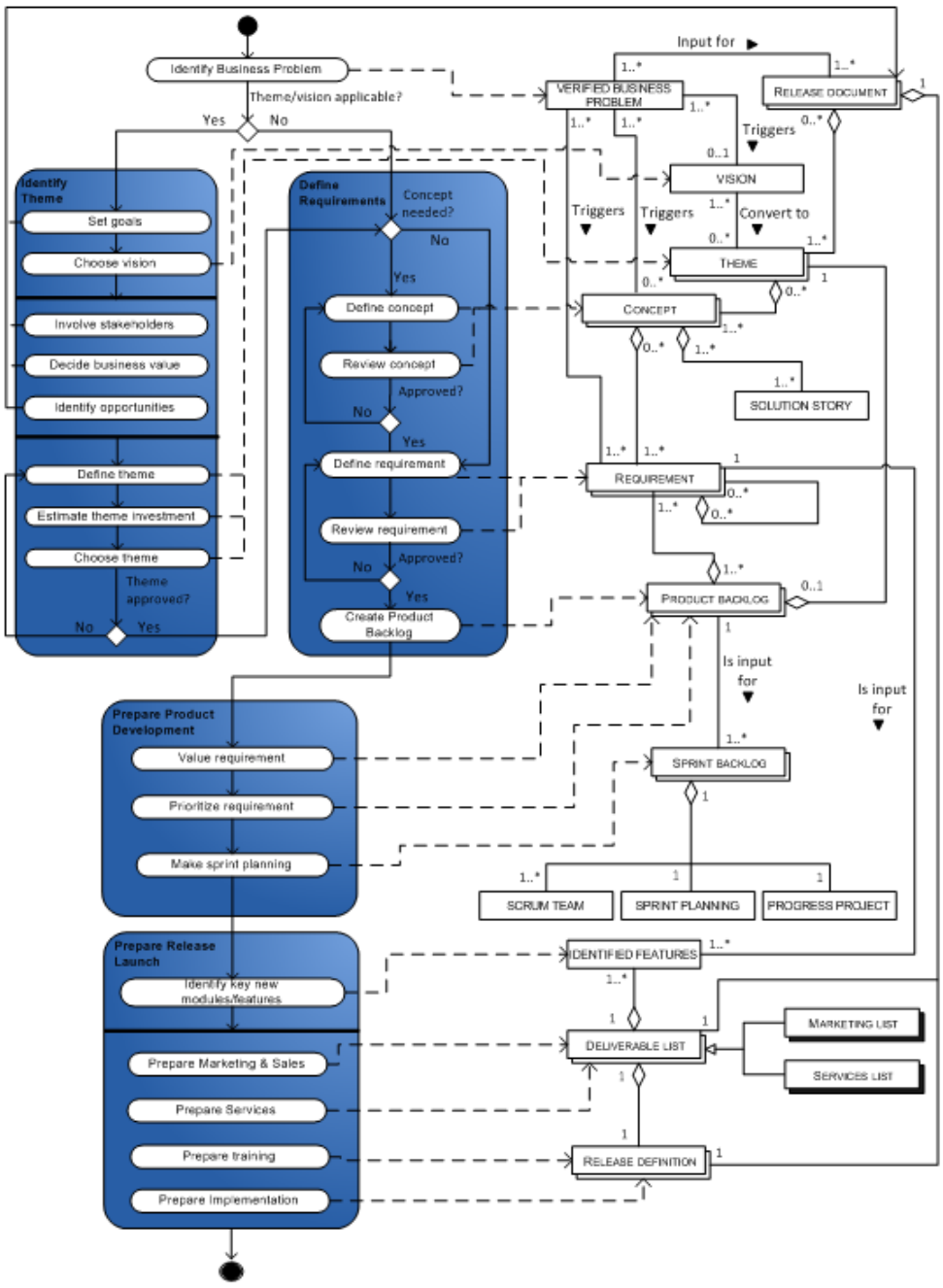

Figure 6: Process-deliverable diagram for the product management process

be redefined.

The chosen themes are, in most cases, broken down into a set of concepts. Every concept contains a set of solution stories. These stories are used for defining the requirements. The concepts are defined by product managers and software architects. When concepts are defined, they are reviewed by software architects and domain experts. If the concepts are approved, the

concept is broken down into a set of requirements. If a concept is not approved, it should be redefined or adjusted.

A requirement engineer and a SCRUM team are responsible for the definition of the requirements. Requirements can be broken down into smaller requirements to fit into a sprint. Requirements are also assigned a priority, after which they are put on the product backlog. The highest rated requirements are to be developed first. This priority rating is assigned by the product board and the sales department.

Requirements are reviewed by lead developers, architects, functional analysts and domain experts. If a requirement is approved, a potential value has to be estimated for it. If a requirement is rejected, it needs to be defined again and goes back to the activity 'Define requirement'. The reason for rejection can be: that it is unclear or that it is not described sufficiently.

