Continuous Release Planning in a Large-Scale Scrum Development Organization at Ericsson

Ville T. Heikkilä[1], Maria Paasivaara[1], Casper Lassenius[1], and Christian Engblom[2]

[1] Department of Computer Science and Engineering, Aalto University, Helsinki, Finland
{ville.t.heikkila,maria.paasivaara,casper.lassenius}@aalto.fi

[2] Oy LM Ericsson Ab, Kirkkonummi, Finland
christian.engblom@ericsson.com

Abstract. Scrum development at large-scale requires a release planning process that supports the agile way of working and planning. Most of the existing release planning processes are plan-driven and ill suited for a large Scrum organization. This case study describes how release planning was conducted in a 350-person Scrum development organization with over 20 teams at Ericsson in 2011, and the related challenges and benefits. Data was collected with 39 interviews which were transcribed, coded and analysed. The release planning process was continuous and characterized by regular scoping and prioritization decisions, and by incremental elaboration of features. The challenges were the overcommitment caused by external pressure, managing non-feature specific work, and balancing between development efficiency and building generalist teams. The benefits were the increased flexibility and decreased development lead time, waste eliminated in the planning process, and increased developer motivation.

Keywords: release planning, scrum, scaling agile, case study.

1 Introduction

The Scrum agile software development method [1] has become mainstream in the software development community [2]. Scrum was originally created for small co-located teams [1]. Scrum emphasizes face-to-face communication [1], which puts a limit on the maximum practical size of the development team [3]. The early normative Scrum literature provided little guidance for the long-term planning of software, as the focus was on the planning and development of software one iteration (sprint) at a time in a single team, single project context [1]. However, large development organizations soon started to adopt Scrum practices [2]. In large organizations, there are multiple levels of planning which are performed on different time horizons and by different actors [4,5,6]. We adopt a three-level planning model where the levels are *strategic planning*, *release planning* and *operational planning*. Strategic planning is the interface between business management and development and it is performed on a long term, multi-release time horizon [4]. Release planning, in agile software development context, is concerned with deciding the feature content of the next release and on planning how to most efficiently create that content [7]. Operational planning is concerned with how the implementation of the features is achieved on a day-to-day basis [5]. The early Scrum literature describes operational planning in depth, superficially covers release planning, and almost completely ignores strategic planning [1].

One way to scale a Scrum development organization is to employ multiple small Scrum teams [6,8,9]. In a such organization, the strategic planning is mostly agnostic towards the development method [4] and the operational planning can follow the Scrum practices [1]. However, release planning must support the Scrum development organization by providing goals and direction on how the release should be constructed [4]. Although successful release planning is an important success factor in agile software development projects [10] and a challenging aspect of agile adoption in market-driven product development [11], there is very little empirical research literature of large-scale agile release planning. Thus, there is a clear need for empirical research that describes how release planning is conducted in large agile organizations. To start filling this gap in the empirical research, we conducted a case study in a large organization that had adopted Scrum. The case organization was a node development organization of Ericsson. The specific research questions were:

RQ1: What was the release planning process?
RQ2: What were the challenges related to the release planning process?
RQ3: What were the benefits of the continuous release planning process?

The rest of this paper is organized as follows: We first review existing related work on release planning in large-scale agile development organizations in Section 2. We describe our research methods in Section 3. We describe the case organization in Section 4. In Section 5, we describe the results of the case study. In Section 6, we discuss the case and threats to the validity of our results. Finally, in Section 7, we provide conclusions and directions for future work.

2 Release Planning in Large-Scale Agile Development

Most release planning research has focused on proposing mathematical optimization models [12]. This approach has resulted in models which either are too simple to be useful in practice, or so complex that practitioners find it difficult to provide the necessary input values and find it hard to trust the output, as they cannot comprehend the process that created it [13,14,15]. In addition, the models typically contain assumptions which do not hold in many software development organizations; the models assume a common understanding of requirements, while in reality such understanding arises thorough continuous knowledge generation and sharing. The models assume that the requirements selection criteria are stable, while in reality the criteria and their weights may change over time. The models assume that dependencies between requirements are clearly

Table 1. Details of the data collection

Interviews	39 (Finland 28, Hungary 11)
Roles[a]	Middle and upper managers (6), Agile coach (1), Scrum Masters (6), Developers (13), Line managers (3), Product owners (7), Technical specialists / architects (5)
Interview lengths	Managers & the coach: 2-3h, others: 1-2h

[a]Total of 41 roles. Two interviewees had dual roles.

defined and pairwise, while in reality the dependencies are often unclear and complex. Finally, the models assume that development capacity is the main constraint, while in reality combined domain and system knowledge often is the critical resource [15].

