Information and Software Technology 57 (2015) 116–140

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Operational release planning in large-scale Scrum with multiple stakeholders – A longitudinal case study at F-Secure Corporation

Ville T. Heikkilä [a,*], Maria Paasivaara [a], Kristian Rautiainen [a], Casper Lassenius [a], Towo Toivola [b], Janne Järvinen [b]

[a] Department of Computer Science and Engineering, Aalto University, PO Box 15400, FI-00076 Aalto, Finland
[b] F-Secure Oyj, PO Box 24, FI-00181 Helsinki, Finland

ARTICLE INFO

Article history:
Received 3 September 2014
Accepted 14 September 2014
Available online 22 September 2014

Keywords:
Agile software development
Scrum
Large projects
Release planning
Software project management

ABSTRACT

Context: The analysis and selection of requirements are important parts of any release planning process. Previous studies on release planning have focused on plan-driven optimization models. Unfortunately, solving the release planning problem mechanistically is difficult in an agile development context.
Objective: We describe how a release planning method was employed in two case projects in F-Secure, a large Finnish software company. We identify the benefits which the projects gained from the method, and analyze challenges in the cases and improvements made to the method during the case projects.
Method: We observed five release planning events and four retrospectives and we conducted surveys in the first two events. We conducted six post-project interviews. We conjoined the observation notes, survey results and interviews and analyzed them qualitatively and quantitatively.
Results: The focal point of the method was release planning events where the whole project organization gathered to plan the next release. The planning was conducted by the development teams in close collaboration with each other and with the other stakeholders. We identified ten benefits which included improved communication, transparency, dependency management and decision making. We identified nine challenges which included the lacking preparation and prioritization of requirements, unrealistic schedules, insufficient architectural planning and lacking agile mindset. The biggest improvements to the method were the introduction of frequent status checks and a big visible planning status board.
Conclusion: The release planning method ameliorated many difficult characteristics of the release planning problem but its efficiency was negatively affected by the performing organization that was in transition from a plan-driven to an agile development mindset. Even in this case the benefits clearly outweighed the challenges and the method enabled the early identification of the issues in the project.

1. Introduction

Planning the next product release is recognized to be a challenging part of market-driven product development [1] and an important success factor in agile software development projects [2]. The main goal of release planning is to select an appropriate scope for a release while taking into account constraints such as the budget, resources, dependencies and technical aspects, and factors such as the importance or urgency of the candidate requirements [1,3,4]. Release planning is especially vital in market-driven product development, as it puts into practice the strategy of the company [5].

Conceptually, software product release planning is performed on two levels [5,6]. On the strategic level, the focus is on selecting the appropriate requirements for the next public release of the product. On the operational level, the focus is on planning how the requirements for the next release can be best implemented [7]. Strategic release planning activities are sometimes called pre-project activities [1], indicating that requirements gathering, prioritization and planning are performed before the development begins. In agile software development projects, strategic and operational level planning activities are not strictly separated [1]. Several strategic release planning activities, such as the market, customer and competitor analysis, are typically performed by the product managers or Product Owners. However, most of the release planning activities on both levels, such as the prioritization of requirements, implementation scheduling and cost/benefit

* Corresponding author. Tel.: +358 50 526 8483.
E-mail addresses: ville.t.heikkila@aalto.fi (V.T. Heikkilä), maria.paasivaara@aalto.fi (M. Paasivaara), kristian.rautiainen@aalto.fi (K. Rautiainen), casper.lassenius@aalto.fi (C. Lassenius), towo.toivola@f-secure.com (T. Toivola), janne.jarvinen@f-secure.com (J. Järvinen).

http://dx.doi.org/10.1016/j.infsof.2014.09.005
0950-5849/

analysis, are performed collaboratively with the development organization [8,9].

Scrum is an iterative and incremental agile development method [10], meaning that work is planned one development iteration, or a sprint, at a time. Scrum was originally created for small co-located teams, and it emphasizes direct and informal communication between the team members, which limits the maximum size of a single development team [8]. The Scrum approach to release planning is very different from the approach used by traditional, plan-driven software development life-cycle models [11]. Instead of employing project plans that are based on a set of predefined factors and constraints, Scrum relies on collaborative human judgement and informal negotiations [8,9].

The majority of published research on software release planning focuses on different kinds of mathematical models and simulations which are designed to create the most valuable, satisfying or risk-free release plans when the candidate requirements can be described in sufficient detail, and key constraints and factors are known and can be estimated sufficiently accurately [5]. The model or simulation is then used to generate one or a set of optimal or near-optimal release plan(s). However, the mathematical approach to release planning has proven to be problematic in practice, since the above conditions are mostly difficult to fulfill in practice [4,9,12–14].

According to the 2013 State of Agile survey [15], the Scrum software development method has reached an established status in the software development community, and many large software development organizations have adopted or aspire to adopt agile methods. Freudenberg and Sharp [16] gathered participants' opinions on what were the most important questions or issues related to agile software development during the XP2010 conference. In their results, "Agile and large projects" was voted as the most important issue. Although neither of these surveys was, strictly speaking, methodologically robust, they both suggest that research on scaling Scrum and Scrum-based software development methods are relevant to the industry.

Furthermore, the Scrum approach to planning focuses on the single iteration level for a single team. Published empirical research on successful practices for adopting agile methods in large-scale software development organizations is still scarce [17], despite the fact that practitioner literature with normative but empirically weakly supported advice exists [18,19].

Considering the importance of release planning for the success of a development project [1,2], the dominance of the model-driven release planning in research [5,13,20], and the lack of solid empirical evidence of successful practices for scaling up Scrum release planning, there is an obvious gap in the research of release planning in large-scale agile software development organizations.

In this paper we work towards filling this gap by presenting two examples of the use of a release planning method in a large Scrum organization. We describe the release planning method used, as well as the benefits and challenges of its use. The release planning method consisted of *Release Iteration Planning events*, during which release planning was performed, and of the preparations conducted for those events. Consequently, we call the method the *Release Iteration Planning method*. The method has been briefly described in the practitioner literature [21]. We concentrate on the collaborative release planning performed in the Release Iteration Planning events. The market facing tasks of product management, such as the market, competitor and customer analysis, are out of the scope of our study. Although these knowledge areas are important in any market-driven product development project, there is already a myriad of literature that provides insight and instructions for these activities (see e.g. [22,23]).

In previous work, we described the Release Iteration Planning method [24]. This paper considerably expands that work by including results from three additional Release Iteration Planning events and from six post-project interviews, as well as describing and discussing the benefits and challenges the case organization experienced when applying the method. Additionally, members of the case organization have published an account of the release planning in the project from an insider point of view [25]. Compared with their account, this paper has a similar topic, but it is considerably broader in scope and contains considerably more detailed description and in-depth analysis of the method and the case projects.

The main contribution of this paper is that, to our knowledge, it presents the first empirical and longitudinal in-depth study of the use of the Release Iteration Planning method. We describe how the method was applied to a real large-scale agile organization, how the organization modified the method over the time, and what the benefits and challenges of applying this method were.

The paper is structured as follows: Section 2 provides an overview of the related work on software release planning, Section 3 describes the research goals and methods, Section 4 presents the case organization and case project backgrounds, Sections 5 and 6 describe how the Release Iteration Planning method was applied in the case projects, Section 7 analyzes the benefits of the method, while Section 8 discusses the challenges faced in applying the method, Section 9 discusses our findings and their validity, and finally Section 10 presents the conclusion and directions for future work.

2. Software release planning

In this section, we review the literature related to software release planning. First, we describe empirical research on the characteristics of release planning and on the reasons why software release planning is difficult. Then, we describe existing research on model-based release planning, the most popular research subject at the time of writing. Finally, we look at the literature on release planning in multi-team Scrum development organizations.

2.1. Characteristics of the release planning problem

According to empirical release planning research, there are several technical and human factors that make release planning a difficult task. A shared understanding of the requirements arises during development and may be weak in the beginning of the release development project [9]. The requirements selection criteria are time dependent and may change both qualitatively and quantitatively during the release development project [9]. The great majority of requirements have dependencies between them that constrain the implementation order. These dependencies may be difficult to identify and complex [9,26]. Decision makers have difficulties expressing how value is created by selecting and prioritizing requirements [27] and gut-feeling, lobbying, politics, sell-in and strong individuals affect the requirements prioritization in practice [28]. The size of the client base affects how the release planning of a product can be performed, and some customers are strategically more important than others [27]. The business perspective is often considered the most important requirements selection factor, which results in implicit prioritization of feature development over system improvement and innovation [27–29].

2.2. Model-based release planning

Software release planning has been widely accepted to be a difficult problem [3,4,13]. Mathematical release optimization models treat release planning as an mathematical optimization problem [4,5]. Such models select requirements to be included in one or

more subsequent releases based on different kinds of factors that are related to the requirements, releases or stakeholders. The goal is to optimize an utility function which is typically defined as a system of equations. The optimization algorithm must also take into account any constraints that are in effect, which are typically defined as inequalities. The inequalities and equations contain variables that reflect the different factors of the planning problem. The utility function is expected to represent the overall planning objective of the software releases. A mathematical algorithm or simulation is then employed to solve the optimization problem described by the inequalities and equations. The models produce one or more release plans which are optimal or near-optimal given the constraints which are in effect.

The model-based planning research proposes logically compelling models to which decision making should conform. The models gather the properties of the requirements in isolation one by one and put the decision making into the hands of a few authoritative stakeholders [7,30–35]. On the operational level, release planning research has concentrated on models for optimizing the assignment the development tasks to the developers in relation to a set goal, for example to maximize the utilization of development resources or to minimize the overall development time [6,7,9,13,36].

The model-driven (or plan-driven) approach to release planning is normative, linear and authoritative and it requires the taming of the release planning problem into a problem which can be algorithmically solved [13]. This approach has resulted either in models which are too simple to be useful in practice, or models which are so complex that the practitioners find it difficult to provide the necessary input values and find it hard to trust the output, as they cannot understand the process that created it [4,9,13].

2.3. Release planning in multi-team Scrum development organizations

Small group research has found that groups of 3–6 members are more productive than larger groups and reach high productivity faster [37], and that software development teams of nine or more members are less productive than smaller teams [38]. The proposed way to scale up the size of a Scrum development organization is to employ multiple small Scrum teams which simultaneously develop the same software system [39]. There are usually architectural complexities which result in a network of dependencies between requirements [4,9], making coordination between the independent Scrum teams difficult. The early Scrum literature provided little guidance for strategic release planning, as the focus was on planning and developing software one sprint at the time in a single team, single project context [8]. However, large development organizations have adopted Scrum practices [15]. In large, market-driven software development organizations the existence of release plans is the norm, although they are often embedded in the product and project plans. In theory, strategic plans are agnostic towards the implementation of the requirements, and by extension, towards the development process employed [5]. Thus, the adoption of Scrum does not need to affect the strategic planning process of the company. In a Scrum project, the release plan provides information on how the goals of the development project are reached over a multi-sprint time horizon [40]. The release plan unifies the expectations about the likely outcome and timeframe of the next release [40].

Existing empirical research on release planning methods in large-scale agile development organizations is scarce [41]. In addition to earlier research [24,25] on the Release Iteration Planning method described in this paper, to our knowledge there is only one other publication on the topic. It describes how Ericsson performs continuous release planning in a large, multi-team agile development organization [42]. Vlaanderen et al. [43] propose an extension to Scrum which applies Scrum principles to software product management. Their model does not explicitly address release planning, but product managers are expected to provide the developers well refined requirements that the developers are expected to implement in during following development sprints. Thus, the selection of requirements that are to be refined implicitly affects the contents of the next release.

The existing prescriptive guidance for organizing development and for release planning in a multi-team Scrum development environment has been written by practitioners and consultants based on their personal experiences. A few notable examples are the books written by Schwaber [39], by Larman and Vodde [44] and by Leffingwell [18,21]. The approaches prescribed in these books differ notably on the organization of development and on the process model.

Schwaber [39] suggest organizing the development using a tree structure of multiple levels of integration Scrum teams in the branch nodes and (development) Scrum teams in the leaf nodes. The integration Scrum teams do not develop functional software, but instead integrate, build and test the software implemented by the (development) Scrum teams. Both kinds of Scrum teams have a dedicated Product Owner. All requirements are listed in a product backlog as user stories. The branch node Product Owners are responsible for assigning sections of the product backlog for the lower level teams. Release planning is performed by the root node Product Owner by selecting a subset of the product backlog as the release product backlog.

Larman and Vodde [44] propose a two-layer model for a large-scale, agile development organization. Development teams are arranged as feature teams that work on a single feature at a time. Feature teams are grouped into technical product areas. Each product area is managed by an area Product Owner, who in turn is managed by a Product Owner. The Product Owner manages the product backlog and assigns backlog items to the product areas. Features are large backlog items that describe functionality that is valuable for the customer. Features are split into smaller backlog items which can be implemented during a single sprint. The dates of releases are planned by the Product Owner and the contents of each release are defined by what is ready by the time of the release.

Our case organization used an early version of Leffingwell's release planning method which was further refined and published in his book in 2011 [21]. The model is based on Leffingwell's experiences as a practitioner and consultant in several software development organizations of different sizes. Among other things, it prescribes a comprehensive model for scheduling, planning and managing releases in a large enterprise. The central concept is the release train, in which the internal and external releases of the software follow each other like the cars in a train. The central principles of the train are frequent timeboxed releases and release planning events, global milestones, continuous integration, synchronized development iterations, and hardening (or finalization) iterations.

In this approach, release planning is performed in release planning events where all stakeholders of the product assemble to plan the next release together. Leffingwell proposes roughly the following agenda for the release planning events: opening, introduction and guidance presentations, team planning breakouts, plan reviews and status checks, final plan review, risk and impediment review, and retrospective. The goal is to create a tentative implementation plan of the next internal or public release on the user story level [21].

On a more general level, van Waardenburg and van Vliet [11] identified challenges in cases where agile methods co-existed with a traditional, plan-driven enterprise. They studied two companies that had adopted agile methods and stabilized their software development processes. Most of these challenges and the strategies for mitigating them were related to planning and requirements engineering. They found that concurrent development

streams, separated layer development and different process approaches increased the IT landscape complexity. The high IT landscape complexity then caused problems with communication, dependent definition of done and difficulties creating change. They identified the following strategies for mitigating the high IT landscape complexity: stimulating a common sense of purpose, managing programme level alignment, combining product backlogs, end-to-end representation in team and facilitating project management. They also found that centralized IT department and traditional project organization caused lack of business involvement. The lack of business involvement caused problems with requirements gathering, slow reaction to change, problems with requirements prioritization and limited feedback from business. The mitigation strategies for lack of business involvement were the following: changing business' mindset, channelling business knowledge through the Product Owner and managing business-level alignment with intense stakeholder communication.

3. Research method

3.1. Research objective and questions

The overall objective of the research presented in this paper is to start filling the large-scale agile planning research gap by

describing the Release Iteration Planning method adopted in a large organization that employed the Scrum software development method in two multi-team development projects.

We aim to reach the research objective by answering the following three research questions:

RQ1: How did the case projects adopt the Release Iteration Planning method in practice?
RQ2: What kind of benefits did the case projects gain from adopting the Release Iteration Planning method?
RQ3: What kind of challenges did the case projects face in adopting the Release Iteration Planning method?

We focus on the release planning that was conducted in the Release Iteration Planning events. The market facing tasks of product and project management are out of the scope of our study. These knowledge areas include, but are not limited to, market analysis, competitor analysis, product strategy, roadmapping, product pricing, recruitment and project budgeting (see e.g. [22,23,45]).

