A Revelatory Case Study on Scaling Agile Release Planning

Ville Heikkilä, Kristian Rautiainen
School of Science and Technology
Aalto University
Helsinki, Finland
{*firstname.lastname*}*@soberit.hut.fi*

Slinger Jansen
Department of Information and Computing Sciences
Utrecht University
Utrecht, Netherlands
S.Jansen@cs.uu.nl

Abstract

A way to scale up agile release planning to meet the requirements of multi-team agile development is a practice called joint release planning. A software product company piloted the joint release planning method. The aim of the company was to improve coordination of work of multiple agile development teams who develop a large legacy software product. Another aim was to improve communication between product management and development. We conducted a case study to explore how the new release planning method was executed. We gathered data by observing two release planning events, observing event planning meetings, and by conducting surveys. The events were attended by approximately 140 stakeholders, including over 10 development teams, who spent several days in a common space. The participants liked the method and considered it efficient. This revelatory case study provides the first detailed empirical description of this emerging method for multi-team agile release planning.

1. Introduction

Planning the next product release is recognised to be one of the most challenging parts of market-driven product development [7] and a critical success factor in agile software development projects [6]. The main goal of release planning is to find an appropriate scope for a release while taking into account budget, resource, technical, and other constraints [7, 10].

Joint release planning has recently emerged in the industry for multi-team agile release planning for complex systems [9]. Similarly to the single team agile release planning, the basic idea of the joint release planning method is to gather all development teams and internal stakeholders in a single space to perform release planning together. However, the sheer number of people, requirements and dependencies makes joint release planning difficult to perform efficiently. Scaling the agile release planning up to a multi-team environment where many teams are developing the same product at the same time also introduces technical complexity. The teams cannot plan releases in isolation, since requirements are selected from the same product backlog and coordination of who-does-what is required. In an ideal agile world all requirements are independent and can be implemented in isolation. In the real world there are often architectural complexities which result in a network of dependencies between requirements [5]. This also affects the implementation order of the requirements.

In this paper we report findings from a company that adopted the joint release planning method. The introduced release planning method was chosen in order to better coordinate multiple agile teams simultaneously developing a large legacy software product and to improve communication between product management and development. We conducted a case study to explore how the joint release planning method was executed and accepted in the case organization. The joint release planning method in question or any empirical evidence regarding it has not been described in any scientific publication the authors know of. This case study is a *revelatory study* [18] of the joint release planning method. Our goal in this paper is to provide the first detailed empirical account of the use of the method in a multi-team development project in a software product company.

2. Related work

The majority of publications on software release planning focus on different kinds of mathematical models and simulations designed to create most optimal or risk free release plans when key parameters such as resource availability, value of requirements, development effort and risk factors are known or can at least be estimated somewhat accurately [16]. The model or simulation is then used to generate one optimal release plan or a set of near-optimal release plans. Such models are called strategic release planning models [13].

The validity of most of the strategic release planning models in large scale industrial setting is questionable. Svahnberg et al. [16] reviewed 22 strategic release planning models. Only four of those models had been validated in large scale industrial use. In addition, the model-driven approach to release planning has proven to be problematic in practice, as many software companies do not have a software development process which could be relied on to record or generate required key parameters [4] and often requirements change so frequently that any long term plan quickly becomes obsolete [2]. Agile software development methods claim to mitigate these issues by not creating detailed long term plans and by adapting to changing requirements and priorities when needed [1].

Release planning in the agile software development context has only been slightly addressed in scientific literature. The little existing literature applies the plan-driven release planning approach in agile software development context, and usually without taking into account that following a highly detailed release plan is not considered necessary or even useful in agile software development (see e.g. [3, 8]).

The Scrum process model [14] defines the product owner role, whose responsibility is managing the development of a product. The agile release planning process in a single-team single-product development scenario is simple. The team and a product owner discuss the features that could be included in the next release until an agreeable plan is reached. The agreed-upon release plan then acts as a vision for planning the individual iterations [14, 15, 12].