When requirements are approved, the costs and business value are calculated. Each requirement is valued and prioritized through a value document. First the countries are prioritized and a weighing-factor is assigned. After the country prioritizing, the requested features are prioritized. The prioritization is put into the product backlog and is used for deciding about which features to develop and when they should 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 SCRUM teams. The requirements are put in the sprint backlog of the specific team and the general sprint backlog of product management. With the general sprint backlog, product management can keep track of the project.

After assigning requirements, the release needs to be prepared. New modules and features are first identified and prepared for the departments Marketing & Sales and Services. A deliverable list is made to prepare the departments. The list contains items about how and what needs to be sold effectively to the market and the customers. The logistics also needed to be prepared. Other items can be about advising customers, how to use the new features and how to implement the new features. Furthermore, Marketing & Sales, Services, customers and partners need to be trained.

5.2 Vision, Scope & Requirements

Before any project can start, it should to a certain extent be clear what is going to be built. Since SCRUM itself does not provide guidelines for the initial phase, Planon has extended this with the concepts 'vision', 'theme' and 'concept'. Definitions of the theme, concepts and requirements are maintained throughout the entire project in a document called the 'Vision, Scope & Requirements'-document.

A vision is basically the starting point for each project. It is an idea, brought up by an employee, a customer or any other stakeholder, and is in many cases only very vaguely defined. Once the idea reaches a product manager or product owner, he or she then converts it into a (set of) theme(s).

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 stakeholders that are involved. This is done based on the experience of the product manager regarding the probable scope of the new themes.

Since a theme is a very high-level concept, it has to be broken down into smaller pieces. This is done by converting a theme into a set of concepts. This is again performed by the product manager. Each concept is a high-level focal point within the theme. A concept definition contains a set of solution stories that can later be used to deduct specific requirements.

Requirements definition is performed in three steps, of which only the first one is performed by product management (PM). PM translates the concepts into a list of requirements without going into a lot of detail. Requirements definitions consist up to this point of a description, a rationale and a 'fit criterion'. The process describing further refinement of the requirements is described in the next section.

5.3 Product Management Sprint

The agile aspect of Planons product management approach lies mainly in the fact that, besides software development, the product management task is also performed according to thirty-day sprints. These sprints are not performed synchronously to the software development sprint, but are shifted two weeks back. This ensures that the software development backlog is always up-to-date and ready to be used once the software development sprint starts. Figure 7 illustrates this concept of asynchronous sprints.

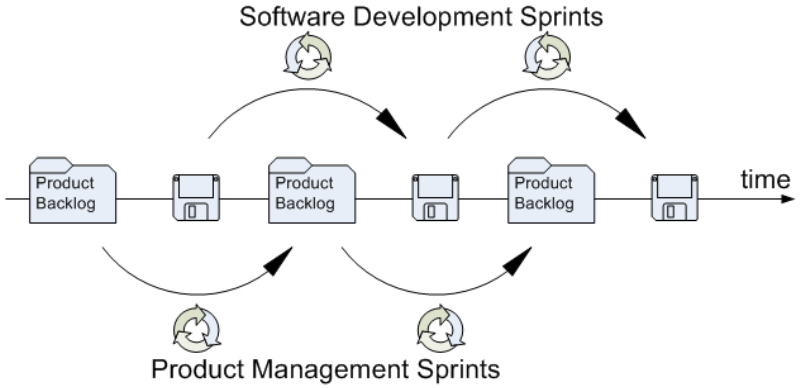

Figure 7: Alternating Sprints

Product management also performs its work within daily sprints. Similar to software development, new, completed or cancelled tasks are continuously kept up-to-date on the product management sprint backlog. Also, a burn-down chart is created continuously to allow monitoring of the progress of the sprint. Another similarity is that the sprint backlog is filled with items from the product management backlog at the beginning of each sprint. However, the items on this list are not seperated into the levels of 'vision', 'theme' and 'concept'.

5.4 Planons New Vision of SCRUM

The changes that Planon has made to SCRUM, especially in the light of agile product management, lead to a revision of the visualization of SCRUM as shown earlier (figure 1). The new diagram is shown in figure 8.

As depicted by the oval shape, a few aspects have been altered or added. Primarily, the default product backlog has been replaced with the new, more detailed PB containing Planon's 'theme', 'concepts' and 'requirements', which we earlier mentioned as the requirements refinery. Also, 'themes' have been added to the illustration, in accordance with the idea that leads to the definition of themes and concepts. Finally, 'bugs' have been introduced to depict an alternative way of generating product backlog items.

Both software development as well as product management follow similar processes. Both work according to a (product) backlog and a sprint backlog, and both perform their work in thirty-day sprints. The difference lies in the product that is created, as the output of product management is a prioritized product backlog, which is then used by the development teams to create a working software increment.

6 Product Management Backlog Analysis

To gain insight into the actual workings of the SPM adaptation of SCRUM, we cannot suffice with mere superficial observations. Instead, through a more profound analysis of the SPM documents we are able to gain a more detailed view of the results and implications of Planons adaptations.

