The Relationship Between Scrum and Release Planning Activities: An Exploratory Case Study

Michail Theuns, Kevin Vlaanderen, and Sjaak Brinkkemper

Abstract

In modern product software development settings, it becomes increasingly important to deal with rapid changes in scope, large numbers of users, and regular releases. These circumstances are ideal for an agile development method such as Scrum to prove its value. However, the implications that Scrum has on software product management (SPM) processes have not been investigated in detail. In this paper, we provide more insight into the link between release planning processes and Scrum, by performing a case study at a large Dutch social network provider. The results show an evolutionary approach to the implementation of Scrum, and the relation between several Scrum concepts and SPM capabilities. The findings presented in this paper contribute to more insight into the link between Scrum and SPM and can be of help to product software organizations that employ the Scrum development method.

1 Introduction

In contrast to traditional software packages tailored to satisfy one specific customer, today's software market shows a variety of product software packages that are aimed to serve an entire market with many customers (Regnell and Brinkkemper 2005). Because product software is released for an entire market instead of for just one customer, the development and management of product software is more complex. For example, while a customer-specific software package has to deal with a limited number of requirements coming from just one customer, product

M. Theuns • K. Vlaanderen (✉) • S. Brinkkemper
Utrecht University - Department of Information and Computing Sciences, Utrecht, the Netherlands
e-mail: m.theuns@uu.nl; k.vlaanderen@uu.nl; s.brinkkemper@uu.nl

software has to deal with both an increasing amount of internal and external stakeholders (Ebert 2007), a large amount of requirements and often a much higher release frequency (Weerd et al. 2006).

These circumstances form an ideal environment for agile software development methods. Agile software development methods such as DSDM (Stapleton 1997), Extreme Programming (Beck 1999) and Scrum (Schwaber 1995) enable software companies to dynamically respond to changes in both development environment and target environment (Schwaber 1995). The benefits of agile software development already gained a lot of attention in scientific literature (Dingsøyr et al. 2006; Fitzgerald et al. 2006; Mann and Maurer 2005) and even the applicability of agile principles to the domain of software product management (SPM) gained some attention recently (Vlaanderen et al. 2011), although more research should reveal its applicability to other areas as well (Maglyas et al. 2011). However, the effects or implications of the implementation of agile development methods for a company's SPM processes haven't been investigated in detail yet. In this paper, we describe the implementation of Scrum at a large Dutch online social network provider. Using the situational assessment method (Bekkers et al. 2010), we identify the steps that were taken during the Scrum implementation and the effects it had on the company's SPM processes. The scope of the changes to the SPM processes at the case company is too large to present in this paper entirely. For this reason, we limit our results to the effects of Scrum on the release planning processes. Release planning is often concerned with large amounts of requirements and a high release frequency (Weerd et al. 2006), making it a critical task in the process of developing a successful product. The results can help companies on the verge of implementing an agile development method by providing guidance on how to prepare their SPM processes to facilitate a smooth and successful implementation. In addition, the results form a valuable addition to the knowledge infrastructure (Weerd et al. 2006; Vlaanderen et al. 2011) that is being developed to support product managers in improving their SPM processes.

The remainder of this paper is structured as follows. In the following section, we describe the research approach followed during this project. In Sect. 3, we position our work within existing scientific literature, after which we present our case study results in Sect. 4. We analyze the results in Sect. 5, where we link Scrum elements to release planning capabilities. We conclude with a discussion of our research in Sect. 6, and some pointers towards open research areas in Sect. 7.

2 Case Study Research Design

2.1 Research Question

This research aims at elaborating the relation between Scrum concepts and release planning processes. This information can be of value to companies that struggle with the interaction between agile release planning and the management of software products. By presenting the link between Scrum concepts and (in this case) release planning processes, and the growth in maturity that can be expected when implementing Scrum concepts, companies are given a handhold that shows which release planning capabilities can be implemented by the introduction to Scrum, allowing them to focus on implementing other software product management capabilities. To guide this research, the following main research question is answered throughout this paper:

> How are release planning capabilities related to the activities and concepts within the Scrum development method?

As acknowledged by Levy and Ellis (2006), building a solid theoretical foundation that is based on high quality resources enables researcher to better explain as well as understand the problems under investigation. Hence, we first focus on providing the reader with a clear description of software product management in general and release planning in particular. Furthermore, since this paper aims at discovering the relation between Scrum concepts and release planning processes, we will explain the concept of the Scrum development method and the elements associated with it. The next two subsections explain how we gathered and analyzed our data at the case company.

2.2 Data Gathering

Because we want to examine a phenomenon in its natural setting by gathering information from one or more entities (Benbasat et al. 1987), this research is set up as a case study. The case study is performed at a large Dutch online social network provider, which is explained in more detail in Sect. 4. Several methods for data collection during case studies are described in literature (Yin 2009). For our research, we initially conduct semi-structured interviews with four employees that were actively involved in improving the SPM processes and the implementation of Scrum. In order to obtain a clear and correct understanding of the evolution of the release planning processes at the case company as they implemented Scrum, we interviewed a developer, two product managers, and the head of product (also a member of the company board). The interviews were done in retrospect, meaning that the process improvements that are subject to this research were already implemented at the time of the interviews. However, all of the interviewees have been employed at the case company for at least 3 years, so they were involved in the process improvements from the beginning. This allowed us to gain a complete and correct picture of both operational and strategic processes, and the effects the implementation of Scrum had on these processes. To guide the interviews, a predefined questionnaire was used to ensure we would gather all the data needed to determine the maturity of the release planning processes over time, as we will describe in the next section. The one and a half hour interviews were semi-structured, because they are both well suited for the reconstruction of the process changes, as well as for the exploration of the perceptions and opinions of respondents regarding complex and sometimes sensitive issues. In addition, they enable probing for more information and clarification of answers (Barribal and While 1994). By comparing and contrasting the interview data from several interviewees, along with several related documents, we obtained a complete and correct overview of the evolution of SocialComp's Scrum and SPM processes over time.