3.2. Case study method

We conducted the research as a longitudinal multiple case study [46]. According to Yin, the case study method is most appropriate when the subject of the study is contemporary, situated in a real-world context and the researcher has little or no control of the events [46]. The case organization was purposefully selected, as it provided an opportunity to perform an information-rich longitudinal study [46,47]. In addition, the first of the studied development projects was the largest in the company history (in terms of the number of developers), which made the project a prime candidate for an information-rich, or a revelatory, case study [46]. Table 1 shows an overview of the data sources and the quantity and type of data from each data source in the two studied projects. Details of the data collection are described in the two following sections.

3.3. Project α data collection

The researchers observed five Release Iteration Planning events which took place over a nine-month time period between December 2009 and September 2010. The Release Iteration Planning events were part of a project (hereafter called Project α) for developing a new version of a software product. The Release Iteration Planning events are hereafter referred to as Events 1–5.

Table 1
Overview of the data collection.

Data source	Data collected	Project
Planning events		
Event 1	Voice recordings (13 h 19 min), notes	α
Event 2	Voice recordings (7 h 34 min), notes	α
Event 3	Voice recordings (5 h 28 min), notes	α
Event 4	Voice recordings (5 h 5 min), notes	α
Event 5	Voice recordings (3 h 41 min), notes	α
Retrospectives		
Retrospective R_{ω}	Voice recording (1 h)	α
Retrospective $R_{\alpha 1}$	Voice recordings (56 min)	α
Retrospective $R_{\alpha 2}$	Voice recordings (4 h 45 min)	α
Retrospective R_{post}	Field notes	α
Surveys		
Event 1 survey	33 Responses (response rate ≈ 33%)	α
Event 2 survey	26 Responses (response rate ≈ 19%)	α
Post-project interviews		
Product Owner	Voice recording (1 h 22 min)	α, β
Scrum Master/Facilitator	Voice recording (1 h)	α, β
SPI Manager	Voice recording (1 h)	α, β
Product Manager A	Voice recording (41 min)	α
Product Manager B	Voice recording (58 min)	β
R&D Line Manager	Voice recording (48 min)	α

Data was collected by three researchers. Data collection was performed using multiple methods. We used several data sources to allow for data triangulation and multiple researchers for investigator triangulation to increase the reliability and construct validity of the results [48,47,46]. During the events, we attempted to identify and observe the most interesting or remarkable discussions and happenings. We observed how the different development teams worked and interacted with the participants. We voice-recorded the plan reviews, status checks and interesting discussions during the events. During the planning events, we acted as neutral observers and did not affect proceedings.

We used two data collection methods in the Release Iteration Planning events. First, we used voice recorders to record the presentations and informal dialogue during the events. Second, we took field notes during the planning events. Immediately after each event, we compared notes and composed an entry into a case diary. Each entry contained an abridged description of the observations and an initial analysis of the observations. We provided feedback reports for the organization after Event 1 (see Section 5.4.6) and Events 2 and 5 (see Section 5.5.10). These reports were based on the case diary. They described the biggest issues we had identified in the events and improvement suggestions, and in the case of Event 2 and 5, observations of improvements from the previous events.

A survey was conducted after Event 1 and Event 2 to gather opinions on the planning method from the participants. The survey conducted after Event 1 contained questionnaire statements on a six-point Likert-like scale, open questions, and a question for grading the event. Event 2 survey contained a subset of the statements from the first survey, selected by the case organization representatives and by the researchers based on the topics seen as the most important for the success of the planning. The surveys were anonymous to increase the reliability of the results. All participants of the events were invited to respond to the surveys.

In addition to the planning events, we participated in four retrospectives. A retrospective (Retrospective R_{ω}) of a project previous to Project α was conducted in June 2009. The discussions during the retrospective were recorded with a voice recorder.

The goal was to identify issues regarding requirements management.

In Event 2 the introduction and vision presentations were followed by a retrospective (Retrospective $R_{\alpha 1}$). The purpose of the retrospective was to find and solve impediments and learn from the good and bad practices observed in the project so far. The retrospective was performed in role-based groups. The groups were given 1.5 h to conduct the retrospective. Each group presented their briefing to the other groups at the end of the retrospective. The briefings were voice recorded.

A retrospective of the first release iteration (Retrospective $R_{\alpha 2}$) was conducted the day before Event 3. The goal was to identify and solve issues in the software development and development management processes in the project. Eleven topics were selected for discussion by the participants. There were three discussion sessions which lasted for 45 min each. The participants split into groups which each discussed a single topic during the discussion session. Between the sessions the topics were changed and the participants selected new groups. There was a 45-min review of the results after the third session. Two researchers observed the retrospective and used two voice recorders to record several discussion sessions and the final review of the results. Overall, this retrospective lasted for three hours.

A retrospective (Retrospective R_{post}) of Project α was conducted after Event 5 in December 2010. The purpose of the retrospective was to learn from the issues identified in the project so far. Due to confidentiality issues, the researcher and case organization representatives agreed that the retrospective would not be recorded. Instead, the researcher took detailed field notes. The retrospective lasted for four hours.

Project α officially ended in October 2011 and in August 2012, the first and third author conducted five post-project interviews in the case organization to gather more data on the Release Iteration Planning method, and to gather data on the developments in Project α after Event 5 and regarding the planning process after the case study period.

The interviewees of the post-project interviews were purposefully selected. The goal was to interview the most information rich informants, and the selection was performed together with a case company representative. The interviewees had the following roles: Product Owner, Scrum Master, Software Process Improvement (SPI) Manager, Product Manager and R&D Line Manager. The interviews were conducted in a semi-structured fashion. The list of interview questions was based on the analysis of the previously collected data. Specifically, the questions covered the history of the project preceding Project α, the motivation for adopting the Release Iteration Planning method in Project α, the changes in the project organization during Project α, the developments in Project α after Event 5, and the evaluation of the Release Iteration Planning method including both challenges and benefits of the method. The full list of interview questions can be found in Appendix A.

Fig. 1 shows the timeline of the data collection. The segments above the timeline show the approximate time span of each release iteration.

3.4. Project β data collection

During the post-project interviews on Project α, we were informed that another project, Project β, had adopted the Release Iteration Planning method. In effort to increase the validity of our results, we decided to include Project β in the scope of our study. The interviews for collecting data of Project β were conducted alongside the interviews about Project α in August 2012. A total of three persons were interviewed concerning Project β. Two of the interviewees were also interviewed about Project α. These were the Software Process Improvement (SPI) Manager and the Scrum Master. The Scrum Master had worked as a Scrum Master in Project α and facilitated the Release Iteration Planning events in Project β (thus he will be called the Facilitator regarding Project β) and the SPI Manager had participated in the introduction of the method to Project β. The third interviewee was the Product Manager responsible for the product that was developed in Project β.

3.5. Data analysis

All the post-project interviews and the recordings from the retrospectives were transcribed in their entirety. The transcribed retrospectives and the transcribed interviews were imported into the qualitative analysis program Atlas.ti for analysis. The transcribed materials were then coded based on a concept list which was created based on the analysis of the data collected during the case study. The concepts included the following stakeholders that were identified during the data collection: architects, development teams, Product Managers, Product Owners, project steering group, Scrum Masters. Due to the novelty of the method in the case company in the first Release Iteration Planning event, the descriptions of the first event and the later events were coded separately with the following concepts: Event 1 description, Event 2–5 description. The following two codes were used to extract opinions towards the method: good in the release planning method, problems in the release planning method. During the analysis of the data, the following additional concepts, that gave further insights into the data, were identified: after the observation period, collaboration between the Product Management and the R&D, dependencies, release planning in Project β, why the method was adopted.

The backgrounds of the case organization and projects were written based on publicly available data on the case organization, based on observation notes and recordings from the presentations in the Release Iteration Planning events, and based on the interview data. The transcribed interviews and retrospectives together with the field notes, feedback reports, survey results and the case diary, were employed to create the description of Project α. After we had constructed the general description of the case, we reviewed the description and identified topics that were especially interesting or confusing. The codes or combinations of codes were used to extract passages related to those topics and the passages were reread to elaborate the topic in the case descriptions. When necessary, the voice recordings from the Release Iteration Planning events were listened to in order to extract details which were not available in the other data sources.

Since the observation period ended in December 2010, the descriptions of what had happened afterward are solely based on the quotations on "after the observation period" from the interviews. The description of Project β was based solely on the full transcriptions of the three interviews.

The benefits of the method and challenges in the adoption of the method were identified based on the quotations on "good in the release planning method" and "problems in the release planning method" from the transcribed interviews and retrospectives, and based on the field notes, survey results, feedback reports and the case diary. Quotations related to benefits and challenges were extracted from all transcribed interviews and retrospective recordings and read in their entirety. Triangulation [46–48] was done by comparing the data collected from different sources, by different researchers and with different methods in order to identify any divergence between the data sources, researchers and methods. Any divergent points of data were then analyzed further to find out what the cause of the divergent data points was and if it affected the validity and reliability of our results.

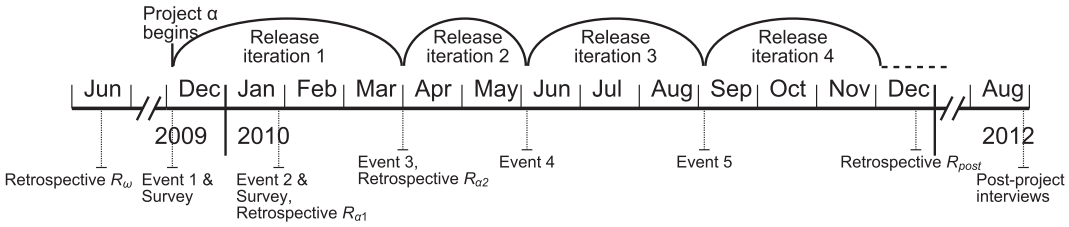

Fig. 1. Timeline of the research in Project α.

4. The case organization and project backgrounds

In this section, we describe the case organization, followed by a more detailed background of the two projects we studied.

4.1. Case organization background

F-Secure Corporation was established in 1988 under the name Data Fellows. The company published its first computer security program in 1991. In 1999, Data Fellows changed its name to F-Secure Corporation and was listed on the Helsinki Stock Exchange. In 2009, the corporation employed over 800 employees and had subsidiaries around the world. In addition to traditional PC computer security software, the offerings included mobile security and data security software.

The adoption of the Scrum development method begun in 2006. The goal was to improve the efficiency and decrease the lead-time of software development. When Project α begun in 2009, the different parts of the organization were still in different stages of adopting Scrum. Some parts had employed Scrum for several years and some parts were still working using traditional, plan-driven processes.

4.2. Project α background

The first version of the software product developed in Project α was published in 2003. A new version of the product had been published yearly since then. Previously to Project α, the development organization of the software product had been divided into the three following areas: the front-end, the back-end and the engine. This division existed because of the different technologies employed in the different parts of the product and due to legacy reasons. The developer level communication between the different areas had been minimal and dependencies had been managed using well-defined APIs. In the front-end area the developers had been organized into 6–7 member Scrum teams. Each team had a high abstraction level component area which they were the most familiar with (for example automatic updates or user notifications). The engine and back-end development had been organized as line-organizations with continuous incremental development processes. The front-end part of the software had been rewritten in the previous project.

While the overall goal of Project α was the development of the new version of the software, one of the subgoals was to rewrite the back-end using a new architecture. To accommodate the simultaneous rewrite of the back-end and improvement of the front-end, teams from both parts were included in the project organization, while the engine part was kept separate. The existing teams were kept mostly intact at the beginning of the project. The development organization of the project consisted of approximately 140 stakeholders, including 10 software development teams of six to seven members each. One of the teams was from an off-shore site located in Malaysia. During the project three more off-shore teams from the Malaysian site were added as well as one contracted team from Poland.

4.3. Project β background

The Release Iteration Planning method was also employed in an other project of the case company, Project β. At the time of our interviews in August 2012, Project β had successfully applied the method twelve times starting in the summer of 2010. Projects α and β did not share any resources. In Project β, four development teams and two Product Owners were located in Russia, while two Product Managers and the main architect were located in Finland and visited the Russian site once a month. In this project, the Release Iteration Planning events were conducted every other month as face-to-face meetings at the Russian site with all members (the Product Managers, Product Owners and development teams) physically present.

5. Release planning in Project α

In this section, we provide a detailed description of the release planning in Project α. We first provide an overview of the project. Then, we describe the first Release Iteration Planning event (Event 1) in detail. In the subsequent section, we describe the notable changes made to the Release Iteration Planning method over the following events (Events 2–5). After the event descriptions, we describe our feedback reports to the case organization and finally we describe what happened in the project after Event 5.

5.1. Motivation for adopting the Release Iteration Planning method

The motivation for adopting the Release Iteration Planning method was tied to the issues the case organization experienced in the development project previous to Project α (hereafter referred to as Project ω) and also related to the size and complexity of Project α, which had the largest project organization the case company had ever had in a single project. A retrospective of Project ω was held in June 2009 (Retrospective R_ω). The two main issues related to the planning and monitoring of Project ω identified in Retrospective R_ω are described below.

The first issue was that the developers were unavailable to support project planning. At the beginning of Project ω, the development teams were still finalizing the project preceding Project ω and had no time to assist the Product Owners and Product Managers in the planning of the project. However, the estimates provided by developers were considered crucial for performing the cost/benefit analysis for the candidate requirements of Project ω.

The second issue was the difficulty of monitoring the progress of software development during the project. While the goal of the managers had been to organize the development in an agile way, the requirements management was still quite waterfall-like. The whole project had been planned on feature level before the development begun. Features had been assigned to the development teams by managers. After a six month development period,

a two-month period to improve quality and finalize the project had been scheduled. The Product Managers expressed that they could get no information on the progress during the first six months and could not inform marketing and sales about upcoming features and improvements, which was considered a problem:

> *And now that we are at the last two months we know what we are getting. We can start making our sales material, we are confident on what we get. But the first six months of the project, I would not dare to talk to any partner or whoever about [Project ω], because we had basically zero confidence on what would be the end product. That only comes when we are really close to the release . . . but through the project the visibility and the confidence is extremely low on what you will be getting.*
>
> [Product Manager, Retrospective R_ω]

The case organization had hired an external agile development consultant to assist in their release planning. The external consultant suggested conducting planning events which he called "joint release planning events", and we refer to as the Release Iteration Planning events. According to the consultant, he had previously facilitated such events in several companies, but the Project α organization was the largest organization he knew that had tried the method. The benefits the organization expected to gain from the Release Iteration Planning method were the following:

1. Better communication between the development organization and Product Management.
2. Better transparency of development progress for Product Management.
3. Better coordination between the development teams, especially between the front-end and back-end teams.
4. Reduced planning overhead and faster planning.

5.2. Overview of the project

The project was initially distributed to two sites: Finland, which was the main site, and Malaysia. During the project, the number of development teams in Finland and Malaysia varied and one additional contracted Polish team was added. Table 2 shows the overall number of development teams at each site during the studied Release Iteration Planning events. The Product Owners and Scrum Masters had responsibilities as prescribed in the Scrum method [39]. However, in the beginning of the project, the Product Owners were not dedicated to the teams. During all planning events, most members of the Finnish development teams were present. The Malaysian and Polish teams were typically represented in the planning events by each team's Scrum Master and the rest of the teams participated remotely via a videoconferencing system. In addition, several other stakeholders participated in the planning events. They will be described in detail in the following sections.