The existing empirical research on release planning in large-scale agile development is scarce. When using Scrum, large-scale release planning can be performed in joint release planning sessions where all development teams and other stakeholders come together to plan the next release [6,16].

3 Data Collection and Analysis

Our case was purposefully selected, as it provided an opportunity to perform an information rich study [17,18] in a large organization with a long history of developing a complex product. The organization had adopted Scrum gradually over the 18 months preceding the interviews, which made the project an excellent candidate for the study. We first interviewed nine people who had managerial roles (who were our *key informants* [18]). They provided us with an overview of the organization history, goals, growth, structure and the planning process used in the organization. To enable the triangulation of data sources [17], the rest of the interviewees had different roles, belonged to different Scrum teams and had different amounts of experience. All interviews were voice-recorded. The first three authors conducted the interviews. We selected the *general interview guide* approach [17] in order to maintain adaptability to the roles and individual experiences of the interviewees while simultaneously making sure that the relevant topics were explored. We updated the interview guide constantly based on new insights from the previous interviews [17]. We asked the interviewees to describe their own experiences of how the release planning process worked and the successes and challenges in the release planning process. Details of the data collection are shown in Table 1.

The interviews were transcribed by a professional transcription company. We coded the interviews with a process that was inspired by the grounded-theory method [19]. During the coding process, we combined related concepts into new concepts and categories using the constant comparison technique [19]. In total, we coded 625 passages (with some minor overlap). Finally, we extracted passages related to categories of planning and organization and re-read all the passages to construct the descriptions of the release planning process and the challenges and benefits related to it. The challenges and benefits included in the results were perceived as the most important by multiple interviewees from multiple roles in the organization.

4 The Case Organization

4.1 Background

This paper is based on a case study of an Ericsson node development unit in 2011. The unit developed a large systems product consisting of both software and hardware. The product was a single node which handled specific type of traffic in telecommunications networks. The development of this product had started over ten years ago and at the time of the study it was used by operators all over the world, while the further development of the product still continued. The focus of this paper was on the organization that developed the software of the product.

The organization begun the process improvement initiative in 2009. The existing, plan-driven process worked quite well, but the management wanted to decrease the development lead time, improve flexibility, increase motivation of the developers, and increase the efficiency of quality assurance. The management studied different options and chose Scrum as a best fit for their needs. They started with one pilot Scrum team to test the approach. Soon a few more teams were created, and in quick succession the rest of the Scrum teams were formed. At the time of the interviews in 2011, all of the over twenty development teams, spread across two sites in Finland and Hungary, had been using Scrum for almost a year. The transformation did not stop there and the way of working was continuously improved, as reflected by the interviewees who called the transformation a “journey”.

Before the transformation, the development had been arranged as a traditional plan-driven project organization. The release planning had begun two years before the release date when the scope of the next release was decided by the product management. Technical specialists then created an implementation plan for the requirements and the plans were handed to the developers for implementation. When the implementation was ready, the software was put thorough multiple stages of testing and verification, and finally shipped as a part of the generally available products and as software updates.

4.2 Case Organization Structure

The development organization and its stakeholders are illustrated in Figure 1. In the rest of this section, we describe the roles and responsibilities of the members of the case organization.