2.3 Data Analysis

We first describe the implementation of Scrum, based on what we learned during the interviews. Next, we use the situational assessment method (Bekkers et al. 2010; Bekkers and Spruit 2010) to create four maturity matrices that illustrate the state of the software product management processes at different points in time. The situational assessment method was designed to aid product managers in improving their software product management practices. The maturity matrices present all of the important practices (called capabilities) related to the management of software products in a best practice order for implementation. Each capability is associated with a certain level of maturity, making the maturity matrices a convenient technique to visualize which capabilities are implemented in an organization and which capabilities should ideally be implemented. By comparing the four maturity matrices, we can identify the process improvements that were implemented over a period of 3 years, and thus reveal the evolution of the case company's software product management processes.

Next, we extend the situational assessment method by following a similar approach as used by Weerd, Brinkkemper and Versendaal (2010) and model the release planning processes in process-deliverable diagrams. A process-deliverable diagram consists of a process-side (based on UML activity diagrams) and a deliverable-side (based on UML class diagrams), and can be used to design and analyze the meta-models of methods, revealing both the activities and artifacts of a certain process (Weerd and Brinkkemper 2008). By modeling a snapshot of a process in retrospect and comparing it to a snapshot of the same process in a later point in time, we can identify the process steps or method increments that led to the process's current state. This provides us with much more detail about the release planning processes, and the improvements that were implemented over time. A method increment is basically any adaption in order to improve the overall performance of the method of subject (Weerd et al. 2007).

The result is four maturity matrices and four process-deliverable diagrams of the release planning processes, associated with four distinct points in time. We then compare these 'snapshots' of the software product management processes with the information we gathered about the implementation of Scrum. By analyzing which Scrum elements, and which software product management capabilities were implemented at different points in time, we can reveal the relation between the implementation of various Scrum elements and the evolution of the release planning processes.

3 Related Literature

3.1 Software Product Management

In order to create a profitable software product, software vendors have to take into account all the market requirements coming from the target market. As software products get bigger and more complex, proper management of these software products has become of critical value to the success of the software products (Ebert 2007; Weerd et al. 2006). This led to a new field of research called software product management, which can be defined as "the discipline that governs a product (or solution or service) over its whole lifecycle, from its inception to the market/ customer delivery, in order to generate the biggest possible value to the business" (Ebert 2007). Although software product management has many similarities with product management in other sectors, managing software products is usually more complex due to higher release frequencies, difficulties tracking changes in the design of the software products and the fact that product managers often have little authority over the development department (Weerd et al. 2006). To aid companies in improving their software product management practices, a reference framework called the SPM Competence Model has been developed (Weerd et al. 2006; Bekkers et al. 2010).

The SPM Competence Model (Fig. 1) presents an overview of all the aspects that are important to software product management, including the relevant external and internal stakeholders. The model addresses 15 focus areas, divided over 4 main business functions.

On a strategic level, the software product manager is responsible for managing the product portfolio, by developing product strategies, making decisions about product lifecycles and establishing partnerships with other companies in the software's ecosystem (Jansen, Finkelstein, and Brinkkemper, 2009). The goal of the portfolio management function is to maximize the products' value, spread risks and align with the company's strategy (Cooper et al. 2001).

On a more tactical level, the software product manager is concerned with translating the product strategy into a comprehensive roadmap which forecasts and plans future development steps in terms of release contents, time-to-market and stakeholders involved (Vähäniitty et al. 2002). Hence, the product planning function is mainly concerned with gathering information about a software product (line) and processing this information into product roadmaps that illustrate the ipcoming product releases over a time frame of approximately 3–5 years (Regnell and Brinkkemper 2005), and the use of resources, elements, and their structural relationships in that period (Vähäniitty et al. 2002).

Based on the roadmap, it is the software product manager's task to determine the set of requirements for the next release while keeping in mind all stakeholder demands, effectively managing scope changes to prevent delays and ultimately launch the release to the market. This is done in the release planning function, which comprises the process of selecting an optimal subset of requirements through the prioritization of requirements in accordance with all relevant stakeholders

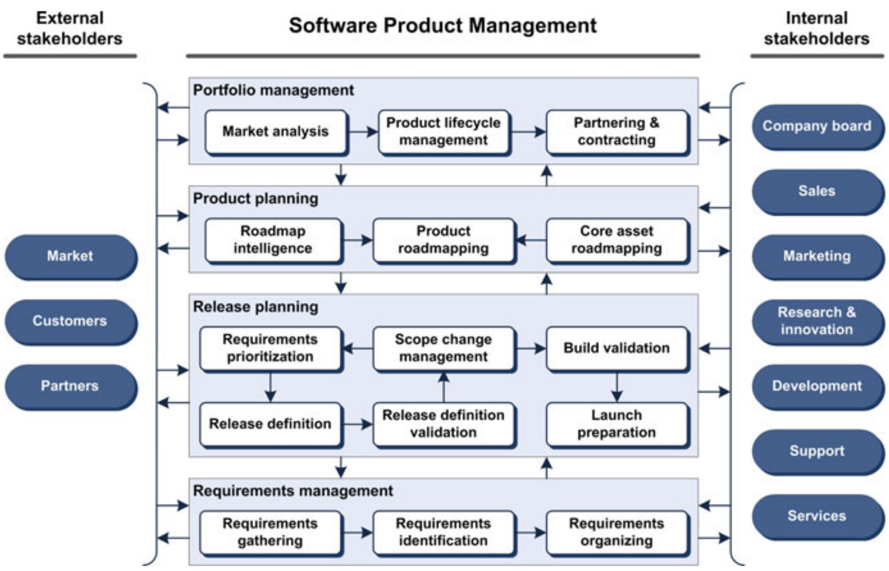

Fig. 1 SPM Competence Model (Bekkers et al. 2010)

(Carlshamre 2002) in order to plan, manage and launch a new release (Bekkers et al. 2010). The release plan contains a detailed description of the requirements to be included in the next release, a planning to ensure the release can be delivered on time, as well as various important technical, resource, budget, and risk constraints (Ruhe and Saliu 2005).