3. Research methodology

Release planning of a new version of a product that belongs to an existing product line was studied in this longitudinal case study [18]. Two successive joint release planning events of the next product release were observed. The first event was an initial joint release planning event (called *Event A* hereafter) and the second was a release re-planning event conducted eight weeks later (called *Event B* hereafter).

3.1. Case background

The agile software development process in the case company is based on Scrum [14]. Except for the parts that are directly related to describing the release planning events, a more precise description of the case company's development and product management model is out of the scope of this paper. Also, a broader description of the case company is omitted for confidentiality reasons.

In the case company's release management model, requirements management is based on a five-level hierarchy introduced by Leffingwell [9]. *Strategic themes* denote strategic focus areas for the company's business. Within these, *epics* form high-level functional goals for the product(s). Epics can be split into *features*, which in turn can be further split into *user stories*. Finally, user stories are refined into development *tasks*, which denote what needs to be done technically to implement a user story.

The plans for a release project can be seen on two time horizons. Roadmaps, which contain epics and features, show the tentative content of future releases. The next upcoming release can be split into *potentially shippable increments (PSIs)*, which are internal releases of the product containing a subset of the intended features of the final, external release of the product. The joint release planning concentrates on planning the next PSI of a product release. The work during a PSI is done in iterations, which are planned with detailed tasks. Together these levels of planning form a rhythm for product development (aka *cadence*). This way of planning follows the principle of planning long term with abstract requirements and short term with detailed requirements [11, 17].

The release project in the case company was planned to last for approximately six months and contain two PSIs. The first PSI was planned to contain seven two-week iterations. The main purpose of the first joint release planning event (Event A) was the planning of the next PSI. Since the joint release planning method was new to the case company and there were major technical changes in the product architecture, a release re-planning event (Event B) was scheduled to follow the fourth (hardening) iteration. Event B was conducted as a normal joint release planning event where the scope was the rest of the PSI and planning was done taking into account the progress of the PSI so far. Figure 1 shows the case company's planned release project cadence.

The release project organization we studied in the case company consists of over 10 development teams, a user experience team, a system team, and a product management team including a release project manager. Each development team consists of 6-8 developers and has a scrum master and a product owner as additional members. However, all teams do not have a dedicated scrum master or product owner. The system team is a specialized group that is responsible for system level quality assurance, development infrastructure and continuous improvement of development practices. The product management team is responsible for the operational management of the release project.

3.2. Data collection

Two researchers observed both release planning events. The researchers wrote separate observation notes during the events. Two voice recorders were used to record presentations and different discussions. Also, a survey was conducted after each event. The description of the case in this paper is based on those observations, notes, recordings, and survey results. After Event A the researchers compiled a

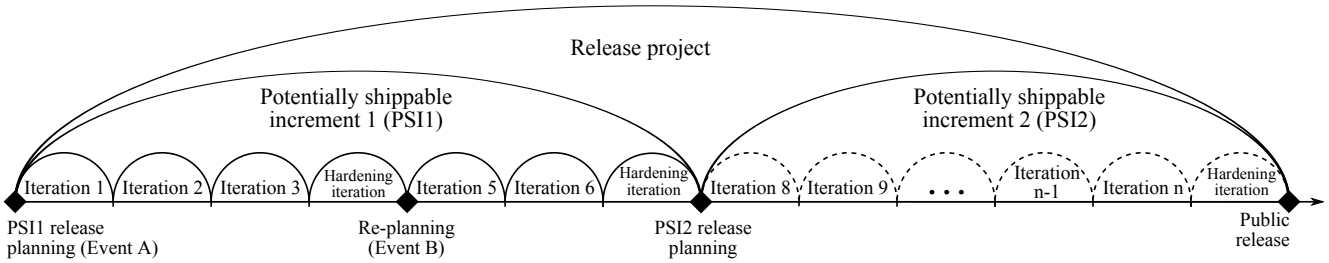


Figure 1. Case company development cadence

report for the case company. The report contained observed weaknesses in the process and improvement suggestions. The researchers also participated in a preparatory meeting for Event B, where the survey results, the report, and changes to the planning process were discussed with release project stakeholders.