Product Owner Team. The organization of product owners deviated from the basic Scrum model [1]. Instead of having team-specific product owners, a product

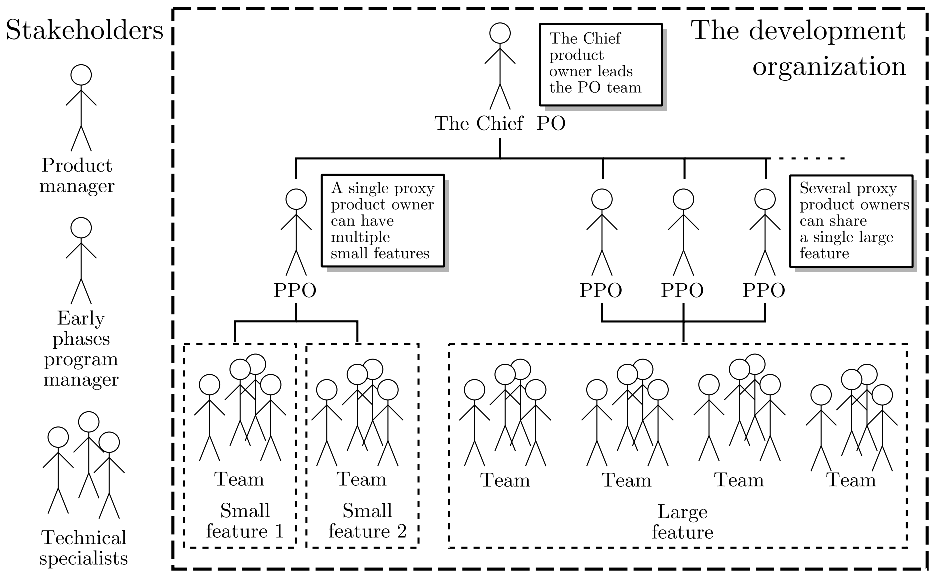
Stakeholders
Product manager
Early phases program manager
Technical specialists
The development organization
The Chief PO
The Chief product owner leads the PO team
PPO
A single proxy product owner can have multiple small features
PPO
PPO
PPO
Several proxy product owners can share a single large feature
Team
Small feature 1
Team
Small feature 2
Team
Team
Team
Team
Large feature

Fig. 1. The case organization structure

owner team (PO team) had been created to accommodate the large number of teams and the globally distributed structure of the organization. The PO team consisted of a Chief product owner (Chief PO) and ten proxy product owners (PPOs). The whole PO team was jointly responsible for the feature development to mitigate personnel risks. The PPOs rotated between teams when features were completed. Based on the size of the features, one PPO could work with two of cross-functional teams when both teams were developing their own small features, or a group of two to three PPOs could take collectively responsibility of one large feature developed by several teams. The Chief PO was responsible for managing the PO Team. The Chief PO acted as an arbiter between the development organization and the stakeholders external to the development organization. The main task of the PO team was to manage and synchronize the work of the development teams.

Development Teams. The Scrum development organization was arranged around Scrum development teams [1]. These teams of 6-7 persons were originally formed with the goal of having all the needed competence for end-to-end development in each team. However, the managers soon realized that this goal would be very challenging to achieve with a large, over ten-year-old product, since the different areas of the product required very specific technical knowledge. Thus, the development teams, in practice, were usually assigned features that were best suited for the competence and previous experience of the team members. The teams had different amounts of experience of Scrum development practices and of working in a cross-functional way. The teams on both sites were located near each other to allow the teams to easily visit each other.

Stakeholders. There were several stakeholders that had an important role in the release planning process, but who did not belong to the development organization. A product management function was responsible for the long term planning of the product from the business perspective. A single product manager (PM) was responsible for the software part of the product and she was also the main point of contact between the development organization and the product management. The PM mostly communicated with the Chief PO although the other members of the development organization could contact the PM directly, if required. In addition, there was an early phases program manager who was responsible for managing the early phases of the release planning process. The product management and the development organization were assisted by technical specialists who were people with extensive knowledge of telecommunication technology.

5 Results

In this section, we present our results. In Section 5.1, we describe the work items, decision makers and process steps of the release planning process. We describe challenges found in the case in Section 5.2 and benefits created by the release planning process in Section 5.3.

5.1 Release Planning Process

The new continuous release planning process, depicted in Figure 2, contained five feature decisions, F0 to F4. The decisions were made by two steering groups, which both had weekly meetings. Decisions F0-F2, which were made by *the portfolio steering group*, belonged to the *early phases* where the feasibility, profitability and risk of the feature were studied and no actual feature implementation was performed. Decisions F3-F4 were made by *the development steering group.*

The public releases of the software were tied to the calendar year. Two major versions of the software and two smaller maintenance updates of the software were published each year. The new development model would have allowed more frequent public releases, but the customers preferred the aforementioned release schedule. The contents of each release were tentatively planned by the product management. The actual contents of a release were based on the features that were completed in time. Following the F-process, those features which had passed the F4-milestone could be included. In the rest of this section, we describe the steps, the planning artefacts, and the stakeholders of the release planning process in detail.