Since a release consists of a multitude of requirements, varying in size and quality, and coming from both internal and external stakeholders, the software product manager is also responsible for effectively managing all requirements. Requirements management encompasses all the activities involved in discovering, documenting, organizing and managing the large volumes of requirements of a software product (Sommerville 2007), the complex dependencies between the requirements (Carlshamre et al. 2001) and the involvement of all the stakeholders (Berander and Andrews 2005). It ensures that requirements are efficiently elicited from all relevant stakeholders (Browne and Rogich 2001), and organized in such a way that they are comprehensible for the development teams.

Each focus area represents a strongly coherent group of predefined goals (also called capabilities) that need to be achieved to reach the maturity levels with which they are associated (Bekkers et al. 2010). To measure the maturity of an organization's SPM processes, a situational assessment method was developed (Bekkers et al. 2010). This situational assessment method employs a capability maturity matrix to determine which capabilities are implemented in an organization and which capabilities should ideally be implemented. A comparison between the current and optimal situation results in an overview of the problem areas that need improvement in order to reach a higher maturity level.

3.2 Scrum

The goal of Scrum is to deliver as much quality software functionality as possible within a series of short time-boxed sprints (Sutherland 2001). Scrum has several distinctive characteristics. The product backlog is a prioritized, non-exhaustive list of functionality to be developed. Usually, product backlog items are not yet well-defined requirements, but rather express functionality in the form of a short description of the feature, defined using the terminology and context of the customer, also referred to as user stories. The product backlog is prioritized by the product owner, so that during the sprint planning meeting, a team of developers can easily pick the top priority items and commit to completing them within the next sprint. The list of top priority features to be developed during the next sprint is called the sprint backlog. During a sprint, the items to be developed are set and cannot change. This helps the development team to remain focused on the goal of the sprint. During a sprint, a storyboard is used to track progress on each sprint backlog item by categorizing them with respect to their status. Furthermore, daily Scrum meetings during which the completed tasks, the work remaining, and any obstacles encountered are discussed help the stakeholders to get an excellent understanding of the sprint progress (Rising and Janoff 2000).

Embedding Scrum within the context of a product software company is not a trivial task. This is recognized by several authors, including one of the founders of the Agile Alliance (Nerur et al. 2005). While agile methodologies can provide significant advantages to a software producing company, there are many challenges that can inhibit a successful move from traditional software development approaches to an agile environment, such as developer resistance, changes in decision making, and the need for increased customer involvement (Boehm and Turner 2005; Moe et al. 2008). Several Scrum implementations have been described in literature. For example, Sutherland (2001) reports on the introduction of Scrum into five different organizations with different technologies. In all five organizations, Scrum improved communication and de-livery of working functionality. Rising and Janoff (2000) describe the experiences three different development teams had with Scrum. The paper acknowledges similar benefits as described by Dingsøyr et al. (2006) and by Mann and Maurer (2005), such as improved customer satisfaction and more flexibility and transparency in the development process. Scrum also proved to be useful in a global, distributed software development environment (Hansen and Baggesen 2009). By employing an virtual task board during online Scrum meetings and moving Product Owners back and forth between the cross-continental locations, they were able to increase code quality and improve trust and understanding between members of distributed development teams, although resource estimation can be a tedious task (Dingsøyr et al. 2006).

Due to the perceived benefits of agile development methods such as Scrum, researchers are now also investigating the applicability of agile principles to other domains. For example, Towill and Christopher (2010) describe the combination of lean and agile principles in supply chain management, while Vlaanderen et al. (2011) apply Scrum principles to SPM to create a regular heartbeat in the SPM process in support of the development process.

4 Case Study Results

4.1 Case Company Introduction

SocialComp was founded in 2004 on the premise of providing a platform through which people could connect and discuss about their everyday lives. Already after 10 months, they reached one million members internationally, with over 80 million page views per month just from the Netherlands. This rapid growth required for several organizational changes. A CEO was appointed to lead the company at corporate level, and several product managers were hired to manage the development department, which grew from only 1 developer in 2004 to over 30 in 2008. After 2008, the number of developers gradually grew to 36, the reason for this declined pace of growth being their high hiring standards and an overall shortage of highly educated developers in the Dutch labor market.

The 36 developers were divided over development teams of approximately 10 developers per team, each team being directed by a product manager. However, despite having appointed a new CEO, the founders stayed actively involved in the development of the social platform. The product managers were often hampered in their work due to conflicting ideas between them and the founders, who were used to address developers directly instead of growing through a layer of product managers. This caused a chaotic working environment, since development teams were often ad hoc assigned to new development projects by the founders, before they could finish their work in progress. During this stage, there was no formal prioritization method in place to determine which projects to develop first. In fact, even a basic process to structurally gather, identify and organize requirements was missing. Prioritization was mainly done according to managers' gut feeling. To make things more complicated, the development teams worked according to the waterfall development model. However, the ever changing requirements made it difficult to finish a phase of the waterfall model since designs often had to be modified to accommodate new requirements. This caused a lot of stress and confusion amongst developers and product managers.