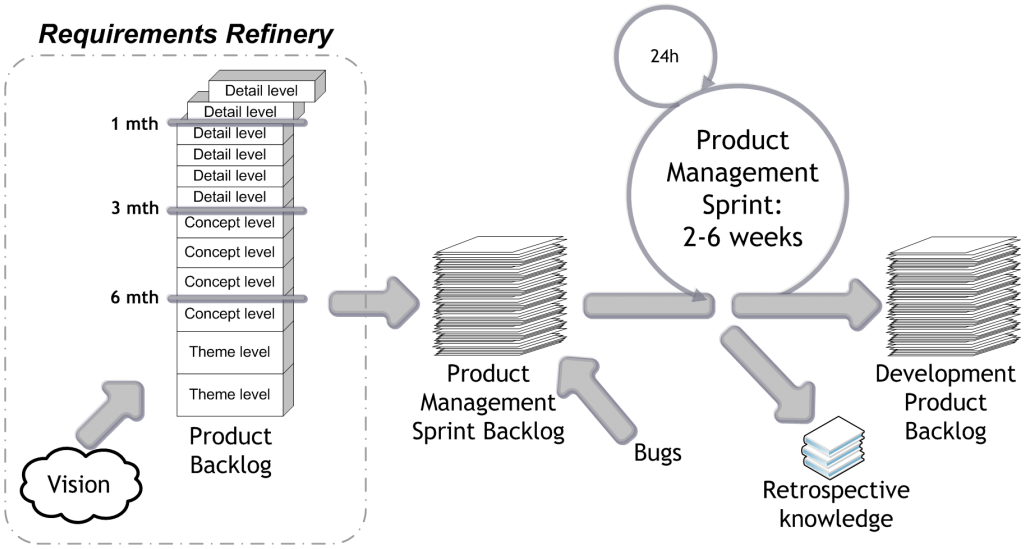

Figure 8: Planon's Vision of SCRUM

For this analysis we have compiled a data-set of twenty-one SPM sprint backlogs, describing an equal amount of months. The scope of the analysis ranges from March 2007, when SCRUM was introduced into the SPM process, until November 2008.

	Mean	**Lowest**	**Highest**
# of Tasks	75.4	43.0	121.0
# of Prod Man	5.6	5.0	6.0
# of Hours at start	537.9	324.0	750.0
# of Hours at end	197.1	102.0	320.0
Avg Hours per Task	7.7	3.7	12.9
Avg Tasks per Person	13.5	7.2	22.4
Avg Hours per Person	97.4	54.0	150.0
% of Hours Fulfilled	63.3%	47.6%	81.1%

Table 1: Descriptive statistics based on twenty-one product management backlogs

In our analysis, we have focused our attention both on the description of general statistics about the task structure, including task duration and workload per person, as well as on the discovery of interesting patterns. In table 1, we have shown some global figures about the data (the complete data-set can be found in appendix C, table 7). From left to right, the table first shows the amount of tasks that were placed on the SPM sprint backlog 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, data is shown displaying 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.

However, the real importance in this analysis lies in the observed evolution and correlation of certain components. During the two years of experience, the SPM backlog provides interesting information regarding the amount of tasks and their characteristics. Firstly, we are able to distinguish a set of recurring, standard activities. The second interesting point is the introduction of themes, concepts and requirements into the SPM backlog. To conclude, we will track two themes

through the entire SPM process.

6.1 Standard Tasks

In order to create certain structure and clarity in the SPM task list, it is valuable to have a set of standard tasks. These standard tasks can form a base structure of recurring activities, mostly with the same amount of hours allocated. These tasks can be used to create a form of rhythm, or a heartbeat, within the team(s). Analysis of the generic tasks can create valuable knowledge that is applicable to other organizations as well. Information about task size and type can be used by other CTOs to structure the SPM sprint backlogs. As fine-tuning of this structure can take considerable time, this can prove to be a considerable advantage.

At the case company, the list of standard activities has evolved from a disorganized list into a rather stable list of tasks during the period that is described in this paper. As described earlier, the SPM sprint backlog was at first mainly structured in a product-focused manner. This had as an effect that recurring tasks were spread across the backlog, resulting in a disorganized list which had to be recreated from scratch every month.

Standard Activities	**Development Sprint Preparations**
Prepare and Attend Product Board	Backlog Preparations
Sprint Review	Sprint Planning with Dev. Teams
Team Retro Meeting	Sprint Review with Dev. Teams
Team Allocation Overview	How-to-demo Stories
Problem and Change Management	Several Product Related Tasks
Release Plan	

Table 2: Standard Activities on the SPM Sprint Backlog

As of month five, a small list of recurring tasks related to the product board can be distinguished. However, this is comprised of not more than 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, at the time when the new SPM sprint backlog is introduced. At this moment, the list of standard activities 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 activities is comprised of six different tasks, quite similar to the tasks of the earlier months, totaling only 80 hours. This final list of standard activities is shown in table 2.