The *development project* was divided into *release iterations* and each release iteration was divided into development *sprints*. Project α was originally expected to last approximately six months, until the end of Release Iteration 2, when the new version of the software was expected to be published. Both release iterations were planned to begin with a Release Iteration Planning event. Before Event 1, the external consultant suggested that an additional Release Iteration Planning event should be conducted in the middle of the first release iteration to adjust the plan. Thus, an additional Release Iteration Planning event (Event 2) was scheduled to be held in the middle of Release Iteration 1.

Release Iteration 1 was expected to end in a test release. Release Iteration 2 was planned to end in a public release of a feature complete version of the software. During Release Iteration 2 the project was first extended by one release iteration and during Release Iteration 3, the project was extended by one more release iteration. Fig. 1 illustrates the overall timeline of Project α and Fig. 2 shows an overview of the schedules of the Release Iteration Planning events in Project α.

Each release iteration consisted of two-week development sprints. The number of sprints in each release iteration varied based on the length of the release iteration. Release Iteration 1 consisted of eight sprints, Release Iteration 2 of four sprints, Release Iteration 3 of six sprints, and Release Iteration 4 of eight sprints. The length of the release iterations varied as the case organization tried to find the optimal number of sprints for a release iteration.

Table 2
Number of development teams at each location during each event in Project α.

Event	Teams per location		
	Finland	Malaysia	Poland[a]
Event 1	9	1	–
Event 2	10	3	–
Event 3	9	2	1
Event 4	8	2	1
Event 5	7	4	1

[a] Contracted team, not a company site.

5.3. Requirements management

The organization had defined a four-level hierarchical model of requirements management in the project. In the organization's model, *epics* formed the high-level goals of the product for the multi-release time horizon. Epics were split into more concrete *features* which described the requirements for the whole release project. Features were expected to encompass functionality that would create concrete value for the customer or user. There was no limit to the size of features except a working definition of "a feature is something that can be implemented in a single release iteration". At the beginning of the project, there were 135 features proposed for implementation. Features in turn were split into *user stories* to be developed in sprints. User stories were expected to describe a small portion of functionality from a user's point of view. The fourth level was *tasks* which described in technical terms what needed to be done to realize the user stories.

At the beginning of the project, the Product Management team was responsible for creating the features and epics and prioritizing them. They were supported by the product architecture team, the user experience team and the engine development team representatives. Together, these stakeholders represented the customers and users of the software in the Release Iteration Planning events. Since one of the goals of the project was to create a new version of the product using the new back-end architecture, many features were required to replicate the functionality of the previous version of the software on the new architecture. Initially, the product management did not see value in prioritizing features until the functionality of the previous version was completed, as they considered that the product could not be released before the completion of those features.

The development teams together with their Product Owners were responsible for creating user stories based on the features and for planning the contents of the development sprints. The implementation order of the user stories was typically based on the dependencies between user stories.

Non-functional requirements, such as usability, reliability, performance and supportability, were included in the feature definition when the nature of the feature required it. Performance standards concerning the memory and processor load created by the software were explicitly defined for the whole system. The user experience team representative provided graphical layouts and

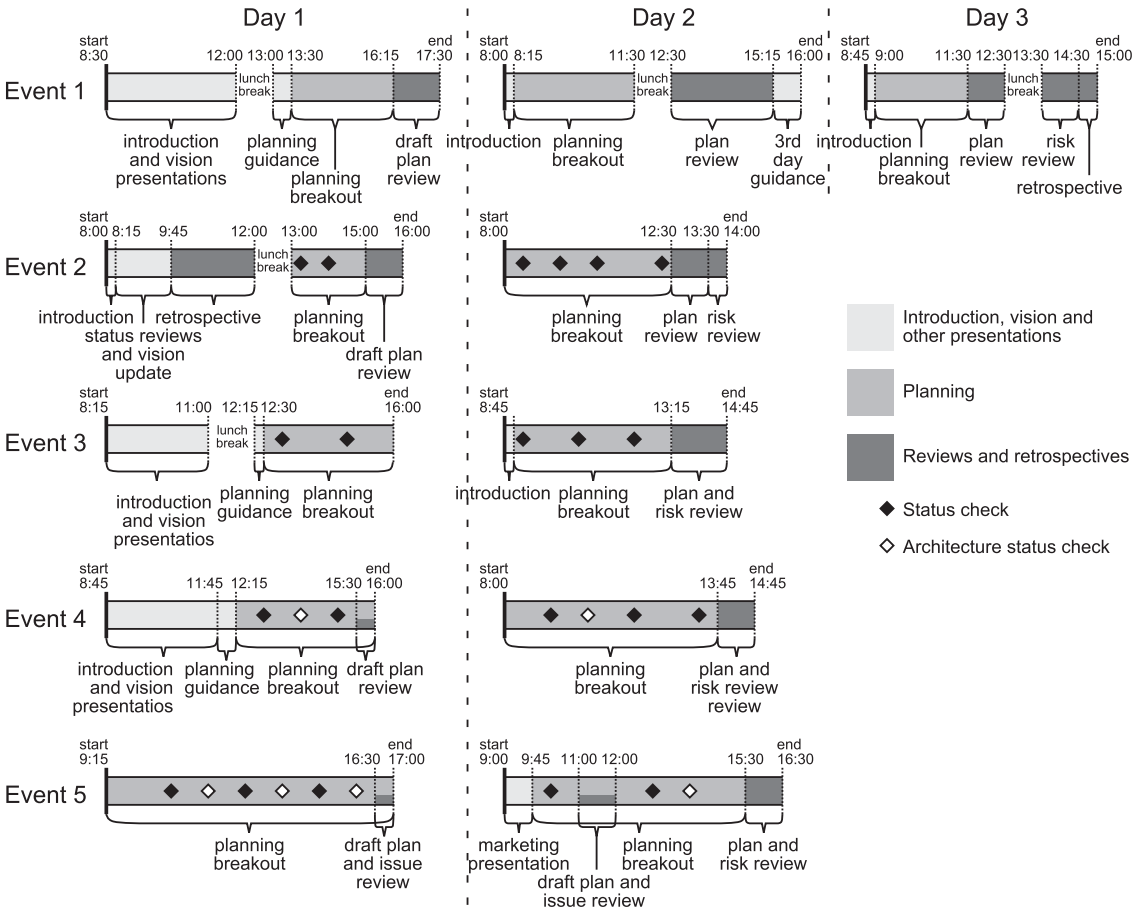

Fig. 2. Schedules of the Release Iteration Planning events.

usability guidance in the planning events. Non-functional requirements were also elaborated in the events by informal discussion between the participants when required.

Initially, each release iteration was expected to end in a release of a test version of the software, or in the case of the last release iteration, in the release of the public, feature-complete version of the software. The purpose of the test releases was to be a milestone for the project, to practice and test the release process, and to gather feedback from test users. The test users belonged to the company's customer feedback program. The test releases were to be the main method of validating if the right requirements were implemented and if the features were implemented in a satisfactory way. The features included in the test release were planned to be complete and of publishable quality.

5.4. Release Iteration Planning Event 1

In this section we first provide an overview of the Release Iteration Planning Event 1. Then, we describe the different segments of the event: introductory presentations, team planning breakouts, draft plan reviews and the final plan review. Finally, we describe the feedback report we gave to the case organization after the event.

5.4.1. Overview of the event

Project α officially begun with Event 1, although materials such as feature and architecture descriptions and user interface guidance had been prepared beforehand by the responsible stakeholders. The overall goal was to create an initial plan for the first release iteration on the feature and user story level. A three-hour training session was conducted a day before Event 1. The purpose of the training session was to give an overview of the agile practices which were to be used in the project. Event 1 was originally scheduled to take two days. After the first day, the external consultant (who also facilitated the event) and the Release Project Manager decided that the planning needed to be extended into third day, as the planning was not close to completion.

Fig. 3 illustrates the project organization, the stakeholders and the work items in Event 1. Most developers from the nine local development teams were present during the event. One developer or tester from each team took on the role of team Scrum Master, with approximately 50% of the working time allocated to the Scrum Master work. The Malaysian team did not have a Product Owner. Instead, the team was jointly managed by an architect, the Product Managers and the team's Scrum Master. One developer and the team's Scrum Master were present from the Malaysian team. There were three Product Owners participating in the event. One Product Owner was guiding the front-end teams and two Product Owners were guiding the back-end teams.

The project had several other external stakeholders with advisory or supporting roles: a user experience team representative, a representative from the engine development organization, a product management team that consisted of Product Managers and a Release Project Manager, and a product architecture team consisting of a lead architect and product architects. A Software Process Improvement (SPI) team also participated in the events. The team's purpose was the continuous improvement of the software development processes and tools in the company.

The external consultant, as an event facilitator, had an important role in the planning event. The facilitator made sure that the

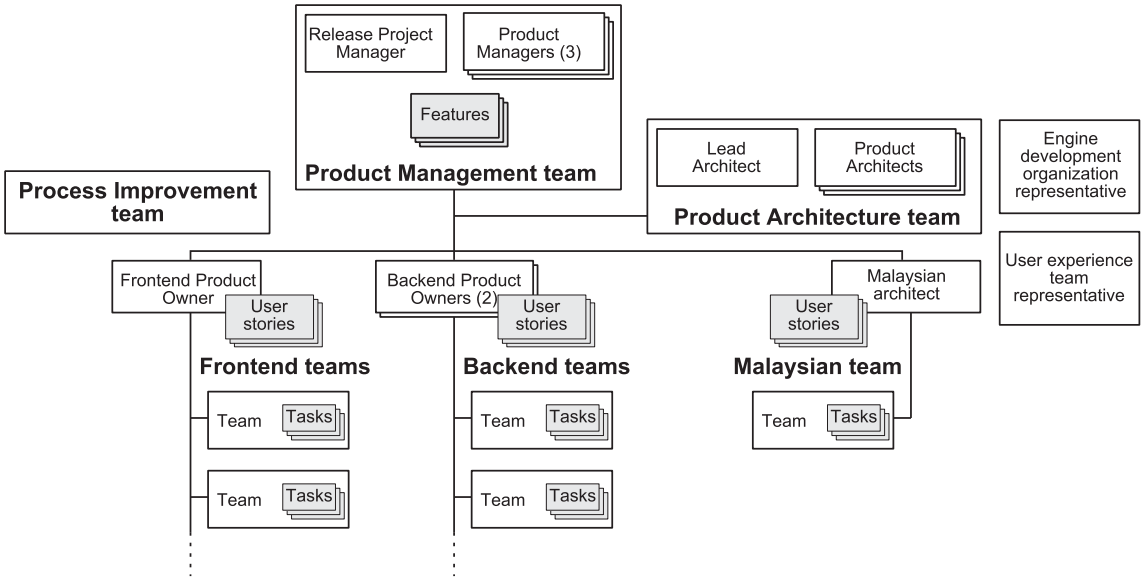

Fig. 3. Project α organization and stakeholders in Event 1.

event was proceeding as planned and within schedule, and solved conflicts and impediments that rose during the event.

The training day and Event 1 were conducted in an external space rented by the case organization. Fig. 4 shows an approximate floorplan of the space. The space had a separate area for presentations. Each team had a dedicated planning table. The tables were separated by movable walls which acted as planning boards and dampened noise, but did not hinder access between the teams.

5.4.2. Introduction, vision and planning guidance presentations

Event 1 began with several introduction and guidance presentations given by the Release Project Manager, a Product Manager, and different stakeholders. These presentations gave an overview of the goals and the schedule of Project α, more detailed information on the features intended to be implemented in the first release iteration, and instructions for architecture and user interface development. In addition, the facilitator gave a presentation on the practicalities and schedule of the planning event and gave planning instructions for the development teams. These included instructions for writing user stories, for writing high level objectives for the whole release iteration (i.e. release iteration objectives), and instructions for using different color sticky notes for recording user stories, dependencies, objectives, and risks. Finally, the teams were instructed to start the planning by discussing the product vision and features with their Product Owner.

5.4.3. Team planning breakouts

After the presentations ended, the development teams started planning the first release iteration. First, the teams gathered around their Product Owner. The Product Owners and the teams then discussed how the features should be assigned to the teams. After each team had been given at least one feature, the teams broke out to their own designated planning tables, hence this segment was called the (first) team planning breakout.

Eventually, with guidance from their Product Owner and, when required, from the Product Managers and other stakeholders, the teams split their features into user stories. The teams then scheduled the user stories into the sprints of the release iteration based on the estimated development capacity of the team. Fig. 5 shows several teams planning during a team planning breakout.

In Event 1, most of the teams did not write user stories from a user's point of view. Instead, they split the features into large technical tasks. During the event, they were repeatedly instructed by the facilitator to use the user story format, but we observed no changes in their conduct. We also observed that many teams struggled formulating their release iteration objectives.

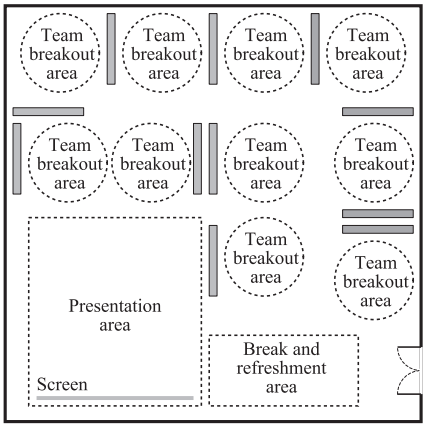

Fig. 4. Approximate floorplan of the event space in Event 1.

Fig. 5. Teams planning during a team planning breakout.

5.4.4. Draft plan reviews

The planning was coordinated using intermediate plan reviews. An intermediate plan review was conducted at the end of both the first and second day. In the intermediate plan reviews, all event participants gathered in the presentation area. A representative from each development team, which usually was the Scrum Master, shortly reported how their planning had progressed and how much time was still needed, what their unsolved issues were, and what dependencies, if any, they had discovered. Each team was given 4 min to give their presentation. The other participants were encouraged to ask questions and comment on the presentations.

Regardless of the short time given for each presentation, the intermediate plan reviews took a considerable portion of the time allocated for the event overall (see Fig. 2). From the approximate 20 h of effective working time, the intermediate plan reviews took approximately 4 h in total. Observing the participants during the plan reviews, we noticed that many developers seemed not to be paying attention to the presentations of the other teams. We also observed that most of the presentations were very technical; the teams explained how they planned to implement some functionality. This issue was also identified by the participants during Event 1:

> *The planning session felt too short, just when we started to get some traction there was someone coming to say you need to start wrapping up and writing those big pictures. . . . I think every day felt really short. I feel that we spent too little time with our own team and too much time listening to the other teams. . . . the explanations were really vague and we could not really understand the [other teams'] team internal speak.*
>
> [Developer, Event 1]

5.4.5. Final plan review

The final plan review was conducted at the end of the third day in a similar fashion as the intermediate plan reviews. Each team had 6 min of time to present the plan and objectives they had for the first release iteration. Fig. 6 shows a picture of a team member presenting the team's plan to the other participants during the final plan review. The teams had written risks on sticky notes during the team breakout sessions. Each risk was discussed briefly and assigned to a person or a team who would be handling the risk. The review was followed by a vote of confidence by a show of hands. First, all developers voted on how confident they were in their team's plan and then everyone present voted for confidence on the whole plan. In Event 1, both votes showed a high overall confidence level. The plan review was followed by a short discussion on how the event went. The facilitator asked the participants to voice their opinions on what went well and what did not go well.

Fig. 6. A team member presenting the team's planning board during the final plan review.