4.2 Implementation of Scrum and Software Product Management

Once the social network grew to be such a large service (over 11 million members in 2011 internationally with over six billion page views per day), the large amount of requirements and the lack of structure of the development department became serious threats to productivity. Therefore, they started searching for alternatives practices to increase productivity and reduce waste of time and resources. Based on the interview results, we could identify four phases in the overall improvement process and the implementation of Scrum and software product management. As described in Sect. 4.1, software product management encompasses many focus areas and capabilities. These focus areas and capabilities are shown in Table 1.

Table 1 Focus areas and capabilities for the release planning domain

Requirements prioritization	Release definition	Launch preparation
RP:A Int. stakeholder involvement	RD:A Basic req. selection	LP:A Internal communication
RP:B Prioritization methodology	RD:B Standardization	LP:B Formal approval
RP:C Customer involvement	RD:C Internal communication	LP:C External communication
RP:D Cost/revenue consideration	RD:D Advanced req. selection	LP:D Training
RP:E Partner involvement	RD:E Multiple releases	LP:E Launch impact analysis
		LP:F Sales & marketing support
Release definition validation	**Build validation**	**Scope change management**
RDV:A Internal validation	BV:A Internal validation	SCM:A Event notification
RDV:B Formal approval	BV:B External validation	SCM:B Milestone monitoring
RDV:C Business case	BV:C Certification	SCM:C Impact analysis
		SCM:D Scope change handling

We chose to only include the changes of the release planning processes in this paper because of two reasons. The main reason is that release planning has many similarities to Scrum and is therefore affected more by the implementation of Scrum than other software product management processes. The second reason is that we wanted to keep this section as concise as possible.

During the first phase (depicted in Table 2 Reference source not found. in the lighter shaded boxes), some capabilities were already implemented, but Scrum and software product management weren't implemented officially yet. Around fall 2010, they began experimenting with Scrum, mainly because of positive prior experiences among some employees. They began by cutting up the fairly large development teams into smaller, independent teams consisting of a maximum of six developers and a product manager, who was assigned the role of product owner. Although the product owners consulted the various stakeholders to determine which set of requirements to develop next, the product owners were now the only ones to direct their development teams. This required the organization to improve its internal communication about the contents of upcoming releases in order to maintain stakeholder satisfaction. This is reflected in Table 2 by 'release definition C' and 'release definition validation A'. 'Requirements prioritization D' shows that they started to take expected costs and revenues of requirements into account during prioritizing, although we could not ascribe this improvement to the implementation of Scrum.

Next, they introduced the 2-week sprints. At the beginning of each sprint cycle, the Product Owners would determine the requirements to be developed during the 2-week sprint during sprint planning meetings. Internal stakeholders were invited to attend the sprint planning meetings and voice their opinion about what to develop next, which is reflected by 'requirements prioritization A' in Table 3. During

Table 2 The evolution of the release planning processes during phases 1 and 2

Process \ Maturity	0	1	2	3	4	5	6	7	8	9	10
Release planning											
Requirements prioritization	▨	▨	A		B	C	■ D	■	■	E	
Release definition			A	B	C	■	■	■	D		E
Release definition validation	▨	▨	▨	▨	■ A	■	■	B		▨ C	▨
Scope change management	▨	▨	▨	A		B		C		D	
Build validation	▨	▨	▨	▨	▨ A	▨	▨	B		C	
Launch preparation	▨	▨ A	▨	B		▨ C	D		E		F

Table 3 The evolution of the release planning processes during phases 2 and 3

Process \ Maturity	0	1	2	3	4	5	6	7	8	9	10
Release planning											
Requirements prioritization			■ A	■	B	C	▨ D	▨	▨	E	
Release definition			■ A	B	▨ C	▨	▨	▨	D		E
Release definition validation	▨	▨	▨	▨	▨ A	▨	▨	B		▨ C	▨
Scope change management				A		■ B	■	C		D	
Build validation					A			■ B	■	C	
Launch preparation	▨	▨ A	▨	B		▨ C	D		E		F

sprints, the set of requirements to be developed is frozen, so it is very important that a set of requirements is chosen that can be delivered when the sprint ends. This is reflected in Table 3 by the implementation of 'release definition A', which means that constraints concerning engineering capacity were taken into account during requirements selection for the next release. Furthermore, the time-boxed nature of Scrum sprints (i.e. the start and end dates are set and do not change) required them to get a much better grip on the development process by monitoring milestones and keeping track of the remaining work. The disorganized funnel was gradually replaced by a structured, prioritized product backlog. This made planning which requirements to develop during the next sprint much easier. In consultation with the Product Owners, development teams could pick a set of requirements from the top of the product backlog, based on the estimated time needed to complete the various requirements. This resulted in several sprint backlogs, for each of the development teams. They introduced a planning board on which the development teams could adjust their sprint backlog items by marking requirements that are completed and estimating the time needed to complete remaining requirements. A burn down chart was introduced to give an overview of the sprint progress. In Table 3, these improvements are reflected by 'scope change management B', which represents the process of milestone monitoring. They also introduced daily standup meetings