A survey was conducted after Event A to gather opinions on the planning method and planning success from the participants. The survey contained questionnaire statements on a six-point Likert-like scale with items "strongly disagree" (1), "disagree" (2), slightly disagree" (3), "slightly agree" (4), "agree" (5) and "strongly agree" (6). In addition, the item "not applicable / cannot say" was provided. The survey also contained free-text fields for textual feedback, a question to grade the event on a scale from one to six and questions for gathering demographic information.

A second survey was conducted after Event B. It contained a subset of statements from the first questionnaire and additional statements regarding the new practices taken into use in Event B. All participants of the events were invited to answer the questionnaires. The Event A survey was sent to 140 participants and answered by 33 respondents, and the Event B survey was sent to 136 participants and answered by 26 respondents. The questionnaires were conducted with an online survey software and they were completely anonymous to increase the validity of the results.

4. Results

4.1. Overview of the joint release planning events

During both events, most members of all development teams were in present. Three of the development teams were offshored and had only three representatives present at the release planning sessions. In addition to the development teams, a user experience team representative, the system team, several representatives of the product management team and several other internal stakeholders with dependencies to the product were present in the events. The total number of participants was approximately 140 in both events.

The release planning facility had a separate area for the different stakeholder presentations and for the teams to present their plans. Team breakout areas were separated by movable walls which acted as planning boards and dampened noise, but also provided easy access between the different areas. In Event B an area was reserved for product managers, architects, and other internal stakeholders where they could be found when they were not with a team during team breakout sessions.

A facilitator had an important role in the joint release planning events. The facilitator was responsible for making sure that the event was proceeding as planned and within schedule, and solving conflicts and impediments that rose during the events. A process consultant facilitated Event A and the preceding training day, and acted as an advisor for the project manager who facilitated Event B.

Because the joint release planning method was previously unfamiliar to the case company, a training session was conducted a day before Event A started. During the training session the facilitator gave presentations on general principles of agile development and on how to perform the release planning.

The joint release planning events in the case company followed the general structure presented in Figure 2. The events started with a presentation of the overview of the event. In Event B it was followed by a role-specific project retrospective. In Event A the project retrospective was not held, since the project had not yet started. Different stakeholders then presented their visions for the product. Next, instructions on how to perform the planning were given by the facilitator. After the presentations the development teams started planning the release in team breakout sessions. During Event A the breakout sessions ended in collective status presentations before lunch and at the end of the day. During Event B there were hourly status checks in addition to end-of-the-day status presentations.

At the end of the events, the teams' plans were collectively reviewed and open risks were processed. As a final step, an event retrospective was held to improve the planning method. Each step of the event structure is described in more detail in the following sections.

The case company's planning events span over several days. Event A was originally scheduled to take two days with one day in reserve in case the planning could not be completed in two days. After the first actual planning day the facilitator and the project manager came to the conclu-

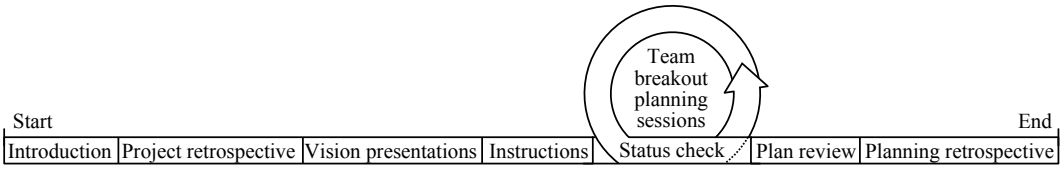


Figure 2. A generic timeline of the joint sprint planning events (not in scale)

sion that the planning needed to be extended into the third day. Event B was originally scheduled to take one and a half days with the rest of the second day in reserve. However, the project manager decided to extend Event B to take the whole second day before the event had started.