5.4.6. Feedback by the researchers

After the event, the researchers wrote a feedback report with the following improvement suggestions: each team should have a dedicated Product Owner. Features should be tentatively preassigned to teams before the planning event. Teams should split features into user stories, instead of large technical tasks. Instead of having the lengthy daily status checks, there should be an hourly short status checks where only one representative from each team participates. Most of these suggestions were followed in the later events. Only writing user stories instead of large tasks continued to be an issue in the later events.

5.5. Events 2–5

Overall, Events 2–5 were conducted quite similarly to Event 1. The number of development teams varied slightly over the course of the project. The Scrum Masters of the Malaysian teams were physically present in the planning events, and the Malaysian teams participated in the events remotely via a videoconferencing system. Before Event 3, a contracted Polish team was added to the project. The team was considered a part of the project, although they were developing an independent small component. The Polish team was represented in the event by their Scrum Master. The notable changes that were made to the Release Iteration Planning method over the course of Project α are summarized in Table 3 and described in more detail below. We also describe the feedback reports we sent to the case organization.

5.5.1. Feature prioritization and assignment

Starting from Event 2, the Product Owners and Product Managers prepared a prioritized list of features and preliminary assignment of those features to the teams. The priority order and assignments were tentative and could be changed during the planning event.

5.5.2. Table for stakeholders

Starting from Event 2, the Product Managers, architects and other stakeholders present in the event had a reserved table in the planning space where they could be found when they were not working with the teams. According to the surveys (see Fig. 9), the support the teams received from the stakeholders, the Scrum Masters and the facilitator was much better in Event 2 compared to Event 1.

5.5.3. Short status checks

The third change in Event 2 were short status meetings: two meetings during the first day and four during the second day. Besides the short status meetings, a draft plan review was still arranged at the end of the first day. These status checks were participated in by a single representative of each team and by the Product Owners and the Product Managers. The goal was to decrease the overhead created by the lengthy draft plan reviews in Event 1, to provide better visibility to the progress of planning and to assist in identifying and solving inter-team dependencies. After Event 2, no draft plan reviews were conducted in the rest of the events, since the frequent status checks had made the draft plan reviews obsolete. Only the final plan review in the end of the last planning day was kept.

5.5.4. Planning matrix

In the Event 4, a planning matrix wall was introduced. On the matrix, each column represented one software development team

Table 3
Notable changes to the Release Iteration Planning method.

Event	Change compared to the previous event(s)
Event 2	A prioritized list of features and preliminary assignment of the features to the teams Short status checks during the planning breakouts A table reserved for the Product Managers, architects and other stakeholders
Event 3	No draft plan review at the end of the first day
Event 4	Introduction of the planning matrix Alternating planning status checks and architectural status checks
Event 5	Reorganization of the development teams, introduction of full-time Scrum Masters A few teams planned with user stories instead of large technical tasks Number of stories brought to the event was limited to estimated capacity plus 15% Presentations were given the day before the event

and each row represented one sprint. Whenever a team finished planning a feature, the wall was updated to show when the team planned to start and when to finish the development of the feature. Fig. 7 illustrates the planning matrix. The location of the feature sticker indicates the beginning of the development and the tail indicates the time span of the development. The purpose of the planning matrix was to provide an overview of the feature development schedule and of the progress of the planning. The planning matrix appeared to help in the identification of risky or conflicting feature development plans.

5.5.5. Architectural status checks

Starting from Event 4, there were two types of status check meetings: *planning status checks* (explained above) and *architectural status checks*. Both types were held in front of the planning matrix. The status checks were held approximately once an hour, alternating between the planning status checks and architectural status checks. The architectural status checks were led by the lead architect. The goal of the architectural status checks was to solve architectural and technology related risks, dependencies and other architecture-related issues. Typically architectural status checks were quite short, but allowed effective identification and solving of issues. Issues were raised during the status checks and smaller groups of people suitable for solving the issues continued discussions after the status checks.

5.5.6. Re-organization of the development teams

Between Events 4 and 5, a portion of the development organization was laid off or moved to other projects due to an internal reorganization of the company's Finnish site. The lay-offs left many development teams short-handed. A week before Event 5, the Finnish development teams were rearranged. The teams were disbanded and new teams were formed from scratch. Each team was also assigned a dedicated full-time Scrum Master. Seven Finnish development teams were formed. Two of the teams were end-to-end teams, one was a back-end team, and four were front-end teams. Moreover, in Event 5, we observed that several teams were using user stories instead of large technical tasks when planning. This might be attributed to the introduction of the full-time Scrum Masters, many of which had formal Scrum Master training.

5.5.7. Limiting the number of features

According to an SPI team member, the number of features brought to Event 5 was limited to the estimated total capacity of the teams plus 15%. The goal was to reduce and focus the preparation work of the Product Managers and Product Owners. The smaller amount of features allowed them to better elaborate and prioritize the included features. In addition, the shorter feature list reduced the development team's pressure to over-estimate their capacity. Since they did not see the whole backlog of features that were proposed to be included in the product release, they planned according to their real capacity instead of trying to fit the whole backlog in the plan, which would have created an unrealistic plan.

5.5.8. Introductory presentations before the event

Instead of having the presentations the same day as the planning, the introductory and guidance presentations were given the day before Event 5. Between the presentations and the first planning day, the managers and architects had time to improve materials and guidance based on questions raised during the presentations. This also enabled the teams to start planning straight away at the beginning of the first day.

5.5.9. Test version releases

The organization failed to publish a test release after the first release iteration due to the incompleteness of the features and due to quality issues. After the first release iteration, the management decided to try to publish test release every two weeks. However, due to the similar incompleteness and quality reasons, only

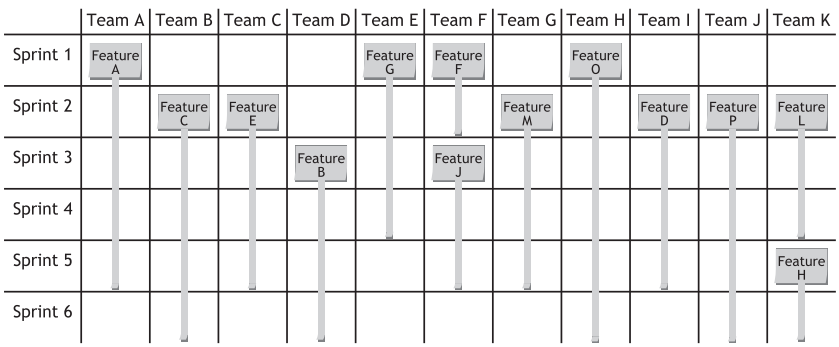

Fig. 7. Illustration of the planning matrix.

approximately half of the test releases were successful between Events 2–5. Two of the four test releases were successfully published during Release Iteration 2, three of the six during Release Iteration 3 and two of the four during Release Iteration 4. After Release Iteration 4, most of the bi-weekly test releases were successful.

5.5.10. Feedback reports

The researchers gave feedback to the case organization after Event 2. The report contained the following observations and recommendations: the communication between the teams was much improved from the first event, but there were some individuals who seemed to have an attitude problem towards the Release Iteration Planning method. Teams still planned using large technical tasks instead of user stories. The new status checks seemed to work well and we suggested that the draft plan reviews could be dropped. The preparations for the event were lacking, and many features were too vague to plan with.

The researchers wrote a feedback report also after Event 5 with the following observations and recommendations for further improvement of the method: the case organization should make sure that the two new end-to-end teams are really working in a cross-functional way, instead of working internally as separate teams. While it looked like the planning was nearly completed after the first planning day, in the beginning of the second day it became apparent that the initial plan was unrealistic and there were still many open issues. Two days seemed to be the minimum timeframe for a Release Iteration Planning event in this kind of a project, since the additional time between the days allows the stakeholders to solve issues identified during the first day and to create additional materials to guide the planning during the second day.

According to the post-project interviews, the observations and recommendations in these feedback reports were agreed by the SPI-team members who read them. No feedback reports were written after Events 3 and 4, as the researchers did not observe any new phenomena significant enough to report.

5.6. Finalizing Project α

Event 5, held in December 2010, was the last Release Iteration Planning event in Project α. After the fifth release iteration, ending in March 2011, the project was considered to be in the finalization stage and it was decided that further Release Iteration Panning events were not needed.

The first version of the product was publicly released in October 2011, with fewer features than was originally planned. Thus, reaching the first public release took two years instead of the six months the product management team had originally envisioned. The project continued development after the first public release with a smaller number of teams divided into front-end and back-end sub-projects. New releases were made every quarter. An interviewee summarized the project in the following way:

> *...initially this was supposed to be 10 km run which might take 40 min, in the end it was a marathon where we got to the goal barely before the time limit, six hours. ... even though we got to the goal, ... the thing we had at that point was really not what we were supposed to have.*
>
> [Product Manager, Post-project interview]

The interviewees gave several reasons for the initial over-optimism in the schedule and scope of the project in the post-project interviews. First, building the new product architecture took considerably more effort than what was initially expected. Second, the product management team assumed that a certain large component of the software was almost completed when the Project α begun, when in reality what was available was more akin to a prototype. Third, the developers tended to over-estimate their capacity and under-estimate the effort required by the requirements they had planned to implement during a release iteration. Fourth, the development organization expected that the adoption of Scrum and the Release Iteration Planning method would increase the speed of the development.

6. Release planning in Project β

Project β was distributed between Finland and Russia. The four Russian development teams and the Russian Product Owners collaborated with the Finnish Product Managers and the lead architect.

The Release Iteration Planning method had been taken into use in Project β because the project organization was planning big changes to the product they were developing and they had heard good things about the Release Iteration Planning method from Project α members. The Project β organization was especially looking to improve requirements transparency so that the developers would better understand what was expected of them and why.

The overall conduct in the Release Iteration Planning events was similar to the ones in Project α. The events were scheduled to last two days. Before each event, the Product Managers, the Product Owners, and the lead architect prepared a prioritized list of features. The planning events begun with a Scrum-style retrospective where all members of the development project tried to solve issues that had been identified during the previous release iteration. Next, presentations on the current status of the project, on the goals for the next release iteration, and on technical information were given. This typically left a few hours for the team planning breakout during the first day. The second day was mostly reserved for the team planning breakout. Short status meetings were conducted every other hour. The second day ended in a collective plan review. All 12 Release Iteration Planning events, conducted by the time of the post-project interviews, had been led by a facilitator from the SPI team. This was perceived as an important practice as the facilitator had brought an external point of view to the events.

Our interviewees emphasized that good preparations were essential for the success of Project β. All features had to be prioritized and the backlog had to be in a good shape before the planning events. They had accomplished this by arranging backlog grooming sessions on two levels: the Product Managers, the Product Owners and the lead architect met face-to-face in monthly grooming workshops, and the Product Owners arranged a grooming workshop together with the teams before each Release Iteration Planning event. However, our interviewees mentioned that they should pay even more attention to backlog management in the future.

In the Release Iteration Planning events of Project β, the Product Owners and Product Managers had recently tried to replace a portion of the status check meetings with status visits to each team. The benefit of the status visits was that the participants were able to see each team's planning wall and ask detailed questions about it. However, our interviewees expected that this might decrease the inter-team visibility which was created by the status check meetings. In this small project, with only four teams, this was not seen as a major challenge and they were planning to continue with the status visit practice. Our interviewees mentioned that in the future they aimed at shortening the planning events to one day by putting even more effort into preparation. By the time of the interviews, the shortest Release Iteration Planning event had taken one day and a half.

Overall, the interviewees thought that the members of the Project β organization were happy with the Release Iteration Planning method. It had made the planning easier in this distributed project

and improved transparency between the sites and between the Product Managers and the development teams. According to an interviewee, the process was well received and perceived to be useful:

> *...widely it [the Russian site] is considered the most successful of the sites that have adopted the joint release planning method. ... And it would not be so impressive if it was only performed in [Russia], but because it is led solely from [Finland], which has a different language and a different time zone...*
>
> [SPI manager, Post-project interview]

7. Benefits of the method

In this section, we present the benefits of the Release Iteration Planning method experienced in the case organization. Overall, the Release Iteration Planning method was regarded as successful in both case projects. All the expected benefits were at least partially realized and additional benefits were recognized. Even though Project α cannot be seen as a total success, the Release Iteration Planning method positively affected the ability of the project to finally create a public release. In Project β, at the time of our interviews, the Release Iteration Planning method had been applied successfully twelve times, and our interviewees saw that this method had brought impressive results.

All our data show an overall positive attitude towards the Release Iteration Planning method. The members of the case organization saw the method as a clear improvement to the previous projects. In the surveys conducted after the first two Release Iteration Planning events of Project α (Survey 1 and Survey 2), the project members rated both events positively. Survey 1 had a median of 5.0 and Survey 2 had a median of 4.0 on a six point scale (1 = Poor, 6 = Excellent). See Fig. 8 for details. Comments on the method in the observed retrospectives were positive as well.

> *We were able to identify some good things that had happened so far. Number one, clearly, was the new planning method. So people liked this way of working, they think it does bring benefit.*
>
> [Manager, Retrospective $R_{\alpha 1}$]

Moreover, all the persons interviewed in 2012 regarded the method as successful and were eager to take it into use in similar future projects. In the case projects, we recognized ten benefits of the Release Iteration Planning method, which we will discuss below.

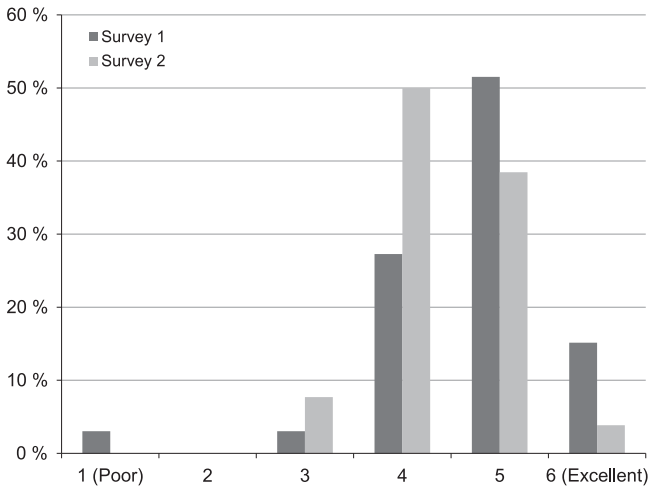

Fig. 8. Results from the survey question "Overall, how would you rate the whole release planning event?".

7.1. Clear and unified goals for all stakeholders

As the Release Iteration Planning events gathered the whole organization in the same space regularly, it was possible to create clear and unified goals for all stakeholders, from Product Managers to developers.

> *...there the whole group gets a common direction and sees who actually belong to the project and can manage the dependencies faster, already during the event.*
>
> [R&D Line Manager, Post-project interview]

> *It is really important that in two, three months interval, we gather everybody in the same room, Product Owners, architects, Product Managers, everybody sees each other's faces, especially when we are not at the same office, and we gain a common direction.*
>
> [Scrum Master, Post-project interview]

The results from the surveys in Project α also support this benefit. The statements: "The necessary information was readily available thorough the event (e.g. agenda posted, handouts distributed, etc.)", "I have a clear vision of what I am going to do for the next business iteration" and "My team's plan for creating the next release is realistic" got medians of 5 (Agree) in both surveys. The statement "I believe this project will create a successful solution" got median of 5 (Agree) in Survey 1 and median of 4.5 (between Agree and Slightly agree) in Survey 2. See Fig. 9 for details of the survey results.