This 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 SPM sprint backlog 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 table 2.

6.2 Themes, Concepts & Requirements

As described earlier in section 5.2, the concepts theme and concept were introduced into SCRUM well after the method was adapted to the SPM process. Although the term concept can be found on the backlog as of month two, themes are introduced for the first time in month fifteen. Regarding the concepts, several interesting notes can be made. The backlogs 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. Further elaboration of product features is displayed under a products-list within the backlog.

The switch to the 'theme/concept/requirement' requirements lifecycle in month fifteen has some clear effects on the backlog. Most notable is the immediate structure and clarity that is created by this adaptation. By dividing the tasks related to the elaboration of requirements into

three sublists, a clearer overview of the workload is obtained. For every task it becomes instantly clear in what phase of elaboration the requirement currently is.

One consequence of this approach can be seen in the evolution of task size and amount. During the two years of experience with SPM backlogs, two distinct trends have developed. On the one hand, the amount of tasks on the sprint backlogs has increased approximately twofold, whereas the average size of the tasks has decreased with approximately 50%. Evidence on the backlogs suggests a relation with the introduction of themes and concepts on the backlog, as larger tasks such as 'describe requirements' are now split into smaller tasks, specific to the current phase.

6.3 Illustration: Maintenance Planning

To illustrate the specific workings of the themes, concepts and requirements within the SPM backlog, we will try to follow the maintenance planning theme 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 is concerned with functionality related to the maintenance of facilities, and was initially introduced on the product backlog in month fourteen of our analysis. The entire SPM lifecycle of the theme lasts seven months.

Although the theme elaboration has not been documented in the SPM sprint backlogs, the theme is elaborated into several concepts. These concepts are 'planned maintenance (PM)', 'planned preventative maintenance (PPM)' and 'maintenance management (MM)'. Besides these, there are several other concepts 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, we will mainly follow the concepts of planned maintenance and maintenance management.

The planned maintenance concept was introduced in month fourteen, right before the introduction of the new backlog structure. Although a theme-section was not yet available in the SPM backlog of that month, it is clear from the task-descriptions that these 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 backlog 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. Analysis of the product backlog reveals that the concept of planned maintenance was elaborated into 152 requirements, subdivided over the groups 'no value', 'contract', 'generic' and 'maintenance planning'.

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 this concept can be found on the sprint backlogs for the first time in month fifteen of our analysis, at the same time as the introduction of the new backlog 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.

6.4 Illustration: Planon Lite

As we have seen in the previous section, the introduction of themes, concepts and requirements on the SPM backlog 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 will make it through the entire process. In fact, the added detail allows for an increased visibility of theme life cycles. As an example of this, we will describe the lifecycle of a new product

idea 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 backlog 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.

7 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 important 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 depicted in figure 7, the software development and the SPM sprint are both performed continuously, but with a difference in starting date of approximately two weeks. This implies that each SPM sprint ends halfway the software development sprint, ensuring that the product backlog 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** – This contribution 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 product backlog and handled in time by the appropriate person.

- **Backlog administration requires discipline** – We have seen that strict documentation of all tasks on the sprint backlog is still difficult to achieve. Although the sprint backlog 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 backlog 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.

8 Discussion

Although we have aimed our efforts at providing reliable and objective results, we must recognize that some aspects of our research have room for improvement in the future. By making these aspects explicit, we are able to take a critical look at our work and to adapt our approach in consecutive projects.

A first critical mark concerns the amount of persons that we have interviewed. The amount of four employees does not allow for sufficient cross-checking, thereby introducing the possibility of reduced reliability of the data. Although the persons that we have interviewed held high positions within the product management process, this is no substitute for the internal validity principal.

A second point of remark can be made regarding the amount of time spent with each interviewee. Time limitations prohibited our ability to go into depth on every relevant aspect. By increasing the amount of time available for future sessions, the quality and amount of information obtained can be increased.

9 Conclusions and Outlook

Up until now, no attempts to create an agile SPM process had been described in literature. In this situation we have proposed such a method based on agile principles and the proven structures of a well-known 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.

From the situation described in this work, it has become clear that an agile development process implies an environment that is significantly different from those created by other, more linear and controlled methods. As a result, such a dynamic environment places its own, unique requirements on the processes that are related to it. The effect of this is an increased demand for agile SPM processes, as described in this work. The main contribution of this work has been the description of a 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.

Furthermore, the specific lessons that Planon has learned during its experience with agile SPM and SCRUM allow companies that wish to implement agile SPM to circumvent a set of potentially dangerous problems. The experiences of Planon have shown that, to ensure effective agile SPM, several factors should be taken into account. Factors such as team composition, effort estimation and product quality assurance become even more critical in such situations. For a successful implementation, these aspects require considerable effort, both in the preparation of a product as well as, mainly, during the lifecycle of the product itself. At this point in time, the maturity of the agile SPM process is still fairly low. Therefore, little is known about the exact requirements of such a process. Future research should be aimed at further formalization of these requirements. 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.