4.2. Presentations and project retrospective

Event A began with an introductory overview of the release project. The topics covered were project background, product positioning in the market, target customers, overview of the competition, project organization and project cadence. In addition an overview of the event schedule was given. In Event B, the presentation was shorter and contained only changes to event practices and schedule.

In Event A, the introduction was followed by the first vision presentation. It was given by the head of product management. He presented the product's vision and business themes and the epics for the whole release project as well as the ones intended for PSI1. Several representatives from different specialized teams and other internal stakeholders gave their own vision presentations: a product architecture manager, a user experience team representative, a research department representative and a system team representative. Since the development of the new version of the product was initiated in Event A, it also acted as a project kickoff. Handouts of each presentation and additional in-depth information on the topics were distributed to all participants.

In Event B, the introduction was followed by a release project retrospective. The purpose of the release project retrospective was to find and solve impediments and learn from the good and bad practices observed during the release project so far. The retrospective was performed in role-based teams. The different teams were software engineers, test engineers, architects, scrum masters, and product management. Each team had a named facilitator selected from the team. The teams were free to conduct the retrospective as they preferred, but they were requested to produce a briefing of corrective actions to present for the other teams and describe one or two things that went well or did not go well since the beginning of the release project.

The retrospective was followed by a presentation of the requirements for the rest of PSI1. One notable change from Event A was that product managers had prioritized features and created preliminary team assignments for them. Next, a representative from each development team gave a short presentation on the team's progress in the release project so far. There was no mandatory format to show the progress information, and nearly all presentations presented the information in a different way. The development team presentations were followed by a presentation on overall status of the project and by a short system architecture status update presentation.

4.3. Instructions

Event A release planning instructions contained several practical issues: the different colors of sticky notes to use for recording user stories, dependencies, objectives, and risks, as well as recommendations for writing requirements as user stories and to split user stories if they are too large to fit in an iteration. The user stories were recommended to contain the user or customer of the story, what he or she needs to accomplish, and motivation for the action. The teams were instructed to write high level objectives for the whole PSI based on the user stories included in it. At a philosophical level, the teams were instructed to plan according to the simplest thing that would probably work in the next product release. The teams were also instructed to start the breakout sessions by discussing the product vision and epics assigned to the team with product owners. Shorter instructions with similar contents were given in Event B.

4.4. Team breakouts

In general, the way teams worked during most breakout sessions in both events was quite uniform. They sat or stood around a table in their designated area and discussed how a feature should be split. Most of the teams did not write user stories from a user's point of view. Instead they split features into large technical tasks. During Event A they were repeatedly instructed by the facilitator to write user stories, but we observed no changes. User stories were not usually estimated immediately after creation, and scheduling was done by first putting stories in implementation order and then estimating and dividing them into iterations based on the team's projected velocity. There was one breakout session during day one of Event A, two breakout sessions during day two and one breakout session during day three. In Event B, there was one breakout session during day one and two breakout sessions during day two.

The first team breakout session of Event A started in the afternoon of the first day. Everything was new for the

teams, from the planning process itself to the new architecture planned for the product. The planning breakout session appeared disorganized and non-productive. The teams seemed to have problems to understand which features they were supposed to plan for. Product management together with architects decided to prepare pre-assignments of features for the teams for the second day of Event A. The pre-assignments were given in the beginning of the second planning day of Event A and they seemed to help most of the teams to get the planning on track. Based on our observations, they seemed to also work fairly well in Event B. However, based on the free-text feedback from the questionnaires, the teams would have preferred to receive the in-depth planning material (including feature pre-assignments) well in advance.