Work Items. The case organization had three level work item hierarchy. The levels were called *features*, *epics* and *user stories.* All three level work items were stored in the product backlog, which was in an electronic backlog management tool Jira.

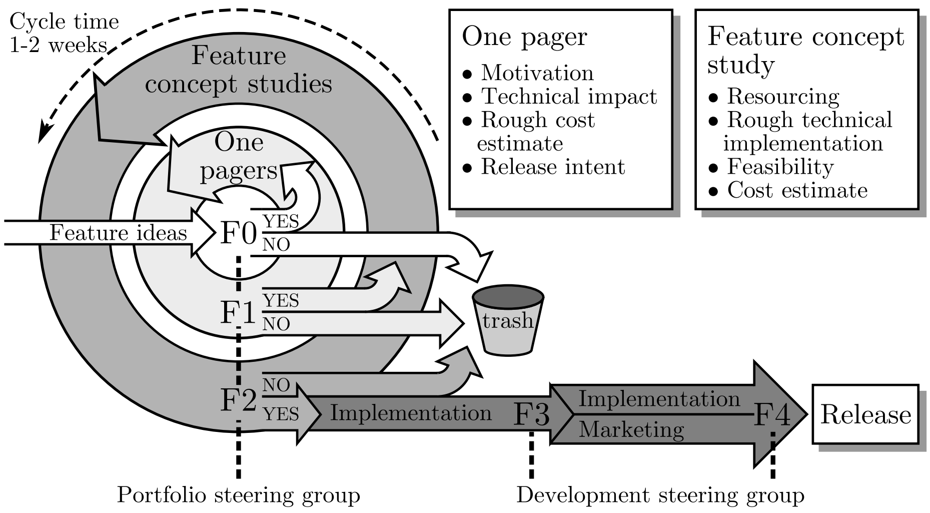

Fig. 2. The release planning process

Features were the main way that requirements were managed in the new process. The Chief PO was responsible for maintaining and prioritizing the product backlog on the feature level. The size of features varied considerably from a single team for a couple of months to a year for ten teams.

Epics were split from features. Epics were large, semi-independent functional requirements that produced value on their own. Each epic was typically team-specific. The purpose of the epic-level was to decrease dependencies between teams working on the same feature by grouping related user stories, and to provide a view to the multi-sprint development plan of the feature. Epics were split into user stories by the PPO(s) responsible for the feature together with the team responsible for the epic in the team's bi-weekly grooming sessions. The user stories were also estimated in the grooming sessions. The teams had a physical sprint backlog which contained the team's user stories.

Portfolio Steering Group. The portfolio steering group consisted of representatives from all stakeholders of the development organization, including the product manager, the Chief PO, the early phases program manager and the technical specialists. The group made decisions related to the early phases of feature development. Any number of F0-, F1- and F2-decisions could be made in a single portfolio steering group meeting, which were held once a week.

Development Steering Group. The development steering group included the Chief PO and the product manager. Selected PPOs and other stakeholders from the organization could also participate in the meetings if deemed necessary. The group made decisions during the implementation of the feature. Any number of F3- and F4-decisions could be made in a single development steering group meeting, which were held once a week.

F0-Decision. The first step in the F-decision process was the F0-decision. Before the F0-decision, the product manager and the early phases program manager had to have a very rough idea of the feature. When time was right, the early phases program manager presented the feature idea in a portfolio steering group meeting. The steering group then decided to either take the feature into development, to postpone it, or to abandon it. If the feature was postponed, it would be brought into F0-decision later on. If the feature was taken into development, the creation of a one pager could begin.

One Pager. The one pager described on an abstract level what the feature was and why it was needed, including the rough estimates of the cost and business impact of the feature, and the intended release which would tentatively include the feature. The goal was to fit all the information on a single presentation slide, hence the name one pager. The early phases PM had the official responsibility of creating the one pager, but in practice it was written by a technical specialist. The maximum effort of writing the one pager was one or two days, and the time given to the writing was two weeks.