References

Abrahamsson, P., Warsta, J., Siponen, M. T., & Ronkainen, J. (2003). New directions on agile methods: A comparative analysis. *Proceedings of the International Conference on Software Engineering*, *May*, 244.

Augustine, S., Payne, B., Sencindiver, F., & Woodcock, S. (2005). Agile project management: Steering from the edges. *Communications of the ACM*, *48*(12), 85 - 89.

Beck, K.(1999). *Extreme programming explained: embrace change*. Boston, USA: Addison-Wesley Longman Publishing Co., Inc.

Berander, P.(2007). *Evolving prioritization for software product management*. Blekinge Institute of Technology.

Booch, G.(1995). *Object solutions: Managing the object-oriented project*. Addison-Wesley.

Carlshamre, P., & Regnell, B.(2000). Requirements lifecycle management and release planning in market-driven requirements engineering processes. In *Workshop on database and expert systems applications.*

Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., & Dag, J. N. och.(2001). An industrial survey of requirements interdependencies in software product release planning. In *Fifth ieee international symposium on requirements engineering.*

Danait, A.(2005). Agile offshore techniques - a case study. In *Agile conference.*

Dingsoyr, T., Hanssen, G. K., Dyba, T., Anker, G., & Nygaard, J. O.(2006). Lecture notes in computer science. In (p. 5-15). Springer Berlin / Heidelberg.

Fitzgerald, B., Hartnett, G., & Conboy, K.(2006). Customising agile methods to software practices at intel shannon. *European Journal of Information Systems*, *15*, 200-213.

Gorchel, L.(2000). *The product managers handbook: The complete product management resource (2nd edition)*. NTC Business Books.

Greer, D., & Ruhe, G.(2004). Software release planning: an evolutionary and iterative approach. *Information and Software Technology*, *46*(4), 243-253.

Jansen, S., & Brinkkemper, S.(2008). Information systems research methods, epistemology, and applications. In A. Cater-Steel & L. Al-Hakim (Eds.), (p. 120-139). Idea Group Inc.

Mann, C., & Mauer, F.(2005). A case study on the impact of scrum on overtime and customer satisfaction. In *Agile development conference.*

Palmer, S., & Felsing, J.(2002). *A practical guide to feature-driven development*. Prentice Hall.

Pichler, M., Rumetshofer, H., & Wahler, W.(2006). Agile requirements engineering for a social insurance for occupational risks organization: A case study. In *14th ieee international requirements engineering conference.*

Pittman, M.(1996). Lessons learned in managing object-oriented development. *IEEE Software*, *10*(1), 43-53.

Poppendieck, M., & Poppendieck, T.(2003). *Lean software development an agile toolkit*. Boston, MA, USA: Addison-Wesley Longman Publishing.

Regnell, B., Höst, M., Dag, J. N. och, Beremark, P., & Hjelm, T.(2001). An industrial case study on distributed prioritisation in market-driven requirements engineering for packaged software. *Requirements Engineering*, *6*(1), 51-62.

Robertson, J., & Robertson, S.(1998). *Volere requirements specification template edition 6.0* (Tech. Rep.). Atlantic Systems Guild.

Schwaber, K. (1995). Scrum development proces. In *Proceedings of the conference on object-oriented programing systems, languages, and applications workshop on business object design and implementation.*

Stapleton, J.(1999). Dsdm: Dynamic systems development method. In *Proceedings of technology of object-oriented languages and systems.*

Weerd, I. van de, Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L.(2006). Towards a reference framework for software product management. In *14th ieee international requirements engineering conference.*

Yin, R. K.(2003). *Case study research - design and methods.* SAGE Publications.

A Activity Tables

Table 3: Activity table Product Development process

Activity	Sub-activity	Description
Prepare Sprint	Review product backlog	An existing PRODUCT BACKLOG is reviewed by the team and product owner to see which PRODUCT REQUIREMENTS needs to be developed in the upcoming sprint. The PRODUCT BACKLOG contains PRODUCT REQUIREMENTS for a project. Each sprint has a number of PRODUCT REQUIREMENTS to be developed. The team decides if they accept or reject the REQUIREMENTS. Rejected REQUIREMENTS are sent back to the Product manager. Accepted REQUIREMENTS are planned and assigned for the upcoming sprint.
	Send to PM	REQUIREMENTS that are rejected will be sending back to the Product Manager. Further description or adjustments has to be made for the team in order to be clear and able to develop. When the REQUIREMENTS are further described or adjusted by the Product Manager, the team can review the PRODUCT REQUIREMENT in the PRODUCT BACKLOG.
	Make planning	A planning for developing REQUIREMENTS is made for each team with their SPRINT BACKLOG for the upcoming sprint.
	Assign requirement	The PRODUCT REQUIREMENTS are assigned to the SCRUM teams. The PRODUCT REQUIREMENTS are put in the SPRINT BACKLOGS of the teams. Each team has their own SPRINT BACKLOG.
Run Sprint	Hold daily SCRUM meeting	Every day is a SCRUM meeting where teams can come together to answer three questions: 1. How did it go yesterday? 2. What will you do today? 3. Have you encountered problems? Every team has their own SCRUM meeting. The SCRUM master gets an update and can remove certain impediments. When this is done the progress can be clearly seen and the SPRINT BACKLOG can be updated.