The teams did not have clear priorities for the features in Event A. The problem became evident especially for one team (α), who started planning one feature the first day of Event A. The feature was then changed to another feature the second day, until product management noticed that one very important feature had not been included in any teams' planning. This feature was then assigned to Team α to be planned during the third day and all previous plans Team α had prepared were discarded. In Event B, materials included a priority ordered list of features to be planned, which seemed to alleviate the problem of changing priorities.

In the beginning of the breakout sessions in Event A the teams did not immediately start discussing with each other, even when they noticed dependencies between teams. We discussed this observation with the facilitator. The facilitator had instructed that the teams should write the dependencies on sticky notes and they would then be discussed in the status checks. However, his original intent was that teams would only write dependencies that could not be immediately resolved in the team or between the participants. The following day the instructions were made clearer, and we observed much more interaction between the teams. This was considered one of the best things about the joint release planning in Event A questionnaire free-text feedback. In Event B, the interaction started immediately from the first day and continued throughout the whole event.

In Event A, the teams would have needed more support and guidance from all the stakeholders than was available. More preparatory work was done by the architects for Event B. They had prepared views of the overall architecture, which were actively used by the teams for planning during Event B. Also, product management had a so-called base station where the representatives could be easily found if they were not working with a team. One team added a new type of sticky note called "help wanted" to their wall to signal their needs to passing stakeholders and to act as reminders in the status checks.

In an Event B preparatory meeting the case company's product management representatives reported that it was difficult to keep track on who was planning what during the breakout sessions, and whether everything was getting proper attention. To allievate this in Event B, product management had given unique identifiers to the features and brought less features to the planning. The teams were instructed to always make a reference from the stories they created to the associated features, both on the sticky notes and in the status checks. No problems regarding this practice were neither reported in the feedback questionnaire nor observed during Event B.

4.5. Status checks

The purpose of status checks was to coordinate the planning work of the teams and to make the progress of planning visible to stakeholders. In Event A each team had a 4-minute time box in which they shortly described what was on their wall and how their plan contributed to the overall goals (features) of the PSI. The short time box was enforced to limit the length of the status checks which, nevertheless, took more than an hour each. The walls had wheels and they were rolled in front of the event participants who were gathered in the presentation area for the status check. The participants were encouraged to ask questions and comment the presentations. There was one status check presentation at the end of the first day and two status check presentations during the second day of Event A.

According to the questionnaire free-text feedback, the feelings about frequent status checks during Event A were mixed. On the one hand there was positive feedback about being able to see and listen to what the other teams had planned. But on the other hand some stated that planning was interrupted too often by the status checks. Based on our observations, some problems were uncovered and resolved too late in Event A because status checks were too infrequent. This resulted in seemingly unnecessary re-planning for some teams.

In order to limit the disturbance to the teams, a scrum-of-scrums practice was introduced in Event B. In scrum-of-scrums, one representative from each team, typically the scrum master, presented the status of the team's planning to the other teams' representatives. Meanwhile, the rest of the team members kept on planning. Scrum-of-scrums was conducted once an hour during the team breakout sessions and seemed to work well. According to the questionnaire conducted after Event B, it was also well accepted: statement 6 ("Scrum of Scrums status check was a useful practice") (see Table 2) got mode answer of "agree" and median of 5.0.

The information gained from status check presentations and scrum-of-scrums was used in coordinating the remainder of the planning events. After each status check the sur-

Table 1. Ratings of the planning events

	Event	
Rating	A (n = 33)	B (n = 26)
6 (excellent)	5	1
5	17	10
4	9	13
3	1	2
2	0	0
1 (poor)	1	0
Median	5.0	4.0

faced impediments and dependencies were discussed in a group consisting of the facilitator of the event and representatives from different stakeholder groups. Whatever solutions for resolving the impediments and dependencies were figured out, they were either immediately communicated to the affected parties, or communicated in the following status check, depending on the urgency. Halfway through Event B one important problem was uncovered, as a representative of one team questioned the ability of all teams to actually deliver what they had planned. This observation led to a joint decision by product management and scrum masters to extend PSI1 with one iteration.