F1-Decision. The F1-decision could be made after the one pager was ready. The Chief PO presented the one pager to the group. The portfolio steering group then decided to either to abandon the feature or to create a feature concept study (FCS). If the portfolio steering group decided to create the FCS, the group could then decide either to initiate the FCS immediately or to postpone it, based on how urgent and large the feature was. In case the FCS was initiated immediately, the steering group selected a PPO and team(s) for writing it. If the FCS was postponed, the assignment would be made in a later portfolio steering group meeting. If the feature was not abandoned, the feature was added to the product backlog and prioritized by the Chief PO.

Feature Concept Study. The writing of the feature concept study begun after the F1-decision to initiate it had been made. The purpose of the FCS was to provide the information that was needed to decide whether the feature should be implemented. The FCS was written by a virtual team consisting of a PPO, who was primarily responsible for the study, and members from one or several teams. The virtual team members were typically from the teams that would be assigned to implement the feature. The virtual team was assisted by the technical specialists if required. Having developers to contribute to the FCS was a notable change from the previous planning process where the developers did not contribute to requirements planning. The length and the writing time of a FCS varied by the size of the feature, but the goal was to get the writing done in under two weeks.

F2-Decision. The F2-decision was the last step in the early phases program. When the feature concept study was ready, the Chief PO presented it to the portfolio steering group, which then decided to either take the feature into development or to abandon it. If the feature was taken into development, it was given a non-binding target release. The Chief PO had the option to postpone the beginning of the implementation if he though that there was more important work and the target release could be reached nevertheless. Otherwise, the team(s) begun the development of the feature immediately.

Feature Implementation. The implementation of a feature could officially begin after a F2-decision to implement the feature was done and when the Chief PO decided it was time to start the implementation. Typically, the implementation was started by the PPO and the team(s) that created the FCS. In large features, more teams were added during the implementation when they become available. When the feature neared completion, the number of teams was reduced to one or two teams that were responsible for finalizing the feature, and the rest of the teams were freed to develop other features. On the team level the planning was performed mostly following the basic Scrum process [1].

F3-Decision. When the implementation of a feature was close to completion, the Chief PO proposed F3-decision in the development steering group, which meant that the Chief PO gave a commitment for the completion date of the feature. If the development steering group agreed to the commitment, a F3-decision was made by the group, which meant that marketing of the feature could begin. Otherwise the feature needed to be further developed before the F3-decision.

F4-Decision. When a feature was implemented, tested and integrated into the product, the Chief PO proposed a F4-decision in the development steering group. The steering group could then make a F4-decision which meant that the feature could be included in the next (or a later) public release of the product.

5.2 Challenges

Overcommitment Caused by External Pressure. According to the interviews, the product management still worked in the "old world" way. They requested long-term feature development plans from the PO team, which were not available in the new release planning process, and pressurised them to give premature feature commitments when the release date was approaching. This caused overcommitment by the development organization and decreased the flexibility of the development.

> *...perhaps the product management is not in the new way of working, it easily goes with the old model that we plan one big release ...it feels like we plan a big future release and see what can fit in it.* – A proxy product owner

The case organization tried to mitigate this issue in two ways. First, they tried to improve the predictability of the development by increasing the detail level of FCS's, and by increasing the amount of slack in effort estimates. Second, they created the concept of a minimum marketable feature, which was the set of functionality they could commit to delivering very probably by the next release.

Managing Non-feature Specific Work. In the previous, plan-driven development model the responsibilities of the project management were clearly defined. In the new model, the PO team assumed the responsibility of feature definition and management, but it was unclear who took care of the other project management tasks. These included the handling of the system planning, non feature specific problem reports, system documentation, and external change requests. The PO team had started to have regular meetings where they addressed such issues although it was contrary to the their originally planned responsibilities.

> *Every week we notice things that are not taken care by anybody, that somebody took care of when we had the project organization. ... For example the product documentation that is not directly related to any feature.*
> – A proxy product owner

An additional problem was the prioritization of system improvement work. All features, epics and user stories originated from the product backlog. The developers had difficulties getting system improvement included into the backlog, and if they got it in, they had difficulties getting it included in a development sprint. They also had difficulties finding time to perform system improvement work, as implementing features was implicitly prioritized higher. The development organization attempted to mitigate this issue by making each team take in at least one system improvement user story every sprint.