Table 4 The evolution of the release planning processes during phases 3 and 4

Process \ Maturity	0	1	2	3	4	5	6	7	8	9	10
Release planning											
Requirements prioritization			A		B	C	D			E	
Release definition			A	B	C				D		E
Release definition validation					A			B		C	
Scope change management				A		B		C		D	
Build validation					A			B		C	
Launch preparation		A		B		C	D		E		F

of approximately 15 min, in which team members were asked to discuss what they did the day before, what they plan on doing today and what obstacles may have occurred. This helped them to gain better understanding of the work that has been done and the work that still remains. Finally, the introduction of demo meetings at the end of each sprint allowed for stakeholder involvement during the validation of the built functionality (‘build validation B’). In this case, partner companies were even allowed the opportunity to test functionality before it was released to the public. Table 4 shows the fourth phase of the improvement process, which represents the current state at the case company. Two more improvements could be identified recently. Although the implementation of Scrum already contributed to the improvement of their requirements prioritization process, it was formalized during this phase with the introduction of Scrum’s planning poker. Everyone in the organization, whether he is a developer, a product manager or a member of the board, now knows how to voice his opinion about the contents of upcoming releases without hampering the development process. This was not yet the case in the third phase, which is why we included ‘requirements prioritization B’ in Table 4. Furthermore, the contents of the release definition (i.e. the sprint backlog) became more structured by adding aspects such as the time path and needed capacity. This can also be attributed to the formalization process, and is reflected by ‘release definition B’ in Table 4

5 Analysis

Since Scrum was largely implemented during the transition from phase 2 to phase 3, we decided to take a closer look into the actual changes that occurred during the transition and to visualize the release planning evolution in the PDD depicted in Fig. 2. The left-hand side of the PDD shows activities performed during the method (based on a UML activity diagram), whereas the right-hand side of the PDD shows the concepts delivered by the activities (based on a UML class diagram). According to the modeling conventions used by Weerd, Brinkkemper and Versendaal (2007),

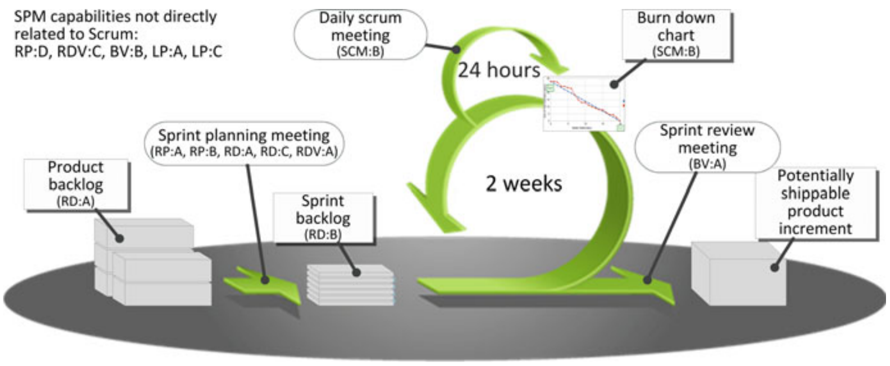
SPM capabilities not directly related to Scrum:
RP:D, RDV:C, BV:B, LP:A, LP:C
Daily scrum meeting (SCM:B)
24 hours
Burn down chart (SCM:B)
Sprint planning meeting (RP:A, RP:B, RD:A, RD:C, RDV:A)
Product backlog (RD:A)
Sprint backlog (RD:B)
2 weeks
Sprint review meeting (BV:A)
Potentially shippable product increment

Fig. 2 Revealing the method increments of the release planning process (from phase 2 to 3)

we used gray markings to visualize the method increments that were inserted, whereas the shadings exhibit deleted parts. Note that the gray marked areas correspond with the darker shaded boxes in Table 3.

Based on our observations, we could relate several SPM capabilities implemented at SocialComp to Scrum concepts (see Fig. 3). The shadowed boxes represent Scrum concepts such as the product and sprint backlogs, the burn down chart and the product increment the sprint delivers. The rounded boxes symbolize Scrum activities such as the various Scrum meetings. It provides a simple overview of the capabilities for which Scrum concepts and activities can account.

As described earlier, sprints are time-boxed events. Hence, it is important to choose a set of requirements that can be completed during one sprint. Before the process improvements started, developers were often hampered in their work by intervening requirements coming from the management team. The implementation of the product backlog required them to estimate the time and resources required for each backlog item, which is why we associated ‘release definition A’ with the implementation of the product backlogs. As sprints came closer, more detailed information was added to the backlog items, and sprint backlog items became more standardized. Hence, we can say that the introduction of sprint backlogs is associated with the implementation of ‘release definition B’ which stands for standardization of release contents.

The sprint planning meetings could be associated with multiple capabilities. Whereas the management team was used to address developers directly, passing by the product managers, the sprint planning meetings created a place to discuss the contents of upcoming releases and sprints. It allowed internal stakeholders to voice their opinion about which requirements to develop (‘requirements prioritization A’; ‘release definition A’), without hampering the development process. Since the internal stakeholders were involved in the process of selecting and prioritizing requirements, ‘release definition validation A’ was automatically covered. Naturally, the implementation of the sprint planning meetings improved internal communication (‘release definition C’) greatly.