7.2. Fast recognition of the size and challenges of the project

According to the post-project interviews, an important positive aspect of the Release Iteration Planning method was that it helped the project organization to realize the true magnitude of the work in Project α. Otherwise, most probably, it would have been realized much later. Moreover, many of the challenges of the project came up and to the knowledge of all stakeholders during the Release Iteration Planning events. Our interviewees presumed that even though individual persons would have known the challenges or thought that the project would require higher effort than expected, this would not have became to the knowledge of a large group of managers and other stakeholders as early as it came now.

> *Good was that we became aware of, already in the beginning ... that this project is massive ... What we wanted complete during this project is impossible. ... After one and half months, when we had the first re-planning, then at the latest we could see that this is much more massive operation than we had realized. And without that kind of events, I believe, we would have realized this many months later.*
>
> [Scrum Master, Post-project interview]

7.3. Improved communication and transparency

The collaboration and visibility between the Product Managers, Product Owners and the development teams had been a big challenge in the organization, as these groups had traditionally worked quite apart from each other. Thus, the improved communication and transparency between these groups was one of the benefits that the organization hoped to gain from the method. This benefit was clearly realized in both case projects, although this happened gradually. Both the developers and the managers thought that the communication and especially the transparency between the groups had improved immensely.

> *The first time ever there has been this kind of visibility for the developers to what the other teams are doing. It is better than ever before. Previously, it has been that the project steering group*

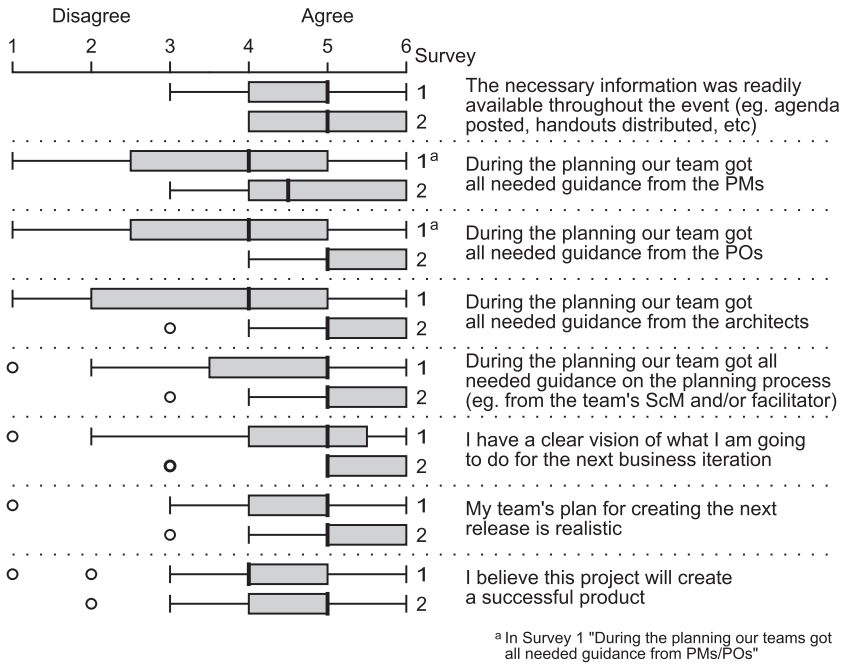

Fig. 9. Results from the surveys.

manages everything and developers don't really know what is happening in the other room.

[Developer, Event 1]

. . . the teams had much better visibility on what was coming, the Product Owners had better visibility and the Product Managers had better visibility on what they were getting, what was possible. It just improves the communication at several levels.

[Scrum Master, Post-project interview]

In the Event 1 survey, 15 of the 19 answers to the open ended question on "the best parts of the joint release planning event" stated collaboration and communication as the best parts. A few representative excerpts follow.

It was easy to discuss with all the relevant people of different problems and synchronize work with other teams in case of dependencies. Also visibility of other teams' objectives was good.

[Anonymous respondent, Event 1 survey]

Having business people present all the time. Showing business how realistic R&D think the targets are right at the beginning, and not 2 months before the release.

[Anonymous respondent, Event 1 survey]

Collaboration between all feature teams and [product] management. All stakeholders are now focused on the same things, with the same expectations.

[Anonymous respondent, Event 1 survey]

The Release Iteration Planning events gave the developers better visibility on what was coming in the future and allowed them to directly ask questions and clarifications from the product management and other stakeholders. The Product Managers saw how the development teams worked, what was problematic, how complex and demanding the developers considered the different features and what the developers considered possible to implement in the following release iteration. Thus, the predictability started to improve after the first few planning events in both case projects.

I cannot think of any other way how we could accomplish the same transparency and fast problem solving [in Project β]. I do believe in this method, especially when I have seen how it works at Russia . . . there business gets transparency to the practical work, especially when in that case the business is located in a different country.

[SPI manager, Post-project interview]

7.4. Fast identification and management of dependencies

In the Release Iteration Planning events, the regular status checks revealed dependencies between the teams and issues regarding the dependencies.

. . . from the team perspective we had to learn that we cannot just think that 'if we do this [our own tasks], then everything is fine' . . . that if you get caught that 'Ops we had a dependency to there, but we didn't notice it.' . . . so this kind of things were so much better visible.

[Product Owner, Post-project interview]

As the dependencies were noticed quickly and all relevant stakeholders were present, it was possible to discuss how to coordinate the teams and solve the issues right away. Dealing immediately with the dependencies made it possible for teams to continue their planning without the need to wait or to leave the issue unsolved.

7.5. Enabling inter-team coordination

We observed that during the first few events of Project α, the teams expected that the other stakeholders, like the Product Owners and architects, would take care of the coordination and solve inter-team dependencies. However, gradually the teams learned to do this by themselves.

As all the teams were present in the events (at least by using a telecommunications system in Project α), coordinating with the other teams was straightforward. Moreover, during these events the team members from different teams learned to know each other and collaborate. Thus, the inter-team coordination became easier also between the events, as the project organization members knew each other and knew the assignments of the features.

7.6. Fast recognition and mitigation of impediments and risks

Recognizing impediments and risks was part of the daily schedule during the Release Iteration Planning events. As the impediments and risks were recognized collaboratively, the collective knowledge improved the coverage of the recognized risks. Moreover, one of the aims of the events was that the impediments and risks were discussed and, if possible, mitigated already during the events. As the impediments and risks were brought to everybody's knowledge, everybody could affect on mitigation of the impediments and risks during their work.

> *...it was especially good that in the end of the events, we checked the impediments, listed them from the whole project so that they were clearly visible, ... and we tried to resolve them as far as possible and assigned owners.*
>
> [R&D Line Manager, Post-project interview]

7.7. Fast decisions in the events

It was seen as very important that all stakeholders were present or represented in the Release Iteration Planning events, as that made it possible to get everybody needed together immediately to discuss identified impediments and make decisions right away on how to solve them. Previously that kind of decisions could take up to several weeks.

> *Without a doubt a good thing was that everyone was in the same space, it was the reason why it works, the reason why it was so useful. Whatever question comes up, you have a person within 30 meter radius who can answer that question.*
>
> [Scrum Master, Post-project interview]

> *Things happen really fast [during the event]. Things that would take weeks to handle over e-mail and over telephone ... in there you gather those three or four people, drag them into a corner and state that we have this kind of issue, what shall we do?*
>
> [SPI Manager, Post-project interview]

7.8. Meeting people face-to-face and creation of a project spirit

Both case projects were distributed and even the people on the same site sat separately. Managers sat in their rooms or cubicles and each team had a dedicated team room, which were located on several office floors. Many team members did not know what the other teams were doing. In the Release Iteration Planning events the Product Managers met and discussed with the team members. Previously, they had been somewhat isolated from the development teams. In the predecessor project of Project α, Project ω, some of the teams had worked on the front-end and others on the back-end. Consequently, the members of the different teams did not initially know each other well. In the release planning events the participants met each other face-to-face. The intensive effort of working a few days in a row in the same space towards a common goal helped create a project spirit.

> *...I didn't know the teams and others beforehand, everyone was there, and we saw that this is 'us', this is our project, kind of 'us' spirit was born. ... I came to know the teams because otherwise the teams are easily anonymous...*
>
> [Product Manager, Post-project interview]

> *[otherwise] I would have never seen all those people over such time. ... if I as a Product Owner saw a lot of people, others did too and started to talk in a different way. ... it was not anymore that people would just be in their own team room and that nobody would disturb them.*
>
> [Product Owner, Post-project interview]

7.9. Showing the importance of the project

Organizing the release planning events where all product managers were present conveyed the importance of Project α to the participants. The CEO of the company visited the first planning event, which was also an sign of the importance of the project. In Project β, sending the Product Managers and the lead architect to the planning events in the Russian site conveyed the development team that they were valued and that the project was important for the company.

7.10. Enabling the successful finalization of the project

Even though Project α took much longer than expected at the beginning, many of our interviewees mentioned that the Release Iteration Planning method immensely impacted to the finalization of the project. First, it revealed during the first few events that the project was much bigger than expected. Without the events the size of the project would probably have been realized much later. Second, the events forced the Product Managers and Product Owners to collaborate. They had to prioritize the features and to leave parts of the originally envisioned functionality out. Third, the events enhanced the inter-team coordination and enabled immediate dependency management. Finally, our interviewees admitted they could not think of a better way to perform release planning in such a large project:

> *The first event, I have heard, was a quite painful experience to everybody, but after that everybody has agreed that this is a good way of working, much better than what we had previously.*
>
> [Scrum Master, Post-project interview]

> *I don't believe we could have managed without [the release planning events]. Because then we had that amount of teams working on exactly the same area. Who must share the understanding. And several sites. ... I have really hard time coming up with an another way which we could have used instead of that kind of large-scale release planning.*
>
> [Product Owner, Post-project interview]

8. Challenges in the application of the method

Even though the Release Iteration Planning method received generally positive comments, there were several challenges in its application in Project α. Many of them originated from the company's previous ways of working, as their application of the Scrum method and agile thinking was still somewhat lacking when they started the project. In this section, we concentrate on the challenges the company had in applying the method, especially in Project α, but also to some extent in Project β. We recognized nine challenges the case organization had in the application of the Release Iteration Planning method. These are discussed below.

8.1. Lack of preparation of the requirements before the events

The lack of preparation of the requirements was most notable in Event 1 of Project α, but it continued to be a challenge also in other events. At the beginning of the project, the Product Managers did not understand the nature of the planning events and it seemed that they tried to prepare all possible requirements beforehand. This created confusion when the first planning breakout started in Event 1. The different Product Managers had not synchronized the requirements between each other and the requirements were not described at a low enough abstraction level for the team members to be able to plan effectively. In addition, the overall number of requirements, which were not prioritized, was many times more than what the teams could realistically implement during the next three-month release iteration.

. . . Week before [Event 1] I saw those business requirements. It was shocking 75 pages of Power Point. No synchronization, you could see that one person had filled in one part and another person had filled in another part, and third person had filled in yet another part. They had never looked crosswise together whether these requirements are even synchronized, and they were not.

[Product Owner, Post-project interviews]

. . . the preparation [for Event 1] was really weakly attended to by us. Everything what came up there was new to the team members, . . . they didn't know what to expect, so for the most part the time went into fumbling around, clarifying basic things that should have been clear already in the planning.

[Product Owner, Post-project interview]

The same problem could be seen from the answers to the Event 1 survey. For example, the answers to the question "In my opinion, the biggest problems in this release planning event were" contained the following answer.

Insufficient preparations for the team breakout materials: too many pages with duplicates in features under different titles, tech & business materials separated (from teams' perspective) - how probable it is that anyone would have the time to go through such amount of material with good enough of understanding on the entity as the presentations never got to that level (epics/vision & features were not in par for the teams to operate efficiently)?

[Anonymous respondent, Event 1 survey]

The preparation improved in over time in the consecutive events as the managers and Product Owners learned to better prepare. However, during the whole study period we observed that communication between the development organization and Product Management was an issue. The development teams requested more guidance on the prioritization and implementation of features and clearer feature information materials.

At the time of the post-project interviews, preparation was not identified as an issue in Project β. Compared with Project α, the preparation was more systematic in Project β. The backlog grooming performed on the two levels made it easier for the teams to start the planning right away in the planning events. In addition, the backlog items were always prioritized before the planning events.

8.2. Lacking feature prioritization and allocation

The list of 135 features that the Product Managers brought to Event 1 in Project α was not in any priority order, nor were the features pre-assigned to the teams. Three Product Owners and ten development teams participated in Event 1. Everything was new to the development teams, from the planning method to the product architecture and features. The developers seemed to have difficulties in getting enough support from the Product Owners and other stakeholders. It took at least one hour for most teams to even select their first feature. The feature list was much better prepared for Event 2. The Product Managers and Product Owners had prepared a prioritized list of features which contained pre-assignments features to the development teams.

. . . in the following [release] iterations we then somewhat pre-designated them [features] based on competency and other things. And then we made sure that the thing that was fifth on the list, this is a thing no-one wants to do because this is kind of boring and difficult and no end in sight, this must be assigned to someone because the teams don't always want to take such things that are challenging and difficult, because they are afraid of failing and such things are taken fast that are impressive end-user features.

[Product Manager, Post-project interview]

According to our observations, the prioritization and pre-assignments seemed to help the teams to start the actual planning much faster compared with Event 1. We also observed that new information uncovered during Event 2 caused several changes to the feature assignments during the event, which suggests that the participants understood that the pre-assignments were only tentative.

According to the post-project interviews, Product Management's role in the feature prioritization in Project α was not clear. This issue existed from the beginning of the project until the reorganization preceding Event 5. The Product Management was officially responsible for prioritizing the features. However, according to the interviewed Product Manager, Product Management was unwilling to give the features a priority order, since there was a set of features mandatory to create a product which could be sold. Thus, from the business point of view, it would have made no sense to prioritize the features before the mandatory set was implemented. According to him, the prioritization of features should have been made by the Product Owners based on the most efficient implementation order of the mandatory features:

. . . we used an airplane [metaphor] then, that we cannot prioritize which is more important, an engine or a wing, because the air plane won't fly anyway and we cannot enter the market if we have an air plane that doesn't fly, that doesn't have seats. . . . [It is a matter of] implementation order because we . . . don't know which is more sensible so we don't want to make that decision, because [there are] those who know and can, so let them make it, Product Owners and architects. Because naturally we want a plane that flies and where you can fit 200 passengers, so whether you make a wing or an engine first . . . does not matter.

[Product Manager, Post-project interview]

According to an interviewee, before the reorganization of the development teams prior to Event 5 of Project α, the feature backlog was too unorganized and unprioritized to be of much use. The Product Owners were not allowed to touch the feature backlog. There were conflicting and misleading features, and different Product Managers had wildly different ways of describing features. After the reorganization preceding Event 5, the Product Owners took the ownership of the feature backlog, which allowed them to prioritize features and write them in a format they could understand. During Event 5, the concept of a minimum marketable feature, which is the minimal set of functionality that is requisite for the publication of a feature, was also discussed, but was not concretely adopted at that time. After the reorganization, the management and prioritization of features was performed in weekly meetings between the Product Owners, Product Managers, architects and other relevant stakeholders. According to an interviewee, this was identified as one of the main learnings from Project α:

. . . backlog ownership cannot be in a place where they don't understand what the actual implementation work is. Or [don't understand] how to write a feature in a way that the people who are implementing it, for example, can only understand it in a singular way.

[Product Owner, Post-project interview]

8.3. Lacking understanding of the creation of end-to-end features and feature teams

According to the large-scale agile model Project α initially followed, all features should be end-to-end features and provide functionality visible to end users, and all teams should be cross-functional and able to implement any feature. The goal was to create features that could be implemented during a single release iter-

ation and included in a release at the end of the release iteration to gather immediate user feedback.