> *... it is very difficult to participate in things affecting the whole organization or the testing of the whole product, because I have the sprint backlog and I have to get the sprint done first and then if there is time I can perform those things.*
> – A developer

Balancing between Development Efficiency and Building Generalist Teams. Initially the goal of the development organization was to create cross-functional generalist teams that could implement features in all components of the software. However, they quickly realized that many components were technically very difficult and required years of experience to completely understand. This had, in several occasions, caused very long lead times (one to half a year) before a team could implement anything useful in a component. Thus, the portfolio steering group had started mostly assigning features to teams that had the best pre-existing competency in the affected components. Balancing between the development efficiency and building generalist teams was seen difficult especially near the release date when the pressure to get features completed was mounting.

> *... building the competencies has been one of the biggest challenges. ... we have very difficult products where the transfer [of knowledge] is very challenging, it cannot be done in a couple of sprints, it requires several months, in practice. We've had to yield in that, we had to give it to the best [team] ...*
> – A scrum master

5.3 Benefits

Increased Flexibility and Decreased Development Lead Time. In the previous model the release planning was conducted during the first six moths of the two-year project. The changes that could be made to the release after the first six months were typically very small, as they needed to pass thorough a tardy change management process. In the worst case, a feature had over three year lead time from a customer request to a public release.

> *. . . [previously] releasing one package took 18-24 months. In the beginning we performed this system planning which took maybe half an year. And if you did not get the right contents in the release during the first half a year . . . it was immensely difficult to get any changes into the project. . . . If some essential functionality was missing from it, we missed the train, I had to wait to the end of the current project and then the two years after [that]. Which was a very long time* – The product manager

The new process was seen as improvement to the flexibility of development. The new release planning process allowed making changes to the contents of the release on a relative short notice. The feature development schedule was no more tied to the release schedule, which immensely decreased the lead time of the feature development.

> *Now it is like, okay, let's add it to the list. And no worries about where we are going with the change. It's there and in a way nothing was changed even though a new thing was added to the list. I think it is a really good improvement. The flexibility is on another level.* – The product manager

Eliminating Waste in the Planning Process. The general concept in the process was that in the F0-F2 steps the sunk costs would be relatively small, and thus early identification of too expensive or infeasible features would save development resources. In addition, by employing the minimum marketable feature concept, the case organization was able to concentrate on developing the most important parts of the features.

> *What is good in it [the F-decision process] is that . . . it in a way divides the decision making, which is a good thing. We can cut it [the feature] at any point, . . . if we see that the feature passes the time window or otherwise. It gives structure to the decision making and enables us to make smaller decisions and in that way separate the feature decision from the release decision.* – A manager

Increased Developer Motivation. The developers were included in the feature planning starting from the early phases, which allowed them to contribute to the planning and gave them the visibility to the big-picture of the feature. This increased the motivation of the developers.

> *. . . one of the product management involved in the [feature] travelled here to [Hungary] and had a one-day workshop. Why [the feature] is needed for the customer, what information they get, what kind of reasoning [is] behind this feature. . . . It was a motivation boost for the team, to see that what they are doing is really, means something for the [customers].* – A developer

6 Discussion

6.1 RQ1: What Was the Release Planning Process?

Before the agile transformation, the releases were planned by project managers as traditional projects with set resources, schedule and goals. In the new agile process, the release planning was a continuous process where features were initiated based on the availability of resources and the priority of the feature. The release planning process was characterized by regular scoping and prioritization decisions and incremental elaboration of the features before the implementation. The release planning was a collaborative action and the developers took part in the feature planning in early phases of the feature elaboration. Our results support Benestad's and Hannay's observations on release planning [15]. In contrast, most of the proposed models for software release planning treat release planning as an activity that is either performed at the beginning of the release project or over lengthy iterations during the release project and conducted by a few authoritative decision makers in isolation [12,14]. The structure of the development organization was similar to the structure proposed by Leffingwell [6]. However, Leffingwell proposes that the tentative contents of each release should be planned on the user story level in the beginning of each release project [6].