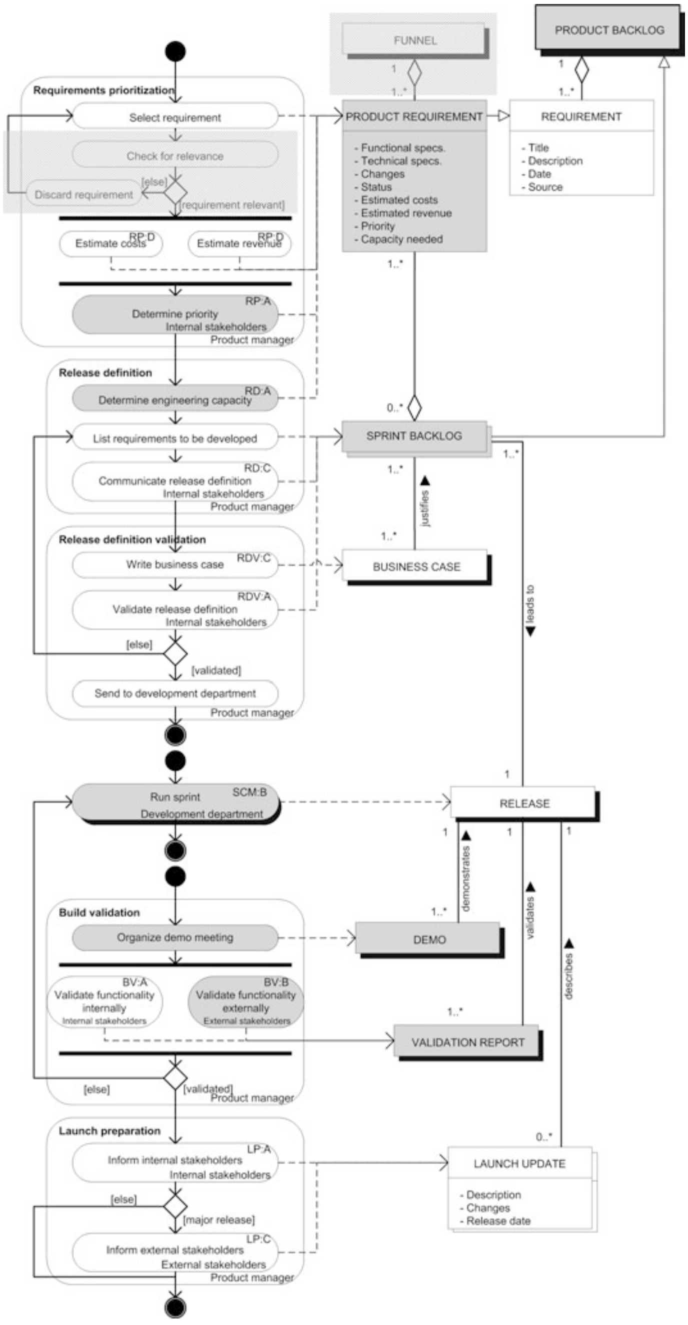

Fig. 3 Relating the SPM capabilities to Scrum concepts

The capability 'scope change management B' was associated with both the daily standup meetings and the burn down chart. The capability reflects the process of monitoring milestones in the development process, which is exactly what they gained from introducing daily standup meetings and the burn down chart. It allowed them to get a better grip on the development process, to identify possible difficulties early on, and monitor the development progress by updating the burn down chart when a task is finished.

'Build validation A' could be associated with the introduction of sprint review meetings, during which the development teams would demonstrate built functionality to the product managers and the management team.

Note that of all the release planning capabilities implemented, five could not be associated with the implementation of Scrum. For example, 'requirements prioritization D' was implemented in phase 2, which means they started to take prospected costs and revenues into account during the prioritization process. There is no Scrum element that prescribes cost/revenue consideration, and it was mainly caused by a change of management. The same holds for the business plan that was introduced to justify for each release plan ('release definition C'). Both were improvements imposed by the new management, who wanted to base their decisions on financial figures rather than on their gut feeling.

Furthermore, allowing partner companies, e.g. companies with a branded marketing campaign on the social network site, to test new functionality before going live ('build validation B') was a way of improving customer satisfaction rather than something prescribed by Scrum. Lastly, SocialComp has always communicated about upcoming releases. Internal (e.g. management or the sales department) and external (e.g. the users) stakeholders were kept informed about upcoming releases and new functionality ever since the start of SocialComp. This is also visible in Table 2, where 'launch preparation A' and 'launch preparation C' were already implemented in the first phase.

6 Discussion and Limitations

Although we used the four validity criteria as described in (Yin 2009) to ensure the quality and reliability of our work., it should be noted that this research is subject to some limitations. The findings presented in this paper are based on information regarding the SPM processes at one case company, posing a threat to the external validity. The external validity entails the possibility to generalize the research findings, so validating our findings at other web companies that recently implemented Scrum should eliminate this threat. Furthermore, although we also modeled the other business functions, we only studied the release planning processes in detail. Consequently, we only identified the relations between Scrum elements and release planning processes. In order to get a more comprehensive view of the relation between Scrum and SPM, future research should include the business functions requirements management, product roadmapping and portfolio management as well.

The construct validity, which concerns the operationalization and correct measurement of the concepts being studied, is safeguarded by validating the maturity matrices and PDDs that were created based on the data collected during the interviews in a second round of interviews. The internal validity concerns the completeness of concepts and the consistency between concepts. This is partly satisfied because we double-checked our information gathered and partly because we were able to link Scrum concepts to SPM capabilities. It should be noted though, that these links should be validated in follow-up research to completely satisfy internal validity. Furthermore, the case study report was reviewed by peer-researchers to ensure a reliable research approach. The empirical validity concerns the reproducibility of the research and is preserved by following a case study protocol. Furthermore, the interview results can be reproduced easily because we based the interviews on the situational assessment method (Bekkers et al. 2010).