4.6. Final plan review and planning retrospective

The final plan reviews of both events were conducted in a similar fashion as collective status check presentations. Each team had a 6 minute time box to present the plan and objectives they had for PSI1. Risks written on sticky notes during the team breakout sessions were annotated to the team and gathered on a wall. Each risk was assigned to a person or team who is responsible for mitigating it, accepted as something that might happen but requires no additional preparation, or had already been mitigated by the team who recorded it and no further action was required.

A vote of confidence in the PSI1 plan was done in two parts. 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. Both votes showed high overall confidence level.

In the Event A retrospective everyone was asked to voice their opinions on what went well and what should be improved, and the results were recorded. The results were later on recorded electronically so they could be easily accessed in further event planning. A planning retrospective was not conducted in Event B because the time allocated for it was needed to complete the planning.

4.7. Opinions toward the created plans and the planning method

According to the survey answers, the opinions on planning method and output are mostly positive. Table 1 shows results of rating the planning events from the surveys. The question was "Overall, how would you rate the whole release planning/re-planning event? (1 = Poor, 6 = Excellent)". Event A had mode rating of 5 and median of 5.0, while Event B had mode rating of 4 and median of 4.0. While Event B got one step lower rating than Event A, both sessions got positive ratings overall.

Table 2 shows the results from the questionnaire statements on opinions toward the planning method and planning. The Value-columns show the number of answers on each step of the scale, which ranged from 6 ("Strongly agree") to 1 ("Strongly disagree"). The n-column shows the number of answers for each statement. Statements one, two, and three gathered confidence ratings of the output of Event A. The mode answer is "Agree" and median value 5.0 for all three statements, which indicates good confidence toward the planning output. Statements four and five gathered attitudes toward the method itself after Event A. Both statements had mode answer of "Agree" and median value 5.0, which indicates that the method was well liked. Figure 3 shows the results from the questionnaire statements on opinions toward the guidance the developers received from the stakeholders in Event A (a) and Event B (b). The questionnaire contained one statement in the form "During the team breakout our team got all needed guidance from stakeholder group X" for each internal stakeholder group (X). The "product management and/or product owner" stakeholder group was split into two groups in the Event B questionnaire.

5. Discussion

5.1. Lessons learned

5.1.1. Features should be tentatively pre-assigned. The first breakout during Event A was disorganized and non-productive. The pre-assignments helped most of the teams to get the planning on track. Based on our observations, this seems to be a practical limit of self-organization in this type of release planning, as it is not practical to have all teams discuss the feature assignments together.

5.1.2. Features should be prioritized in advance. During Event A clear priorities for features were missing. The problem became evident especially for Team α whose feature assignments were changed three times during Event A. Features should be prioritized in advance. This was done in Event B, where materials included a rank ordered list of features to be planned, and we observed that it increased the efficiency of the planning.

5.1.3. Stakeholders should be available at the planning event and prepare materials in advance. Product management representatives and system architects were a bottleneck for planning in Event A (see Figure 3(a)). They

Table 2. Opininons on the method and output

Statement	Value						
	6	5	4	3	2	1	n
1. "I have a clear vision of what I am going to do for the next 90 days"	8	12	7	0	1	1	29
2. "My team's plan for creating the next Potentially Shippable Increment is realistic"	6	14	4	2	0	1	27
3. "I believe this project will create a successful solution"[a]	5	15	10	1	1	1	33
4. "I like this method of release planning"	11	12	3	4	0	1	31
5. "This method of release planning is effective"	8	15	4	2	1	1	31
6. "Scrum of Scrums status check was a useful practice"[b]	5	7	5	2	2	0	21

[a] Solution is a term used in the case company for commercial software products. [b] From Event B questionnaire.