6.2 RQ2: What Were the Challenges Related to the Release Planning Process?

The product management expected precise long-term plans from the development organization. After the transformation, such plans were not created. Many developers would have preferred detailed implementation plans, but such plans were not created any more. Both issues are symptoms of friction between the previous plan driven process and the new agile process. The longing for detailed implementation plans will likely disappear as the developers become more experienced in planning. The conflict between an agile development organization and a plan-driven product management is a recognized challenge in large-scale agile development [2,20]. By employing the minimum marketable feature concept, the case organization was able to provide relatively reliable long term plans without sacrificing flexibility. Although the minimum marketable feature concept is not new in the software development management literature [6,21], it was employed in the case in a novel way to combine long-term planning with flexibility.

There were many tasks which the development teams did not have the competency to perform, for example system level documentation and system level technical planning. In addition, the development of some components required extensive experience and specialized skills. Identifying who should perform such work was a challenge in the case. Initially, Scrum guidance emphasised true cross-functional teams [1]. Several authors of later normative agile development guidance have taken the stance that in large, complex systems there is a place for limited specialization both on the system level and the team level [6,9,22], and our results support this notion.

6.3 RQ3: What Were the Benefits of the Continuous Release Planning Process?

The biggest benefit from the new release planning process was the drastically decreased feature development lead time. The lead time decreased, approximately, from a minimum of two years to a minimum of three months. The short lead time was enabled by the continuous nature of the release planning process and by the flexibility of the Scrum development organization. The short lead time increased the responsiveness of the case organization to customer requests, which created a clear competitive edge [23].

Another benefit was the reduced planning waste. According to the product development queue theory [23], unnecessary inventory is a form of waste that should be eliminated. Compared with the previous, plan driven-process, the incremental elaboration of features in the early phases of the release planning process drastically decreased the inventory of plans and technical specifications waiting on a shelf to be implemented.

The results indicated that software developer's motivation increased because they were included in the decision making and given understanding of the big picture. The existing research on developer motivation [24] supports this result.

6.4 Generalizability and Threats to Validity

In the discussion about the validity of this research, we rely on the definitions of validity and reliability proposed by Yin [18]. Internal validity is not relevant, as this research was neither explanatory nor causal [18]. The main threat to the construct validity of this research was the accuracy of the descriptions. To increase the construct validity, we interviewed multiple persons for each role in the case organization, if possible. The interviews were coded and analysed by the first author. To increase the construct validity, the second and the third author reviewed the analysis. Furthermore, the fourth author of this article was one of our key informants and he also reviewed the analysis.

The external validity of a case study concerns the domain to which the results can be generalized [18]. Based on our study, we can create a hypothesis of the significant characteristics of the domain. First, the system under development was large, multifaceted and technically demanding. Second, the product and release management organizations worked in a plan-driven way and were separate from the development organization. Third, the number of development teams was relatively large. Fourth, the development was distributed on two sites. The results are likely generalizable to single site development, but it difficult to hypothesize how generalizable the results are when the development is distributed on three or more sites.

The main threat to the reliability [18] of this research is the variability in the data collection. The data collection was conducted using the general interview guide approach [17], which introduced variability to the topics discussed in the interviews. However, the large number of interviewees and multiple interviewers allowed data source and investigator triangulation [17] which increased the reliability of the results.

7 Conclusions and Further Work

Release planning is a crucial task in market-driven requirements engineering [11]. Large development organizations have increasingly started adopting agile software development methods [2]. The traditional, plan-driven release planning models are not well suited for agile development organizations where scoping decisions must be made constantly and detailed requirements analysis is performed alongside the implementation [14]. If the release planning process does not support the agile development organization, the development organization will not be able to work efficiently towards the high level goals of the company.

Our case study provides a detailed description of how a large-scale Scrum organization in Ericsson performed release planning, and of the challenges and benefits related to the release planning process. The continuous release planning process is characterized by regular scoping and prioritization decisions, and by the incremental elaboration of features. The challenges were the overcommitment caused by external pressure, managing non-feature specific work and balancing between development efficiency and building generalist teams. The benefits were the waste eliminated in the planning process, the increased flexibility and the decreased development lead time. Our study contributes to the growing knowledge base on scaling agile software development methods.