Due to the history of the case organization, this was not possible in the beginning of Project α. The development teams had knowledge of either the back-end or front-end of the product. Therefore, end-to-end features required involvement from the front-end and the back-end teams. In addition, the Product Management had not previously written end-user oriented end-to-end features. They did not know how to write them and instead wrote new features according to their previous technical and component oriented way, but still managed to create features which often required development in the back-end and in the front-end.

As a result there were lots of dependencies between the front-end and back-end teams, which required keeping the front-end and back-end development synchronized. The teams should have taken care of this network of dependencies, but they lacked the skills to handle the situation. Moreover, dependencies between features caused even more complexities.

> *. . . we didn't know those dependencies beforehand. In the planning, we looked at them, and there were quite many [such] that the features were not sensibly created in a way that you could implement them alone, in a team. Instead, it was easily so that 'okay, we will take this, but we must get that from them [another team] before we can create it, but then those [yet another team] are waiting for this thing from us' . . . and then there is a network when you have dozen teams and each team is contributing to a couple of features per iteration over three months' time. It became very complex.*
>
> [Product Owner, Post-project interview]

The Finnish site development re-organization preceding Event 5 did not solve the issues, as the newly formed end-to-end teams seemed to have challenges in planning their work in an end-to-end fashion. First, the features brought to Event 5 by the Product Management were still component-oriented. We observed that the newly formed end-to-end teams had difficulties in creating end-to-end user stories based on the features. Second, we observed that these teams split into two sub-teams during the planning. One sub-team planned the front-end functionality of the features and the other sub-team planned the back-end functionality. According to the post-project interviews, the end-to-end teams did not perform well, and were disbanded soon after Event 5.

> *We tried this [end-to-end teams], but we didn't have in the backlog cross-functional work to give them. . . . it is a very technical backlog, it is written in a way that would make Ken Schwaber turn in his grave if he was dead. We did not have a back-end-front-end thing to give that would produce customer value.*
>
> [SPI Manager, Post-project interview]

> *It [end-to-end teams] did not work, unfortunately. The main reason, or one of the reasons, was that, at least in the beginning, we did not have end-to-end features to give to the teams. Thus, the teams kept receiving front-end features and back-end features, which means that those end-to-end teams split internally into two small teams.*
>
> [Scrum Master, Post-project interview]

In Project β, the teams were divided into front-end and back-end teams. However, most features were written as end-to-end features, which required collaboration between the teams and had caused some minor coordination challenges.

8.4. Challenges in learning inter-team coordination

During Event 1, the team members did not seem to realize that they needed to, in addition to planning the team's work, take care of the inter-team coordination. Instead, they expected that somebody else, such as the Product Owners and architects, would take care of that. The Product Owners and Product Managers became information bottlenecks, as they had to convey information between the teams in addition to guiding the planning process. On multiple occasions during Event 1, the Product Owners discovered that no team was planning to implement an important part of some feature. This was especially evident regarding dependencies between the front-end and back-end teams. The inter-team coordination issue was raised during Event 1:

> *To some extent the Product Owners were bottlenecks, they were not present all the time they were needed. And also when you were solving dependencies with other team, [and] they needed to ask something from their Product Owner and [he was] not there available to answer our questions.*
>
> [Developer, Event 1]

During the project the teams started to learn which team was doing what and with which other teams they had dependencies. Getting to know the team members from the other teams seemed to help, as well. Thus, the teams were able to perform more of the inter-team coordination themselves.

8.5. Challenges in allocating architecture planning

Another organizational issue, which existed over the whole study period in Project α, was the planning and implementation of the new product architecture. The project was supposed to implement fundamental changes to the product architecture. However, the planning of the new architecture was not very far at the time of Event 1 and some of the participants were concerned about this already after the first event:

> *Since we are doing so large architectural changes, the planning work should have already been started before the [release] planning. Now we were trying to figure out some architectural problems in the middle of [the] planning [event].*
>
> [Anonymous respondent, Event 1 survey]

It was not clear whether only the architects or also the teams should be involved in architecture planning and how should the dependencies rising from architectural issues be coordinated between the teams. Many groups identified challenges related to the architecture and dependencies in Retrospective $R_{\alpha 2}$, one example follows:

> *. . . which levels the architecture should be given to the teams, and what is the role of the lead architect or the lead software engineer. Those are the areas that require clarification and the definition of responsibilities.*
>
> [Anonymous participant, Retrospective $R_{\alpha 2}$]

8.6. No unified understanding of the planning mindset

According to our observations, and according to the post-project interviews, there was also an overall uncertainty regarding the goal of the release planning in Project α. During Event 1, many participants assumed that the goal of the event was to create a plan which should be precisely followed during the next, three month release iteration. However, according to the facilitator, the goal was to create the best possible plan given the time available and then adjust the plan based on new information which was uncovered during the release iteration. This also meant, that the stakeholders had to keep on talking to each other during the release iterations.

However, it was evident from the first planning event that the participants did not have a "unified understanding of the planning mindset". Instead, many of the participants expected that the

planned work would be exactly completed after the following release iteration. They were surprised when this did not happen:

Now [after Event 1] we had a plan ... Well, I expected that now we have planned it, the teams have planned it, that this is it what will come out. And then it took three months and then we looked what we got, and only half of it [what was planned] had been realized.

[Product Manager, Post-project interview]

Actually, I feel that it is still difficult for people to understand the purpose of the [release] planning. Because an engineer usually thinks that when you do some work, the goal is the end result. And in this case it is not so, the purpose of the work is the process which creates the end result, and the plan which comes out is secondary. And this is terribly difficult to explain.

[SPI Manager, Post-project interview]

This issue alleviated over time, but according to our observations the planning mindset was not completely unified in Project α during our study period. In Project β, after the twelve planning events, the planning process and activities were well established and the goal of planning better understood.

8.7. Leaving no slack in the plans

At the beginning of Project α, the company had limited experience with the use of agile methods in large projects. They had challenges both in making realistic estimates as well as in leaving slack in the plans. This led to unrealistic plans, and as plans did not come together exactly, it was easy to blame the new method.

...this is a quite common issue, that we plan too optimistically ... and then we have noticed that we do not know enough about these things, which has made the planning even more difficult.

[Product Manager, Post-project interview]

The development teams continued to over-estimate their capacity during the whole study period and most teams included a little or no slack in their plans. On the other hand, Product Management had considerably underestimated the overall amount of work required to complete the project.

We though that yes, we can in three or six months get these things done, but they were regardless so large ... [that] perhaps more realistic would have been [to estimate] ... that this is really a two year project.

[R&D Line Manager, Post-project interviews]

Later on in Project α, the managers learned that they constantly could expect to get only about half of what was planned. This doubled the length of the project in the eyes of the management, which caused some additional challenges with the development budget and marketing of the product.

The first release planning event, I don't know how accurately it [the plan] was realized. After that the degree of success was approximately 65%. I.e. two thirds of what was planned was realized. And at some point we realized that this is how it is ... we asked the teams for commitment, and took half of it, and called it a high confidence plan.

[Scrum Master, Post-project interview]

Taking in and planning too many features was not efficient and made the events longer in Project α. In Project β, the organization had learned from this. They did not take in so many new features, the planning horizon (or release iteration) was only two months, instead of three months, and the teams got familiar with the upcoming features in the backlog grooming workshops.

8.8. Effort invested in the planning events

The Release Iteration Planning events were a big investment for the case organization. In Project α, the whole project organization spend two to three days every two to three months in the planning events. The managers interviewed in the post-project interviews raised the perceived low efficiency and large man-hour investment as an issue:

We performed this [Release Iteration Planning] for two [or] three quarters and noticed that it is a monstrous investment. [External process consultant] is present for three days, all teams, and the plan is in that sense poor that it doesn't reflect the end result.

[Product Manager, Post-project interview]

The interviewees had different opinions on whether this investment was worth it. On the other hand, the interviewees also stated that given the novelty of Project α and the size of the development organization, they could not think of a better way to perform release planning:

...we calculated that these are the meetings that would be needed [without the Release Iteration Planning events] ... it would take three weeks only for the meetings where they talk and agree on things. ... I don't see it [the effort put into release planning] as a downside ... on the contrary, if it is performed correctly, it saves time later on, as everyone has the same goal in mind, as everyone hopefully has dependencies in control. From this offset I believe it is absolutely worthwhile.

[R&D Line Manager, Post-project interview]

Several of the post-project interviewees expressed that one of the improvement goals regarding the method was to shorten the length of events to one day. In Project β, the Release Iteration Planning events had already been shorter in duration than in Project α. They lasted for from one day and a half to two days. The interviewed manager, who had facilitated several of release iterations planning events in Project β, believed that with proper preparation they could shorten the length even more:

It is two days currently, but it could be squeezed to one day only, by arraigning the retrospective separately ... and putting even more effort on grooming before the events.

[Project β facilitator, Post-project interview]

Project β was much smaller than Project α, employing only four development teams. Moreover, the Project β organization was already quite experienced with the Release Iteration Planning method, as they had been using it for two years. According to our interviewees, one day planning events might well be possible in a project like Project β. For a large project containing several new elements, such as Project α, one day would most probably not be enough.

...However, if the [preparatory] work is unfinished ... then you just have to use the three days. I would not shorten it by force, if it is just not possible, if the organization is not ready for it. ... Especially if a totally new thing starts, a new project, ... then you take a bit more massive event, that you sit two days and try to create a common understanding on what is going to happen, what is possible.

[Project β facilitator, Post-project interview]

8.9. Over-optimism towards the new method

A lot of expectations were placed on the new method. In the case company it was expected that the application of the new method with the help from an experienced facilitator would make Project α a success, even though there was much new for the com-

pany and lots of challenges; The project was the biggest one in the company's history thus far, it employed more teams and involved more geographically distributed sites than ever before, and it strived to build a new architecture for the existing product while keeping the normal yearly release rhythm. The initial expectations towards the increased development speed brought by Scrum and the Release Iteration Planning method were clearly overly optimistic.

> *...there were gigantic problems. First, we wanted a normal yearly release, ... at the same time we had to do a totally new architecture and at the same time implement new business models, that nobody actually knows what they are ... So, a lot of unrealistic expectations from everywhere, everyone of these would be too much as such, and then we try to do all that at once and then we add subcontracting...*
>
> [SPI Manager, Post-project interview]

> *...the whole autumn was promoted [to us] that with this [release planning method] we can get this whole thing done ... and there had been discussions how much this agile and Scrum makes it faster. So they [business] saw that this allows them to put there just any amount [of new features].*
>
> [Product Owner, Post-project interview]

9. Discussion

In this section, we first answer the three research questions and discuss the implications of the results. We also discuss the implications of our findings to the software engineering theory, our contributions to software engineering practitioners, the threats to the validity and limitations of our results. Finally, we discuss future work on the topic.

9.1. RQ1: How did the case projects adopt the Release Iteration Planning method in practice?

Overall, the adoption of the Release Iteration Planning method was considered a success, although Project α took considerably more time than the managers of the project originally expected. The method was also successfully adopted in the considerably smaller Project β. The method was considered very helpful in Project β.

Although several changes were made to the preparations and to the communications practices over the study period, the conduct in all the events in both projects was quite similar. First, the participants were informed about the current status of the project. Second, the participants were given information on what they were expected to contribute in the following release iteration. Third, the participants planned the following release iteration. Fourth, the plans and the associated risks and issues were reviewed.

The Release Iteration Planning method was based on both informal and formal communication between the software developers and other stakeholders who were gathered to the same space. The informal communication became more and more prevalent during the Release Iteration Planning events in Project α. This suggest that the informal face-to-face communication between the participants was considered more efficient than the formal communication in the form of presentations and status reviews although the more formal, regular short status meetings were required to synchronize the work of the teams and to solve issues. This finding is in line with the agile software development principle of direct and informal communication. Our results suggest that the principle holds also during Release Iteration Planning in a large, multi-team development organization.

Fogelström et al. [1] claim that agile software development methods are inherently misaligned with the needs of market-driven software product development. Possible problems they claim to be caused by agile methods in market-driven software product development include short-term thinking, architectural deterioration, integration problems, limited understanding about the value and cost of requirements and the increased difficulty of change management [1]. Our results suggest, that with some alterations, agile principles can be successfully applied also in marked-driven software product development. Although some of the problems listed above were observed in Project α, our analysis suggests that the observed problems were created by the transition period from a plan-driven to an agile development method and the problems alleviated when the organization become more mature in their agile adoption. The challenges are discussed in detail in Section 9.3.

Table 4 shows how the Release Iteration Planning method addresses several characteristics of the release planning problem on both operational and strategic level. However, it does not solve all issues related to it. Gut feeling, lobbying, politics, sell-in and strong individuals [28] may affect the decision making in the Release Iteration Planning events, although the presence and availability of the whole project organization may mitigate the negative effects. The product management still has the responsibility for analyzing the complex and competitive market and making strategic decisions. The Release Iteration Planning method is not focused on assisting the product management in their market-facing responsibilities, but the relatively rapid, partial test releases do allow them to gather feedback during the project, which provides them accurate information about the market and the reception of the released features.

While the Release Iteration Planning method adopted in the case organization was based on the method described by Leffingwell [21], the Project α organization made several alterations to the method over the study period. Leffingwell [21] suggests that the last development sprint of each release iteration should be a so-called hardening sprint which is reserved for fixing any remaining bugs and for general quality assurance activities. However, hardening sprints were not scheduled in the release iterations of Project α. Instead, after Event 5 the project was shifted to a finalization phase where the last features were completed, and integration testing, verification, documentation and localization work was finalized.

Leffingwell [21] suggests that a draft plan review should be conducted at the end of the first day of a release planning event. In Project α, draft plan reviews were not conducted after Event 2. The draft plan reviews were abandoned because they were perceived to take too much time and also provide little value to the participants. Instead, short status check meetings were arranged regularly during the planning. Although the problem might have been that the development team members did not have enough experience of communicating their plans for other teams to find the best level of detail, we did not observe any issues caused by the lack of draft plan reviews. The informal communication between the participants and the frequent status checks conducted starting from Event 2 seemed to be enough to convey planning progress information between the participants. The communication of planning progress was further enhanced by the introduction of the planning matrix during Event 4.

Originally, the schedules of the Release Iteration Planning events in Project α did not explicitly include time for solving problems at the end of the first planning day. However, such a practice emerged. After the development teams had left for the day, the managers, Product Owners and Scrum Masters stayed behind to discuss and solve problems. The solutions and possible changes to plans were then presented to the development teams at the beginning of the second day. This practice is explicitly suggested

Table 4
Comparison of the characteristics of the release planning problem and the properties of the Release Iteration Planning method.

Release Planning Characteristic		Release Iteration Planning method
Shared understanding of requirements arises during the development and may be weak first [9]	⇐	Iterative process that takes into account new understanding from the previous iteration(s)
The values of the requirements selection criteria are time dependent [9]	⇐	Iterative process that allows reprioritization of requirements during the project
Great majority of requirements have complex dependencies between each other [9,26]	⇐	The whole project organization is present and can identify and solve dependencies efficiently
Decision makers have difficulties expressing how requirements should be prioritized [27]	⇐	The final plan is based on face-to-face discussions instead of a simple authoritative list of priorities
Feature development is implicitly prioritized higher than system improvement and innovation [27–29]	⇐	The prioritization between feature development and system improvement and innovation is explicit and forces the decision makers to justify their prioritization decisions

by Leffingwell [21] and, according to our study, is an effective way to solve problems during a Release Iteration Planning event.