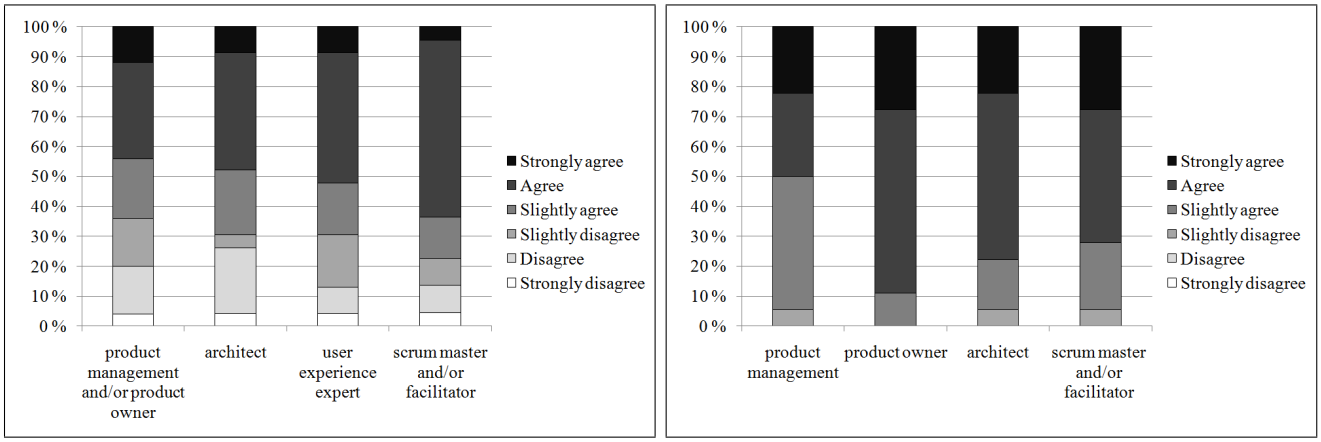
could not help all the teams at the same time with the new architecture and features. The improved preparatory work and presence in Event B helped and the teams got better support (see Figure 3(b)). However, the teams would have still preferred to have the materials in advance.

5.1.4. A combination of infrequent thorough status checks and frequent short status checks should be used. Based on our observations, status checks were too infrequent during Event A to uncover important dependencies and impediments. On the other hand, many participants thought that the status check presentations took too long and were not useful. The combination of infrequent long status checks to give everyone an overview of the plan and short frequent status checks to solve impediments and dependencies alleviates both problems. In addition, scrum-of-scrums -style status checks do not interrupt the planning of the whole team.

5.1.5. Dependencies should be resolved immediately when possible. In the beginning of Event A the teams did not immediately start discussing dependencies between teams. We observed much more interaction between the teams after the teams were instructed to solve dependencies immediately and only defer problems they could not solve to the next status check.

5.1.6. The number of included features should be limited and the included features should be traced. The case company's product management representatives had difficulties in keeping track of the planning during breakout sessions. One reason for this was that many non-important features were included in the list of features, even though they had no chance to be included in PSI1. Another reason was that the features did not have unique identifiers so that the teams could have easily associated their stories to related features. In Event B unique identifiers were given to features and references from stories to the associated features were made. According to our observations and feedback from the questionnaire, these practices helped product management to trace features' planning status.

5.1.7. The release planning method helps uncover potential problems for the whole project. In Event A the teams filled their iterations with stories based on the teams' projected velocity without accounting for the uncertainty in



(a) (b)

Figure 3. Results of the statement "During the team breakout our team got all needed guidance from stakeholder group X" after Event A (a) and Event B (b)

their effort estimations. Many details of the new architecture were unknown and the complexity of the whole system was high, which resulted in effort estimates that were too optimistic. This problem was uncovered in a scrum-of-scrums during Event B and a solution was created. This can be seen as a strength of the joint release planning method, which allowed a participant to uncover a potential problem for the whole project and made it possible to take corrective action immediately.