We will continue to study the case organization as the Scrum development organization becomes more mature and all the stakeholders have had time to adjust to the new development and planning processes. Specifically, we are interested in studying how the product management organization is changed to better work with the Scrum development organization. In addition, it would be interesting to study how other large organizations that have adopted agile development methods perform release planning.

Acknowledgment. We would like to thank Oy LM Ericsson Ab for making this study possible and all the anonymous interviewees for providing valuable contributions to this research. We would like to thank the TiViT Cloud Software Finland-program for funding this research.

References

1. Schwaber, K., Beedle, M.: Agile software development with Scrum. Prentice-Hall, Upper Saddle River (2002)
2. VersionOne, Inc.: 6th Annual "State of Agile Development" Survey (2011)
3. Cockburn, A.: Agile software development. Addison-Wesley, Boston (2002)
4. Rautiainen, K., Lassenius, C., Sulonen, R.: 4CC: A framework for managing software product development. Eng. Manag. J. 14(2), 27–32 (2002)
5. Cohn, M.: Agile estimating and planning. Prentice Hall Professional Technical Reference, Upper Saddle River (2005)
6. Leffingwell, D.: Agile software requirements: lean requirements practices for teams, programs, and the enterprise. Addison-Wesley, Upper Saddle River (2011)

7. Ruhe, G., Saliu, M.O.: The art and science of software release planning. IEEE Softw. 22(6), 47–53 (2005)
8. Schwaber, K.: The enterprise and scrum. Microsoft Press, Redmond (2007)
9. Augustine, S.: Managing Agile Projects. Prentice Hall Professional Technical Reference, Upper Saddle River (2008)
10. Chow, T., Cao, D.B.: A survey study of critical success factors in agile software projects. J. Syst. Softw. 81(6), 961–971 (2008)
11. Fogelström, N.D., Gorschek, T., Svahnberg, M., Olsson, P.: The impact of agile principles on market-driven software product development. J. Softw. Maint. Evol.-R. 22(1), 53–80 (2010)
12. Svahnberg, M., Gorschek, T., Feldt, R., Torkar, R., Saleem, S.B., Shafique, M.U.: A systematic review on strategic release planning models. Inform. Softw. Tech. 52(3), 237–248 (2010)
13. Carlshamre, P.: Release planning in market-driven software product development: Provoking an understanding. Requir. Eng. 7(3), 139–151 (2002)
14. Jantunen, S., Lehtola, L., Gause, D.C., Dumdum, U.R., Barnes, R.J.: The challenge of release planning. In: Proceedings of the Fifth International Workshop on Software Product Management, pp. 36–45 (2011)
15. Benestad, H.C., Hannay, J.E.: A comparison of model-based and judgment-based release planning in incremental software projects. In: Proceeding of the 33rd International Conference on Software Engineering, pp. 766–775. ACM, New York (2011)
16. Heikkilä, V., Rautiainen, K., Jansen, S.: A revelatory case study on scaling agile release planning. In: Proceedings of the 36th Euromicro Conference on Software Engineering and Advanced Applications, pp. 289–296. IEEE Computer Society (2010)
17. Patton, M.Q.: Qualitative research and evaluation methods, 3rd edn. Sage Publications, Thousand Oaks (2002)
18. Yin, R.K.: Case study research: design and methods, 4th edn. Sage Publications, Thousand Oaks (2009)
19. Adolph, S., Hall, W., Kruchten, P.: Using grounded theory to study the experience of software development. Empir. Softw. Eng. (2011)
20. Lyon, R., Evans, M.: Scaling up pushing scrum out of its comfort zone. In: Proceedings of the Agile 2008 Conference, pp. 395–400 (2008)
21. Denne, M., Cleland-Huang, J.: The incremental funding method: Data-driven software development. IEEE Softw. 21(3), 39–47 (2004)
22. Larman, C., Vodde, B.: Practices for scaling lean & agile development: large, multisite, and offshore product development with large-scale scrum. Addison-Wesley, Upper Saddle River (2010)
23. Reinertsen, D.G.: Principles of product development flow: second generation lean product development. Celeritas Publishing, Redondo Beach (2009)
24. Beecham, S., Baddoo, N., Hall, T., Robinson, H., Sharp, H.: Motivation in software engineering: A systematic literature review. Inform. Softw. Tech. 50(9-10), 860–878 (2008)