Finally, the situational assessment method employs a questionnaire and a maturity model to determine the capabilities implemented in an organization and to reveal the areas that need improvement. We noticed during interviewing, that some of the capabilities are not applicable to agile software companies with a very informal organizational culture (for example, capabilities that prolong decision-making because all stakeholders have to get involved). By extending the situational assessment method with PDDs, we gain more insight in the actual processes and the associated capabilities. By modeling the processes, we were able to determine SocialComp's SPM maturity more accurately. Furthermore, the method offers a way to translate the maturity matrix to PDDs by adding information regarding the implemented capabilities to the activities depicted in the PDDs. A difficulty associated with the model-driven assessment method is that it is often challenging to get a correct picture of the entire process, since processes can be very complex with multiple concurrent activities and related concepts. While PDDs provide more detail about a certain process, areas that need improvement are not as conspicuous as in the maturity matrix. Furthermore, revealing method increments in PDDs is somewhat cumbersome, because the approach officially dictates that deleted and adapted activities or concepts should still be modeled.

7 Conclusions and Further Research

In this paper, we have described the results from a case study performed at a large, Dutch social network provider. We have gathered data regarding the implementation of Scrum, and the linkage between Scrum concepts and SPM capabilities. This data has been used to provide more insight into the effects of Scrum on SPM, and the co-evolution of Scrum and SPM when Scrum is implemented in an incremental manner.

The research presented in this paper forms yet a step towards a knowledge infrastructure that helps product managers in incrementally improving the SPM processes in their organizations. As noted by Vlaanderen et al. (2010), in order to establish a successful product software knowledge infrastructure, it is important to determine how certain methods in the SPM domain can change, what method increments are commonly found in practice and how method fragments can be analyzed. This research forms an addition to the existing knowledge base on method increments found in practice.

We are convinced that case study descriptions such as the one presented in this paper are a valuable addition to both scientific as well as the industrial software engineering field. However, in order to make such descriptions more concise and better comparable, we are in need of a more structured approach of modeling increments. Such an approach should be able to reflect changes in the process in relation to organization's contextual factors. Moreover, this research further matures the scientific literature on Scrum and release planning by revealing the link between the two. Providing insight into the association between Scrum concepts and the implementation of SPM capabilities can be of vital help to companies that want to implement either or both, as it provides insight into the maturity levels that can be expected.

References

Barribal, K. L., & While, A. (1994). Collecting data using a semi-structured interview: A discussion paper. *Journal of Advanced Nursing, 19*(2), 328–335. doi:10.1111/j.1365-2648.1994.tb01088.x.

Beck, K. (1999). *Extreme programming explained: Embrace change*. Boston: Addison-Wesley.

Bekkers, W. & Spruit, M. (2010). The situational assessment method put to the test: Improvements based on case studies. *Proceedings of the 4th international workshop on software product management, Sydney* (pp. 7–16). doi: 10.1109/IWSPM.2010.5623871

Bekkers, W., Spruit, M., Weerd, I. van de, Vliet, R. van, & Mahieu, A. (2010). A situational assessment method for software product management. *Proceedings of the 18th European conference on information systems, Pretoria* (pp. 22–34). Retrieved from http://dblp.uni-trier.de/db/conf/ecis/ecis2010.html

Bekkers, W., van de Weerd, I., Spruit, M., & Brinkkemper, S. (2010). A framework for process improvement in software product management. In A. Riel, R. O'Connor, S. Tichkiewitch, & R. Messnarz (Eds.), *Systems, software and services process improvement* (pp. 1–12). Berlin/Heidelberg: Springer. doi:10.1007/978-3-642-15666-3_1.

Benbasat, I., Goldstein, D. K., & Mead, M. (1987). The case study research strategy in studies of information systems. *MIS Quarterly, 11*(3), 369–386. doi:10.2307/248684.

Berander, P., & Andrews, A. (2005). Requirements prioritization. In A. Aurum & C. Wohlin (Eds.), *Engineering and managing software requirements* (pp. 69–94). Berlin/Heidelberg: Springer.

Boehm, B., & Turner, R. (2005). Management challenges to implementing agile processes in traditional development organizations. *IEEE Software, 22*(5), 30–39. doi:10.1109/MS.2005.129.

Browne, G. J., & Rogich, M. B. (2001). An Empirical Investigation of User Requirements Elicitation: Comparing the Effectiveness of Prompting Techniques. Journal of Management Information Systems, 17(4), 223–249.

Carlshamre, P. (2002). Release planning in market-driven software product development: Provoking an understanding. *Requirements Engineering, 7*(3), 139–151. doi:10.1007/s007660200010.

Carlshamre, P., Sandahl, K., Lindvall, M., Regnell, B., & Natt och Dag, J. (2001). An industrial survey of requirements interdependencies in software product release planning. *Proceedings of* *the 5th IEEE international symposium on requirements engineering, Toronto* (pp. 84–91). doi: 10.1109/ISRE.2001.948547

Cooper, R. G., Edgett, S. J., & Kleinschmidt, E. J. (2001). Portfolio Management for New Product Development: Results of an Industry Practices Study. R&D Management, 31(4).

Dingsøyr, T., Hanssen, G. K., Dybå, T., Anker, G., & Nygaard, J. O. (2006). Developing software with scrum in a small cross-organizational project. In I. Richardson, P. Runeson, & R. Messnarz (Eds.), *Lecture notes in computer science* (Software process improvement, Vol. 4257, pp. 5–15). Berlin/Heidelberg: Springer. doi:10.1007/11908562_2.

Ebert, C. (2007). The impacts of software product management. *Journal of Systems and Software, 80*(6), 850–861. doi:10.1016/j.jss.2006.09.017.

Fitzgerald, B., Hartnett, G., & Conboy, K. (2006). Customising agile methods to software practices at Intel Shannon. *European Journal of Information Systems, 15*(2), 200–213. doi:10.1057/palgrave.ejis.3000605.