5.2. Limitations

Three types of triangulation were utilized to increase the construct validity of the results [18]. The observations were carried out by two researchers. This paper was also reviewed by a case company representative who participated in both release planning events (investigator triangulation). The researchers wrote separate observation notes which were compared and consolidated, two voice recorders were used to record the event, and the recordings were used to validate the observation notes during data analysis (data triangulation). In addition to passively observing the events, the observations and improvement suggestions created based on the release planning event were reviewed and discussed together with case company representatives (methodological triangulation). The fourth type of triangulation, theoretical triangulation, is of no importance to this paper, as the case is not explanatory nor builds a theory. This paper only describes a single revelatory case study. It should not be read as a generalization of the method, but as a description of an interesting phenomenon.

6. Conclusions and future work

The main contribution of this paper is a revelatory case study report of two joint release planning events in a Finnish software product company. In addition, we describe improvements made between the two events and present lessons learned. The description of the two joint release planning events can be utilized by other companies to help them set up their own joint release planning events. Companies already utilizing a similar method can get ideas on what works and what does not work and improve their release planning events.

We believe that the joint release planning method is a significant advancement in the continuing effort in the industry and the research community to find ways to scale agile software development methods to multi-team software development contexts. We continue to study the release planning events in the case company. In addition, we aspire to study additional software product companies which use similar methods for release planning to build a generic model of the joint release planning method. We also encourage other researchers to study and report any instances of similar research planning methods they encounter.

References

[1] P. Abrahamsson. New directions on agile methods: A comparative analysis. In *Proc. International Conference on Software Engineering (ICSE 2003)*, 2003.

[2] B. W. Boehm. Requirements that handle ikiwisi, cots, and rapid change. *Computer*, 33(7):99–102, 2000.

[3] B. W. Boehm and R. Turner. *Balancing agility and discipline: a guide for the perplexed*. Addison-Wesley, Boston, MA, 2003.

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

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

[6] T. Chow and D.-B. Cao. A survey study of critical success factors in agile software projects. *Journal of Systems and Software,*, 81(6):961–971, 2008.

[7] N. D. Fogelström, T. Gorschek, M. Svahnberg, and P. Olsson. The impact of agile principles on market-driven software product development. *Journal of Software Maintenance and Evolution: Research and Practice*, 22(1):53–80, 2010.

[8] J. A. Highsmith. *Agile software development ecosystems*. Addison-Wesley, Boston, MA, 2002.

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

[10] A. Ngo-The and G. Ruhe. A systematic approach for solving the wicked problem of software release planning. *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, 12(1):95–108, 2008.

[11] K. Rautiainen, C. Lassenius, and R. Sulonen. 4cc: A framework for managing software product development. *EMJ - Engineering Management Journal*, 14(2):27–32, 2002.

[12] J. Rothman. *Manage it!: your guide to modern, pragmatic project management*. The Pragmatic Bookshelf, Raleigh, NC, 2007.

[13] G. Ruhe and J. Momoh. Strategic release planning and evaluation of operational feasibility. In *Proc. 38th Annual Hawaii International Conference on System Sciences (HICSS '05)*, 2005.

[14] K. Schwaber and M. Beedle. *Agile software development with Scrum*. Prentice-Hall, Upper Saddle River, NJ, 2002.

[15] A. Shalloway, G. Beaver, and J. Trott. *Lean-agile software development: achieving enterprise agility*. Addison-Wesley, Upper Saddle River, NJ, 2009.

[16] M. Svahnberg, T. Gorschek, R. Feldt, R. Torkar, S. B. Saleem, and M. U. Shafique. A systematic review on strategic release planning models. *Information and Software Technology*, 52(3):237–248, 2010.

[17] K. Vlaanderen, S. Jansen, and S. Brinkkemper. The agile requirements refinery: applying scrum principles to software product management. In *Proc. 3rd International Workshop on Software Product Management*, 2009.

[18] R. K. Yin. *Case study research: design and methods*. SAGE Publications, Thousand Oaks, CA, 1994.