Continued on next page

Table 3 – continued from previous page

Activity	Sub-activity	Description
	Develop requirement	After the daily SCRUM meeting, the team does their own work and develops the assigned REQUIREMENTS. Each REQUIREMENT has one or multiple DESIGNS, one STATUS, programmed in one or more PROGRAMMING LANGUAGES and has one Document. Depending on the priority and importance of the REQUIREMENTS, developed REQUIREMENTS are chosen and tested. Therefore not every REQUIREMENT has a TEST CASE. But a REQUIREMENT can have multiple TEST CASES. If the sprint has not ended, it goes to the activity "Hold daily SCRUM meeting". When the sprint ends it goes to the activities "Check developed software" and "Evaluate past sprint".
	Check developed software	Developed software is checked if they meet the criteria and ready to be presented as a demo. The four criteria are: 1. Junit test has been passed successfully. 2. No urgent bugs. 3. Documentation is made. 4. Software is working. When one of the criteria is missing, the REQUIREMENTS are taken to the next sprint and goes to the activity "Review product backlog". If the software meets the criteria it goes to the activity "Deliver working software".
	Evaluate past sprint	An evaluation takes place at the end of every sprint. The teams discuss what went well and what went wrong and what improvements can be made for the next sprint. These IMPROVEMENT ITEMS are written down and a selection of one to three IMPROVEMENT ITEMS is made. They are taken onto the SPRINT BACKLOG.
	Deliver working software	When the sprint has ended and the software meets the criteria, an INTERNAL RELEASE is delivered. Also a DEMO is made to present the working software. Each team delivers working software. The working software is the REQUIREMENTS combined together to an INTERNAL RELEASE.

Continued on next page

Table 3 – continued from previous page

Activity	Sub-activity	Description
Release Product	Review sprint	The team presents the demo of the working software in the Sprint review. Every stakeholder (employees, customers, product management) can attend the sprint review. NEW REQUIREMENTS can be derived from the review. These NEW REQUIREMENTS are put into the PRODUCT BACKLOG.
	Release external version	When the DEMO is given and the sprint has been reviewed, The INTERNAL RELEASE is released. Each half year an EXTERNAL RELEASE of the software is released. When the half year hasn't passed, a new sprint starts and goes to "Review product backlog".
	Release customer	If the half year has passed and there have been six successful releases of the INTERNAL RELEASE, an EXTERNAL RELEASE of the software is released to customers. A new sprint starts if the project is still running and goes to "Review product backlog". Project can end because of several reasons; no business value, no market needs or software has been developed successfully and no more adjustments/extensions have to be made.

Table 4: Activity table Product Management process

Activity	Sub-activity	Description
Identify Business Problem		The inception of the process always starts with identifying a VERIFIED BUSINESS PROBLEM. These problems are the triggers for the process. Then the decision is made whether VISION and THEMES are applicable to solve the business problem. If applicable or needed it goes to "set goals" or else "Create Product Backlog".
Identify Theme	Set goals	Set goals for the upcoming project and based on the goals and the business problems, VISIONS are chosen.
	Choose vision	The input for the VISION can come from customers, employee or a stakeholder. Visions are chosen to solve the business problem.
	Involve stakeholders	Stakeholders are involved in the decision making process and information is acquired from them.
	Decide business value	VISIONS are depicted in a graph to see their business value. The cost is compared against their added value. The business value is described in the RELEASE DOCUMENT.
	Identify opportunities	Based on the business value, opportunities are identified.
	Define theme	The VISION is further described to a THEME. A VISION is defined to a THEME or a set of THEMES.