According to Leffingwell [21], each development team should create release objectives which depict their overall goal for the next release iteration. In Project α, during Events 1 and 2, the teams were instructed to create release objectives. Forming meaningful release objectives seemed to be a very difficult task and the value of creating release objectives, in addition to user stories and features, was unclear. In the subsequent events the teams were not instructed to create release objectives. We did not observe any detrimental effects from leaving out the release objectives.

9.2. RQ2: What kind of benefits did the case projects gain from adopting the Release Iteration Planning method?

Table 5 summarizes the benefits we identified in the cases. Most of the benefits were related to the way the Release Iteration Planning method facilitated decision making. During the Release Iteration Planning events the participants could communicate directly which *improved communication and transparency*, *enabled inter-team coordination*, and allowed *fast decisions in the events*, *fast identification and management of dependencies* and *meeting people face-to-face and created project spirit*. The collaborative and interactive decision making can be considered to be the main benefit of the Release Iteration Planning method.

Since the most members of the project organization and most stakeholders were present in the same space, the managers could present *clear and unified goals for all stakeholders* and *show the importance of the project*. Such information sharing was clearly a major benefit of the method. In the Release Iteration Planning events, the managers were present and available during all planning events, which implicitly showed the importance of the project and allowed them to directly answer any questions regarding the goals of the project. This was clearly an improvement over simple project kick-off or project status update presentations often employed in traditional, plan-driven projects.

In addition to the communication-related benefits, the relatively rapid iterative nature of the Release Iteration Planning method allowed *fast recognition of the size and challenges of the project* and *enabled the successful finalization of the project*. These were clearly benefits over traditional, plan driven projects where the realization of the lateness of the project often comes late in the project. The rapid feedback from real users following each release is one of the proposed strengths of the agile software development methods [10]. Although only approximately half of the planned test releases were successfully completed during the case study period in Project α, these releases provided the Product Management feedback about the features and revealed the progress of the development in a very concrete way. This was clear improvement over the previous projects where the Product Management had little information on the development progress during the first six months of a development project.

9.3. RQ3: What kind of challenges did the case projects face in adopting the Release Iteration Planning method?

Table 5 summarizes the challenges we identified in the cases. We recognized only two challenges that were clearly caused by the way the method was used. At the beginning of Project α the external consultant and the managers expected that the features would be elaborated, prioritized and allocated during the events. However, it quickly became apparent that *lacking of preparation of the requirements before the events* and *lacking feature prioritization and allocation* were major causes of confusion and slow progress during the early planning events. The informal elaboration, prioritization and allocation of features are good examples of agile practices that have been claimed to work well in the bespoke, single team, single customer representative Scrum sweet spot [8], but created challenges in Project α due to the size and complexity of the project.

The pre-assignment of features to teams and the improved feature descriptions clearly reduced the overburden of managers and

Table 5
Benefits and challenges in the application of the method.

Benefits	Challenges
Clear and unified goals for all stakeholders	Lack of preparation of the requirements before the events
Fast recognition of the size and challenges of the project	Lacking feature prioritization and allocation
Improved communication and transparency	Lacking understanding of the creation of end-to-end features
Fast identification and management of dependencies	Challenges in learning inter-team coordination
Enabling inter-team coordination	Challenges in allocating architecture planning
Fast recognition and mitigation of impediments and risks	No unified understanding of the planning mindset
Fast decisions in the events	Leaving no slack in the plans
Meeting people face-to-face and creation of project spirit	Effort invested in the planning events
Showing the importance of the project	Over-optimism towards the new method
Enabling the successful finalization of the project	

sped up the planning events. We also observed that the pre-assignments were, indeed, considered tentative and the assignments of features to teams changed during the events when new information affecting the assignments came up. Initially, the prioritization of features was a challenge, as the Product Managers considered most of the features mandatory and equally valuable. Eventually, the product management started to produce prioritized lists of features, but explicit criteria for requirements prioritization or value was not given during any of the events. Thus, the user stories were split from the features and prioritized based on dependencies and on informal discussions between the teams, the Product Owners and the Product Managers, which required lots of time and created bottlenecks. Previous research has found that decision makers often have difficulties providing explicit criteria for requirements prioritization [27] and our results affirm those findings.

One solution proposed to the problem of splitting and prioritizing requirements in the agile software development literature is to employ the concept of the minimum marketable feature (MMF) [21]. For each feature, the minimum set of essential functionality that is valuable to the customers (i.e. the MMF) is first implemented. The rest of functionality is postponed until feedback from customers using the MMF can be employed to improve and prioritize the rest of the feature. Often customers are satisfied with the MMF and the development resources can be employed to implement other, more exiting features instead of extending the MMF. This solution was also considered in Project α, but not adopted during the study period.

Over-optimistic schedules are a well known problem in software projects [49]. This problem manifested in *leaving no slack in plans* and *over-optimism towards the new method* in Project α. In this case, leaving no slack in the plans was caused by the inexperience of the developers and by the pressure created by the managers who, in turn, had unrealistic expectations about the scope and schedule of the project. The Release Iteration Planning method was expected to reduce the planning overhead and thus speed up the project. However, the initial expectations were clearly over-optimistic. The true scope of the project was at least four times larger than what was expected in the beginning of the project. It is clear that finishing the project according to the original schedule and scope was not possible regardless of the release planning, or development, method used.

The case organization was clearly still in progress of transforming from a plan-driven line organization to an agile project organization. The front-end teams had been arranged as Scrum teams in the project previous to Project α, but it was the first Scrum project for the back-end teams. Many of the challenges we identified in Project α were caused by the immature adoption of the large-scale agile model and agile mindset.

The immature agile transformation caused *lacking understanding of the creation of end-to-end features and feature teams* and *challenges in learning inter-team coordination*. At the beginning of Project α, the Product Managers did not have enough experience creating end-to-end features that the teams could easily understand and implement, and the development teams did not understand that they were responsible for inter-team coordination of feature development. Both of these challenges alleviated during the project as the teams and Product Managers got more experience in the new agile development process and in the Release Iteration Planning method.

The transformation from a plan-driven process to an agile method requires a change of mindset of both developers and managers. The most obvious challenge caused by the traditional, plan-driven mindset was that the project organization had *no unified understanding of the planning mindset*. Although the introductory presentations in the planning events emphasized that the release plan should only be thought as a tentative starting point for further adjustments during the following release iteration, many managers and developers expected that the plan should be exactly executed during the following release iteration.

The role of software architecture and software architects in large-scale agile development is still an issue which does not have a clear solution [41]. In Project α the project organization had *challenges in allocating architecture planning* between the development teams and the software architects. This challenge alleviated over time as the architecture of the system stabilized and the teams became more experienced in developing the system. The teams would have benefited from more detailed architectural guidance at the beginning of the project.

Whether the *effort invested in the planning events* was a problem or not in Project α depended on the perspective it was viewed from. The whole development organization spent two or three days without developing any software, which can be viewed as waste of effort. On the other hand, the requirements, schedules and plans would have needed to be created and communicated in the project organization anyway. Effort spent on those activities was previously included in the daily work of the project organization and the release iteration method made the effort visible and calculable. Only true experiments could reveal if there is any significant difference between the effort spent on those activities in plan-driven and agile projects.

When the Release Iteration Planning method was introduced to Project β, the organization was already more mature regarding the agile transformation. Moreover, Project β was also much smaller than Project α. In Project β the Release Iteration Planning method seemed to function successfully without any major challenges.

9.4. Implications for theory

Although there have been claims to the contrary [29], a release planning method based on agile software development principles can be successfully applied in market driven product development and in large-scale projects. Release planning is not an unsolvable issue when agile software development methods are scaled up to large, multi-team projects. However, to accommodate the size of the project and to make the planning more efficient, the preparations to the release planning events must be more rigorous than in small, single-team agile development projects. Although release planning is a difficult problem [3,4,13], many of its problematic characteristics can be ameliorated by applying a planning method that is iterative and based on face-to-face discussions (see Table 4). Most of the previous research on software release planning has concentrated on authoritative, deterministic and plan-driven methods and tools [5,13]. Our results support the previous findings [4,9,12–14] that such tools have limited applicability, especially in agile software development projects.

According to the traditional project success criteria, that is, the deviations from the originally planned budget, schedule and scope are within acceptable range [23], Project α would be no doubt considered deeply challenged. One of the proposed benefits of the Scrum development method are the short sprints that enable all stakeholders to see the development progress at the end of each sprint in the sprint demo [8,50]. Our findings in Project α suggest that the Release Iteration Planning method enabled the project organization to quickly realize how unrealistic the original plan for Project α was. Our results suggest that an iterative and incremental release model combined with the Release Iteration Planning method may help project organizations to identify unrealistic project plans quicker than they could identify using the traditional, single release model.

van Waardenburg and van Vliet [11] identified challenges created by increased IT landscape complexity and by lack of business involvement when agile development methods were employed in

traditional, plan-driven context. The context of our study was somewhat similar to theirs. In Project α, especially the business-oriented stakeholders in the case organization were still in transition from a plan-driven mindset to the agile mindset. The Release Iteration Planning method implements many strategies that Waardenburg and Vliet identified for mitigating the challenges. We found that the Release Iteration Planning method enabled the meeting people face-to-face and creation of project spirit, provided the stakeholders with clear and unified goals and enabled inter-team coordination. These benefits match with the mitigation strategies of stimulating a common sense of purpose, end-to-end representation in team and intensive stakeholder communication. The Release Iteration Planning method also helped to show the stakeholders the importance of the project, improved communication and transparency and enabled fast decision making in the events. These benefits match with the mitigation strategies of changing business' mindset, channelling business knowledge and managing business-level alignment. This comparison suggests that the Release Iteration Planning method might be a powerful tool for mitigating challenges that rise when an agile development organization works with a plan-driven business organization.

9.5. Contributions to practitioners

We have described how a software development organization adopted the Release Iteration Planning method. Although our goal was not to provide a prescriptive guide for the Release Iteration Planning method, companies that are in comparable situation can employ our results as a starting point for creating their own release planning method. The benefits and challenges we have detailed help practitioners to identify whether this kind of method is suitable for their situation and to avoid or mitigate the challenges our case organization faced. We have also provided a summary of why release planning, in general, is difficult, and a description of characteristics that should be taken into account when a release planning method is considered.

9.6. Limitations and threats to validity

We employed both quantitative and qualitative methods for data collection and analysis. In the discussion of the validity and reliability of our results and analysis, we rely on the definitions proposed by Yin [46] and by Shadish et al. [51]. We discuss the validity of our research from four different aspects which are the internal validity, the construct validity, the external validity and the reliability [46]. The fourth type of validity, the statistical conclusion validity, is not relevant to this study as we do not employ statistical analyses to infer causal relationships [51].

Internal validity concerns the validity of the causal relationships observed in the case [46]. On several occasions we have suggested a causal relationship between a change made to the Release Iteration Planning method and a perceived improvement to the efficiency of conduct in the following planning events. Due to the limited scope of this research, we cannot completely rule out the possibility that the perceived improvements were caused by a confounding factor or by the maturation of the organization [51]. This is especially true in this case where no existing theory explains the causal relationships and no similar studies in other organizations exist.

In case study research, construct validity concerns how well the description of the cases represents the reality [46]. The post-project interviews were conducted approximately two years after the Release Iteration Planning events in Project α. In the post-project interviews, we asked the interviewees to recall what had happened during the project two years earlier. This may have decreased the validity of the interview data concerning the Release Iteration Planning events in Project α. However, we triangulated the interview data with data that was collected during the project. The construct validity of a case study can be increased by the triangulation of data sources, investigators, theories and methods [46–48]. Of these, we employed the investigator, method and data source triangulation. Three different investigators collected and analyzed the data. We employed three different research methods: observations, interviews and surveys. Our data sources included observation notes and recordings from the Release Iteration Planning events, interview recordings and quantitative and qualitative survey responses. Moreover, this manuscript was reviewed by two employees of the case organization who agreed that this manuscript presents a fair account of the two case projects.

The external validity of research concerns the domain to which the results of research are generalizable [46]. Both of the studied projects were carried out by the same company, which makes it difficult to identify the characteristics of a project or organization that are mandatory for our results to hold in other contexts. To summarize the main characteristics of the cases: the lifecycle-model was based on Scrum, multiple teams were working on the same software system and the development was market- and project-driven. The Release Iteration Planning method and the benefits and issues identified in this research are likely generalizable to projects that share the aforementioned characteristics. However, more studies of multi-team Scrum release planning in other organizations are required to truly assess the generalizability of our results.

The reliability of a case study concerns whether different researchers had produced the same results if they had studied the same projects [46]. The main threat to the reliability is the variability in the data collection. The observation notes written during the Release Iteration Planning events were an account of the observations the researchers found noteworthy and different researchers might have noted other things. However, two researchers wrote separate observation notes which increased the reliability of the observations. Discussions and meetings during the events were also recorded whenever it was possible. These recordings provide an additional account of the Release Iteration Planning events and increase the reliability of our results. The post-project interviews provided third account of the Release Iteration Planning events. This data source triangulation makes our results robust against threats to reliability [46–48].

Most of the data we collected converged between the investigators, methods and data sources and revealed no notable threats to the construct validity or reliability of our results. Triangulation revealed one significant point of divergence between different data sources, which was the cost-efficiency of the Release Iteration Planning method. Further analysis revealed that the different stakeholders had different opinions on the cost-efficiency of the method. Additional quantitative financial and effort data would have been required to analyze the cost-efficiency of the method. Since such data was not collected by us or by the case organization, we cannot reach a conclusion regarding the cost-efficiency of the method, which is a clear limitation of our study.

Project α was unique in the case company as the scope of the project and the size of the development organization were larger than ever before in the case company. Without a point of comparison, we cannot say what kinds of effects the Release Iteration Planning method had to the success of the project or if the traditional, plan-driven project management approach would have been better. Thus, the conclusions on the effects of the method to the success of the project are limited to observed benefits from the method we have disseminated in this article.

9.7. Future work

Our in-depth case descriptions allow other researchers of the field to compare and contrast their findings to our results and anal-

ysis. Building evidence based and solid theories of agile software development has been identified as an important goal for the research field [41]. The long-term goal of the agile release planning research should be the building of an overall theory of agile release planning, and our study is a small step towards that goal.

10. Conclusion

We studied a case organization that had faced challenges with release planning and consequently adopted a new release planning method which we call the Release Iteration Planning method. The method was based on the agile software development principles of direct and efficient communication between the stakeholders of the project and iterative and incremental development. This article presents the first extensive scientific study of the Release Iteration Planning method.

We studied two projects of the case organization that adopted the Release Iteration Planning method. We described how the method was applied and improved over a twelve-month time period, what kinds of benefits the method brought to the projects and what kinds of challenges the projects faced when applying the method. Although the first project faced several challenges at the beginning, the method, adopted from Leffingwell [21] and subsequently altered to suit the large organization better, contributed to the eventual success of both projects.

We identified the following ways the method ameliorates the difficult characteristics of the release planning problem: the communication between the development organization and the Product Management enabled by the events allows both of them to better understand the requirements from the business and the technical points of view. The iterative and incremental releases enable frequent feedback from the users and customers and allow the reprioritization of the requirements if their value changes over time. The face-to-face communication in the planning events enables fast dependency identification and management. The Product Management shares the responsibility for the final feature prioritization with the development organization instead of providing a simple authoritative list of prioritized features. The prioritization between feature development and system improvement and innovation is explicit and can be discussed in the planning events, which prevents excessive focus on the short term feature development tasks.

The market facing product management activities are important in market driven software product development. However, our final conclusion is that release planning in agile software development organizations requires collaborative effort between the market facing stakeholders and the development organization.