Hansen, M. T., & Baggesen, H. (2009). From CMMI and isolation to scrum, agile, lean and collaboration. *Proceedings of the agile conference, Chicago* (pp. 283–288). doi: 10.1109/AGILE.2009.18

Jansen, S., Finkelstein, A., & Brinkkemper, S. (2009). A sense of community: A research agenda for software ecosystems. International Conference on Software Engineering (pp. 187–190). IEEE.

Levy, Y., & Ellis, T. J. (2006). A systems approach to conduct an effective literature review in support of information systems research. *Informing Science, 9*(1), 181–212. Retrieved from http://www.informingscience.us/icarus/journals/informingscij

Maglyas, A., Nikula, U., & Smolander, K. (2011). What do we know about software product management?. A systematic mapping study. *Proceedings of the 5th international workshop on software product management, Trento* (pp. 26–35). doi: 10.1109/IWSPM.2011.6046201

Mann, C., & Maurer, F. (2005). A case study on the impact of Scrum on overtime and customer satisfaction. *Proceedings of the agile development conference, Denver* (pp. 70–79). doi: 10.1109/ADC.2005.1

Moe, N. B., Dingsøyr, T., & Dybå, T. (2008). Understanding self-organizing teams in agile software development. *Proceedings of the 19th Australian conference on software engineering, Perth* (pp. 76–85). doi: 10.1109/ASWEC.2008.4483195

Nerur, S., Mahapatra, R., & Mangalaraj, G. (2005). Challenges of migrating to agile methodologies. *Communications of the ACM – Adaptive Complex Enterprises, 48*(5), 72–78. doi:10.1145/1060710.1060712.

Regnell, B., & Brinkkemper, S. (2005). Market-driven requirements engineering for software products. In A. Aurum & C. Wohlin (Eds.), *Engineering and managing software requirements* (pp. 287–308). Berlin/Heidelberg: Springer.

Rising, L., & Janoff, N. S. (2000). The scrum software development process for small teams. *IEEE Software, 17*(4), 26–32. doi:10.1109/52.854065.

Ruhe, G., & Saliu, M. O. (2005). Supporting software release planning decisions for evolving systems. *Proceedings of the 29th annual IEEE/NASA software engineering workshop, Greenbelt* (pp. 14–26). doi: 10.1109/SEW.2005.42

Schwaber, K. (1995). SCRUM development process. *Proceedings of the conference on object-oriented programming systems, languages, and applications, workshop on business object design and implementation* (pp. 117–134).

Sommerville, I. (2007). *Software engineering* (8th ed.). Boston: Addison-Wesley.

Stapleton, J. (1997). *DSDM: Dynamic systems development method*. Boston: Addison-Wesley.

Sutherland, J. (2001). Agile can scale: Inventing and reinventing scrum in five companies. *Cutter IT Journal, 14*(12), 5–11. Retrieved from http://www.cutter.com/itjournal.html

Towill, D., & Christopher, M. (2010). The supply chain strategy conundrum: To be lean or agile or to be lean and agile? *International Journal of Logistics Research and Applications, 5*(3), 299–309. doi:10.1080/1367556021000026736.

Vähäniitty, J., Lassenius, C., & Rautiainen, K. (2002). An Approach to Product Roadmapping in Small Software Product Businesses. *Conference Notes of the 7th European Conference on Software Quality* (pp. 12–13). Helsinki: Finland.

van de Weerd, I., & Brinkkemper, S. (2008). Meta-modeling for situation analysis and design methods. In M. R. Syed & S. N. Syed (Eds.), *Handbook of research on modern systems analysis and design technologies and applications* (pp. 38–58). Hershey: Idea Group Publishing.

van de Weerd, I., Brinkkemper, S., & Versendaal, J. (2007). Concepts for incremental method evolution: Empirical exploration and validation in requirements management. In J. Krogstie, A. Opdahl, & G. Sindre (Eds.), *Lecture notes in computer science* (Advanced information systems engineering, Vol. 4495, pp. 469–484). Berlin/Heidelberg: Springer.

van de Weerd, I., Brinkkemper, S., & Versendaal, S. (2010). Incremental method evolution in global software product management: A retrospective case study. *Information and Software Technology, 52*(7), 720–732. doi:10.1016/j.infsof.2010.03.002.

Vlaanderen, K., Weerd, I. van de, & Brinkkemper, S. (2010). Model-driven assessment in software product management. *Proceedings of the 3rd international workshop on software product management, Sydney* (pp. 17–25). doi: 10.1109/IWSPM.2010.5623868

Vlaanderen, K., Jansen, S., Brinkkemper, S., & Jaspers, E. (2011). The agile requirements refinery: Applying scrum principles to software product management. *Information and Software Technology, 53*(1), 58–70. doi:10.1016/j.infsof.2010.08.004.

Vlaanderen, K., Weerd, I. van de, & Brinkkemper, S. (2011). The online method engine: From process assessment to method execution. *Proceedings of the IFIP WG 8.1 working conference on method engineering, Paris* (pp. 108–122).

Weerd, I. van de, Brinkkemper, S., Nieuwenhuis, R., Versendaal, J., & Bijlsma, L. (2006). On the creation of a reference framework for software product management: Validation and tool support. *Proceedings of the 1st international workshop on software product management, Minneapolis/St. Paul* (pp. 3–12). doi: 10.1109/IWSPM.2006.6

Weerd, I. van de, Versendaal, J., & Brinkkemper, S. (2006). A product software knowledge infrastructure for situational capability maturation: Vision and case studies in product management. *Proceedings of the 12th working conference on requirements engineering: Foundation for software quality* (pp. 97–112). Luxemburg: Luxemburg

Yin, R. K. (2009). *Case study research: Design and methods* (4th ed.). London: Sage Publications.