Continued on next page

Table 4 – continued from previous page

Activity	Sub-activity	Description
	Estimate theme investment	An estimation of the investments and how long it will take to realize the THEME is made. The costs and the business value are calculated.
	Choose theme	Based on the business value comparison, stakeholders of Product Management and theme investment, THEMES are chosen for the upcoming project. The chosen THEME is described further in the RELEASE DOCUMENT. When THEMES are chosen, the next decision is if CONCEPTS are needed or go straight to defining REQUIREMENTS.
Define Requirements	Define concept	A THEME is broken down to a set of CONCEPTS. These CONCEPTS contains SOLUTION STORIES. These stories are used for defining the REQUIREMENTS. CONCEPTS are defined by product managers and software architects.
	Review concept	When CONCEPTS are defined, they are reviewed by Software Architects and Domain Experts. If the CONCEPTS are approved, a set of REQUIREMENTS are defined in the next activity. If the CONCEPT is not approved, it should go back to the activity 'Define Concept'.
	Define requirement	Each CONCEPT is further described and broken down to a set of REQUIREMENTS. A team is responsible for defining REQUIREMENTS. A REQUIREMENT can be broken down to smaller REQUIREMENTS to fit in a sprint. The REQUIREMENTS are put into the PRODUCT BACKLOG.
	Review requirement	REQUIREMENTS are reviewed by Lead Developers, Architects, Functional analysts and Domain Experts. If a REQUIREMENT is approved it goes to the next activity 'Value requirement'. If a REQUIREMENT is rejected, because they are unclear or not sufficiently described, they need to be defined again and goes back to the activity 'Define requirement'.
	Create Product Backlog	A PRODUCT BACKLOG is made to put in the chosen THEMES, CONCEPTS and REQUIREMENTS for the upcoming project.
Prepare Product Development	Value requirement	The costs and the business value of a REQUIREMENT are determined. These REQUIREMENTS are valued and prioritized in the next activity.

Continued on next page

Table 4 – continued from previous page

Activity	Sub-activity	Description
	Prioritize requirement	A value document exists to value and prioritize the REQUIREMENTS. First the countries are prioritized and a weighing-factor is assigned. After the country prioritizing, the requested features are prioritized. REQUIREMENTS get a priority and are put into the PRODUCT BACKLOG. The highest rated REQUIREMENTS needs to be developed first.
	Make sprint planning	When REQUIREMENTS are clear and have enough detail they are assigned to SCRUM teams. The REQUIREMENTS are put in the specific sprint backlog of each team and the general sprint backlog of Product Management. With the general sprint backlog, Product Management can keep track of the project. The progress of a project can be tracked and monitored. There is a planning and an overview of the teams with their REQUIREMENTS. Each development team has their specific sprint backlog containing the progress, planning and the members of the team. The sprint backlog contains the following items: PROGRESS PROJECT, SPRINT PLANNING and SCRUM TEAM.
Prepare Release Launch	Identify key new modules/features	After assigning REQUIREMENTS, the release needs to be prepared. New modules and features are first identified and prepared for the departments: Marketing & Sales and Services.
	Prepare Marketing & Sales	A DELIVERABLE LIST (MARKETING LIST) is made to prepare the department Marketing & Sales. The list contains items about how and what needs to be sold effectively to the market and the customers. The logistics are also needed to be prepared.
	Prepare Services	For an effective implementation a DELIVERABLE LIST (SERVICES LIST) is made. Items about advising customers, how to use the new features and implementation of these new features are part of the list.
	Prepare Training	The departments Marketing & Sales and Services needs to be trained and is described in the IMPLEMENTATION LIST. This list can also be found in the RELEASE DOCUMENT. Besides a training description for the departments, also the customers and partners of the company need a training description.
	Prepare Implementation	The necessary items to effectively implement the new modules/features are described in the RELEASE DEFINITION and are also a part of the DELIVERABLE LIST and RELEASE DOCUMENT.

B Concept Tables

Table 5: Concept table Product Development process

Concept	Description
PRODUCT BACKLOG	Product backlog contains product requirements for the upcoming project. The product backlog acts as the intermediary of product development and product management. Product requirements are chosen and assigned to SCRUM teams.
PRODUCT REQUIREMENT	Planned requirements for the project and is described in the product backlog. Each requirement has a priority and a number of complexity points. The highest priorities are to be developed first. The complexity points indicate how much developer time is needed to create it.
SPRINT BACKLOG	Each team has their own sprint backlog. This backlog contains the assigned requirements, improvement items from previous sprint and the progress of the sprint can be monitored. It gets updated every day.
DAILY UPDATE	After each sprint meeting, a daily progress comes out.
REQUIREMENT	These are the requirements that are developed during sprints. Each requirement has a design, a certain status, programmed in a language and is documented. Requirements with the highest priority and most complexity points, has a test priority. They need to be tested before delivering a service pack.
DESIGN	A functional and technical design is made before developing the requirements.
STATUS	Assigned requirements for the sprint can have one of the following statuses: Not Started, In Progress, Completed, and Cancelled.
PROGRAMMING LANGUAGE	The used programming language depends on the known programming language by the teams and the fit for a certain requirement/module.
DOCUMENTATION	Documentation of requirements and working software is made during the sprint.
TEST CASE	Requirements can have different test cases, depending on the importance (potential business damage) and complexity of the requirements.
RETROSPECTIVE DOCUMENT	After each sprint, the teams reviews and evaluates their sprint. Things to improve are put in a retrospective document. This document contains improvement items for the next sprint.
IMPROVEMENT ITEM	These are the items a team has acknowledged to be improved in the next sprint. A maximum of three improvement items are chosen and put in the next sprint backlog.
INTERNAL RELEASE	When a sprint has ended and tests are successful, a piece of working software as an output from a sprint is delivered. The developed requirements are combined to an internal release.
DEMO	A demo is made from the internal release to show the developed functionalities.
NEW REQUIREMENT	During the sprint review when the demo is given, new requirements can come up from the attendees. These are put into the product backlog and further specified.