Acknowledgements

We would like to thank F-Secure Oyj for making this study possible and all the anonymous interviewees for providing valuable contributions to this research.

This work was supported by TEKES as part of the Cloud Software Finland and the Need for Speed research programs of DIGILE (Finnish Strategic Centre for Science, Technology and Innovation in the field of ICT and digital business).

Appendix A. 2012 Interview questions

A.1. Project α specific questions

- What was your role during Project α and before it?
- How did you find out about the Release Iteration Planning method?
- Why did you decide to try the method?
- How did you try to perform the planning before adopting the method?
- Were there some specific challenges you tried to solve with the method?
- Did you have any training in the method?
- Did you have a specific process to improve the Release Iteration Planning method?
- When did you start to adopt the Scrum model?
- What was the release cycle of the product?
- How did you prepare for the first Release Iteration Planning event?
- How do you think the first event went in general?
- Do you think the later [compared to Event 1] events were better?
- Who belonged to the project organization?
- How many teams there were at different times in Project α?
- What was the organization of the development teams?
- How was product management organized in Project α?
- Did every team have a dedicated Product Owner?
- Did every team have dedicated Scrum Master?
- Where did the Scrum Masters come from?
- Did the small number of Product Owners compared to the teams cause any problems?
- Did the Malaysian teams have Product Owners?
- Is your organizational culture very hierarchical?
- What caused the friction between the product management and the development organization?
- Was writing requirements in the feature format done before?
- How did you prioritize features in the beginning of Project α?
- Did features typically have dependencies between them?
- Did you measure how much effort went into the preparation for the first Release Iteration Planning event?
- What kind of training you had before the first event?
- Did you have trouble getting approval for this method from the upper management?
- Would you do things identically if you had similar project [to Project α] starting now?
- Did you try to create cross-cutting features in the beginning [of Project α]?
- Did each team have a certain component they knew best?
- Were the same teams involved in the previous project?
- How did you communicate the feature development plans to the teams in the previous project?
- How did you monitor the progress of development in the previous project?
- Can you think of anything that was especially good in the Release Iteration Planning method?
- Do you think the planning was successful?
- Do you have any improvement suggestions [to the method]?
- Can you think of any other downsides of the method?
- Do you think there is an upper size limit when the method stops working well?
- Would you recommend this method to others?
- Did you see or read the feedback reports we sent to the company?
- How many teams were from Poland and how many from Malaysia after [Event 5]?
- How did you accommodate the distributed teams in the planning events?
- If you would now conduct this kind of planning event, would you like to have all teams present or is it enough to have a videoconference?
- Why do you think the management did not initially realize the project would be impossible to complete in the given time?
- What happened in the project after [Event 5]?

- Why did the project stop having the Release Iteration Planning events after [Event 5]? When did it happen? Why did it happen?
- When did you complete everything that was planned for the product?

A.2. Project β specific questions

- What was your role in Project β?
- What is your role in the organization?
- When was the Release Iteration Planning method adopted in Project β?
- How did you, concretely, initiate the adoption of the method?
- In how many Release Iteration Planning events you have been in?
- How many Release Iteration Planning events there has been in Project β?
- Was the Russian site involved in the project from the beginning?
- Did the Russian site develop the previous version of the software?
- Why did you take the Release Iteration Planning method into use?
- Where did you hear from about the Release Iteration Planning method?
- What had you heard about the Release Iteration Planning method?
- Before adopting the Release Iteration Planning method, how did you try to show the developers the big picture?
- Did you have any forum where you [the PO and developers] discussed?
- What was the typical schedule in the Release Iteration Planning events?
- Who facilitated the Release Iteration Planning events?
- On what precision level you try to predict the feature development schedule?
- How many POs there are in [Project β]?
- How do product managers collaborate with the Russian site?
- Why does the frontend backend division exist?
- Are the teams split between the frontend and backend?
- How many teams do you have [in Project β]?
- Do all teams work solely on [Project β]?
- How are features brought to the Release Iteration Planning events?
- Is backlog grooming performed by each team individually?
- Are features specific to the frontend or the backend?
- How do you handle the coordination between the frontend and backend when features are end-to-end?
- On what abstraction level are your backlog items?
- Is there an intermediate step between features and user stories?
- When are product demonstrations given?
- What is the typical length of the Release Iteration Planning events?
- Is the goal to get features done over single release iteration? Has it been successful?
- Is the backlog of the Russian teams electronic?
- Do you have status meetings during the Release Iteration Planning events?
- Does every team have their own Scrum Master?
- Is it typical that everyone gathers together at the end to review the plan? Is this a good way to review the plan?
- Can you think of anything especially good in the Release Iteration Planning method compared to the previous planning method?
- Do you think that the developers have courage to speak about issues when you and other managers are present in the events?
- Is the product architect from [Russia] or from [Finland].
- Do you have a team of architects that thinks about the architecture?
- Do you have any dependencies to other products of the company?
- How do you manage the dependencies?
- Have you noticed if the teams in the Release Iteration Planning events talk to each other?
- Did you pre-assign backlog items to the teams?
- From your point of view, what are the things that could be improved [in the Release Iteration Planning method]?
- Do you think the planning has been successful?
- Do you have any problems related to the Release Iteration Planning in Project β?
- Would you recommend this method to others?
- Do you know why the method is not used in other parts of the company?

References

[1] N.D. Fogelström, T. Gorschek, M. Svahnberg, P. Olsson, The impact of agile principles on market-driven software product development, J. Softw. Maint. Evol.: Res. Practice 22 (1) (2010) 53–80.

[2] T. Chow, D.-B. Cao, A survey study of critical success factors in agile software projects, J. Syst. Softw. 81 (6) (2008) 961–971.

[3] A. Ngo-The, G. Ruhe, A systematic approach for solving the wicked problem of software release planning, Soft Comput. – Fusion Found., Methodol. Appl. 12 (1) (2008) 95–108.

[4] P. Carlshamre, Release planning in market-driven software product development: provoking an understanding, Requirements Eng. 7 (3) (2002) 139–151.

[5] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S.B. Saleem, M.U. Shafique, A systematic review on strategic release planning models, Inform. Softw. Technol. 52 (3) (2010) 237–248.

[6] A. Al-Emran, D. Pfahl, Operational planning, re-planning and risk analysis for software releases, in: J. Münch, P. Abrahamsson (Eds.), Proceedings of the 8th International Conference on Product-Focused Software Process Improvement (PROFES 2007), Lecture Notes in Computer Science, vol. 4589, Springer, Berlin Heidelberg, 2007, pp. 315–329.

[7] G. Ruhe, J. Momoh, Strategic release planning and evaluation of operational feasibility, in: Proceedings of the 38th Annual Hawaii International Conference on System Sciences (HICSS '05), 2005.

[8] K. Schwaber, M. Beedle, Agile Software Development with Scrum, Prentice-Hall, Upper Saddle River, NJ, 2002.

[9] H.C. Benestad, J.E. Hannay, A comparison of model-based and judgment-based release planning in incremental software projects, in: Proceeding of the 33rd International Conference on Software Engineering (ICSE'11), ACM, New York, NY, USA, 2011, pp. 766–775.

[10] A. Cockburn, Agile Software Development, Addison-Wesley, Boston, MA, 2002.

[11] G. van Waardenburg, H. van Vliet, When agile meets the enterprise, Inform. Softw. Technol. 55 (12) (2013) 2154–2171.

[12] L. Cao, B. Ramesh, Agile requirements engineering practices: an empirical study, Software 25 (1) (2008) 60–67.

[13] S. Jantunen, L. Lehtola, D.C. Gause, U.R. Dumdum, R.J. Barnes, The challenge of release planning, in: Proceedings of the Fifth International Workshop on Software Product Management (IWSPM 2011), IEEE, Piscataway, NJ, USA, 2011, pp. 36–45.

[14] B.W. Boehm, Requirements that handle IKIWISI, COTS, and rapid change, Computer 33 (7) (2000) 99–102.

[15] VersionOne, Inc, 7th Annual State of Agile Development Survey, 2013. <http://www.versionone.com/pdf/7th-Annual-State-of-Agile-Development-Survey.pdf> (cited May 2013).

[16] S. Freudenberg, H. Sharp, The top 10 burning research questions from practitioners, IEEE Softw. 27 (5) (2010) 8–9.

[17] M. Paasivaara, C. Lassenius, Scaling Scrum in a large distributed project, in: Proceedings of the International Symposium of Empirical Software Engineering and Measurement (ESEM 2011), 2011, pp. 363–367.

[18] D. Leffingwell, Scaling Software Agility: Best Practices for Large Enterprises, Addison-Wesley Professional, Reading, MA, 2007.

[19] C. Larman, B. Vodde, Scaling Lean & Agile Development: Thinking and Organizational Tools for Large-scale Scrum, Addison-Wesley, Upper Saddle River, 2009.

[20] L. Lehtola, M. Kauppinen, S. Kujala, Requirements prioritization challenges in practice, in: F. Bomarius, H. Iida (Eds.), Proceedings of the Product Focused Software Process Improvement conference (PROFES 2004), Lecture Notes in Computer Science, vol. 3009, Springer, Berlin Heidelberg, 2004, pp. 497–508.

[21] D. Leffingwell, Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise, Addison-Wesley, Upper Saddle River, NJ, 2011.

[22] E. Ferrari, Product Management for Software, Mondo Strategies Press, 2008.
[23] H.-B. Kittlaus, P.N. Clough, Software Product Management and Pricing: Key Success Factors for Software Organizations, Springer, Berlin, 2009.
[24] V. Heikkilä, K. Rautiainen, S. Jansen, A revelatory case study on scaling agile release planning, in: Proceedings of the 36th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), IEEE Computer Society, 2010, pp. 289–296.
[25] G. Gunyho, J.G. Plaza, Evolution of longer-term planning in a large scale agile project – F-Secure's experience, in: A. Sillitti, O. Hazzan, E. Bache, X. Albaladejo, W. Aalst, J. Mylopoulos, M. Rosemann, M.J. Shaw, C. Szyperski (Eds.), Agile Processes in Software Engineering and Extreme Programming, Lecture Notes in Business Information Processing, vol. 77, Springer, Berlin Heidelberg, 2011, pp. 306–315.
[26] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, J. Natt och Dag, An industrial survey of requirements interdependencies in software product release planning, in: Proceedings of the IEEE International Conference on Requirements Engineering (RE '01), IEEE Computer Society, Piscataway, NJ, USA, 2001, pp. 84–91.
[27] S. Barney, A. Aurum, C. Wohlin, A product management challenge: creating software product value through requirements selection, J. Syst. Architect. 54 (6) (2008) 576–593.
[28] M. Lindgren, C. Norström, A. Wall, R. Land, Importance of software architecture during release planning, in: Proceedings of the 7th IEEE/IFIP Working Conference on Software Architecture (WICSA 2008), 2008, pp. 253–256.
[29] N.D. Fogelström, M. Svahnberg, T. Gorschek, Investigating impact of business risk on requirements selection decisions, in: Proceedings of the 35th Euromicro Conference on Software Engineering and Advanced Applications (SEAA '09), IEEE Computer Society, Piscataway, NJ, USA, 2009, pp. 217–223.
[30] G. Ruhe, A. Ngo, Hybrid intelligence in software release planning, Int. J. Hybrid Intell. Syst. 1 (1–2) (2004) 99–110.
[31] J. Momoh, G. Ruhe, Release planning process improvement – an industrial case study, Softw. Process: Improv. Pract. 11 (3) (2006) 295–307.
[32] V. Heikkilä, A. Jadallah, K. Rautiainen, G. Ruhe, Rigorous support for flexible planning of product releases – a stakeholder-centric approach and its initial evaluation, in: Proceedings of the 43th Hawaii International Conference on System Sciences (HICSS '10), IEEE Computer Society, 2010.
[33] M.O. Saliu, G. Ruhe, Bi-objective release planning for evolving software systems, in: Proceedings of the 6th Joint Meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on The Foundations of Software Engineering (ESEC-FSE '07), ACM, New York, NY, USA, 2007, pp. 105–114.
[34] K. Logue, K. McDaid, Agile release planning: dealing with uncertainty in development time and business value, in: Proceedings of the 15th IEEE International Conference and Workshops on the Engineering of Computer-Based Systems, IEEE, Piscataway, NJ, USA, 2008, pp. 437–442.
[35] M. Li, M. Huang, F. Shu, J. Li, A risk-driven method for eXtreme programming release planning, in: Proceedings of the 28th International Conference on Software Engineering (ICSE '06), ACM, New York, NY, USA, 2006, pp. 423–430.
[36] A. Al-Emran, K. Khosrovian, D. Pfahl, G. Ruhe, Simulation-based uncertainty analysis for planning parameters in operational product management, in: Proceedings of the 10th International Conference on Integrated Design and Process Technology (IDPT-2007), Society for Design and Process Science, USA, 2007, pp. 191–201.
[37] S.A. Wheelan, Group size, group development, and group productivity, Small Group Res. 40 (2) (2009) 247–262.
[38] D. Rodríguez, M.A. Sicilia, E. García, R. Harrison, Empirical findings on team size and productivity in software development, J. Syst. Softw. 85 (3) (2012) 562–570.
[39] K. Schwaber, The Enterprise and Scrum, Microsoft Press, Redmond, WA, 2007.
[40] M. Cohn, Agile Estimating and Planning, Prentice Hall Professional Technical Reference, Upper Saddle River, NJ, 2005.
[41] T. Dingsøyr, S. Nerur, V. Balijepally, N.B. Moe, A decade of agile methodologies: towards explaining agile software development, J. Syst. Softw. 85 (6) (2012) 1213–1221.
[42] V.T. Heikkilä, M. Paasivaara, C. Lassenius, C. Engblom, Continuous release planning in a large-scale Scrum development organization at Ericsson, in: H. Baumeister, B. Weber (Eds.), Agile Processes in Software Engineering and Extreme Programming, Lecture Notes in Business Information Processing, vol. 149, Springer, Berlin Heidelberg, 2013, pp. 195–209.
[43] K. Vlaanderen, S. Jansen, S. Brinkkemper, E. Jaspers, The agile requirements refinery: applying SCRUM principles to software product management, Inform. Softw. Technol. 53 (1) (2011) 58–70.
[44] C. Larman, B. Vodde, Practices for Scaling Lean & Agile Development: Large, Multisite, and Offshore Product Development with Large-scale Scrum, Addison-Wesley, Upper Saddle River, NJ, USA, 2010.
[45] J. Rothman, Manage it!: Your Guide to Modern, Pragmatic Project Management, The Pragmatic Bookshelf, Raleigh, NC, 2007.
[46] R.K. Yin, Case Study Research: Design and Methods, fourth ed., Sage Publications, Thousand Oaks, CA, 2009.
[47] T.D. Jick, Mixing qualitative and quantitative methods: triangulation in action, Adm. Sci. Quart. 24 (4) (1979) 602–611.
[48] M.Q. Patton, Qualitative Research and Evaluation Methods, third ed., Sage Publication, Inc., Thousand Oaks, CA, 2002.
[49] K.E. Emam, A.G. Koru, A replicated survey of IT software project failures, IEEE Softw. 25 (5) (2008) 84–90.
[50] K.H. Pries, J.M. Quigley, Scrum Project Management, CRC Press, Boca Raton, FL, USA, 2011.
[51] W.R. Shadish, T.D. Cook, D.T. Campbell, Experimental and Quasi-experimental Designs for Generalized Causal Inference, Houghton Mifflin, Boston, MA, USA, 2001.