Continued on next page

Table 5 – continued from previous page

Concept	Description
EXTERNAL RELEASE	Each half year, the internal releases are combined to an external release.

Table 6: Concept table Product Management process

Concept	Description
VERIFIED BUSINESS PROBLEM	Identified business problems are business cases which is the inception of starting the process of Product Management.
RELEASE DOCUMENT	This document is the formal documentation of a software development project. The document contains the following items: the business case, product description, vision, theme, concepts, requirements, deliverable list and a release definition. These items are described in detail and a planning can also be found in this document.
VISION	Starting point of each project. Vision is an idea of a stakeholder, customer or employee. A vision can be divided into a set of themes.
THEME	A formal elaboration of a vision. Each theme can be broken down to a set of concepts.
CONCEPT	A concept contains a set of story solutions to realize a theme. Each concept can be broken down to a set of requirements.
SOLUTION STORY	The solution stories are used for deduction into requirements.
REQUIREMENT	Requirements are detailed functionalities of a software product. The requirements can also be broken down to several sub-requirements. The developed requirements are input for identifying new features for the market.
PRODUCT BACKLOG	Product backlog contains a part of the release document. It contains the same items as the release document except for deliverable list and release definition. The product backlog acts as the intermediary of product development and product management. Both of them can work in the product backlog. This Product Backlog is reviewed by development teams. They decide if the assigned requirements are accepted or rejected.
SPRINT BACKLOG	The sprint backlog contains key items of a product backlog for the project. Containing the following items: progress project, sprint planning and SCRUM team. The sprint backlog provides product management the progress of the project, the planning, effectiveness of team and watch if requirements are finished/unfinished.
SCRUM TEAM	Every SCRUM team has their own sprint backlog. The product management has a general sprint backlog which contains an overview of the whole project. The performance of the teams can be measured and are depicted as burndown charts.
SPRINT PLANNING	A planning is made for every sprint. A sprint is one month. In every sprint a number of requirements or tasks are assigned to the SCRUM teams.
PROGRESS PROJECT	The product management can keep track of the progress by looking at the number of finished or unfinished requirements.
IDENTIFIED FEATURES	These are the new key features or modules of an upcoming product. Describing how to sell, implement and take care of the logistics of the project evolve around these identified features.

Continued on next page

Table 6 – continued from previous page

Concept	Description
DELIVERABLE LIST	A set of items that needs to be described in order to prepare the departments Marketing & Sales and Services. The deliverable list can be a marketing list or services list. Depending on the department that needs to be prepared.
MARKETING LIST	A list of items describing how to sell and communicate the new features to customers and the market.
SERVICES LIST	A list of items describing of what is needed and how to effectively implement the new features. Another issue is how to advice customers.
RELEASE DEFINITION	This contain the plan how to do the implementation of new features and how to train customers, partners and own employees.

C Backlog Statistics

Sprint	# Tasks	# Hours Start	# Hours End	Avg Hours per Task	Avg Tasks per Person	Avg Hours per Person	Efficiency
2007-03	45	449	154	10.0	9.0	89.8	65.7%
2007-04	57	513	177	9.0	11.4	102.6	65.5%
2007-05	56	527	147	9.4	11.2	105.4	72.1%
2007-06	57	539	102	9.5	11.4	107.8	81.1%
2007-07	112	409	179	3.7	22.4	81.8	56.2%
2007-08	95	574	301	6.0	19.0	114.8	47.6%
2007-09	81	550	215	6.8	16.2	110.0	60.9%
2007-10	58	750	274	12.9	11.6	150.0	63.5%
2007-11	53	544	188	10.3	10.6	108.8	65.4%
2007-12	43	324	142	7.5	7.2	54.0	56.2%
2008-01	60	648	193	10.8	10.0	108.0	70.2%
2008-02	67	497	226	7.4	11.2	82.8	54.5%
2008-03	65	604	250	9.3	10.8	100.7	58.6%
2008-04	84	669	320	8.0	14.0	111.5	52.2%
2008-05	100	619	248	6.2	16.7	103.2	59.9%
2008-06	121	586	198	4.8	20.2	97.7	66.2%
2008-07	85	530	208	6.2	14.2	88.3	60.8%
2008-08	91	443	197	4.9	15.2	73.8	55.5%
2008-10	82	523	176	6.4	13.7	87.2	66.3%
2008-11	81	543	116	6.7	13.5	90.5	78.6%
2008-12	91	455	129	5.0	15.2	75.8	71.6%
Avg	75.4	537.9	197.1	7.7	13.5	97.4	63.3%
Lowest	43.0	324.0	102.0	3.7	7.2	54.0	47.6%
Highest	121.0	750.0	320.0	12.9	22.4	150.0	81.1%

Table 7: Results of the product management sprint backlog analysis