An Experience in Combining Flexibility and Control in a Small Company's Software Product Development Process

Kristian Rautiainen
Helsinki Univ. of Technology
P.O.Box 9600
FIN-02015 HUT, Finland
Kristian.Rautiainen@hut.fi

Lauri Vuornos
Smartner Information Systems
Paciuksenkatu 29
FIN-00270 Helsinki, Finland
Lauri.Vuornos@smartner.com

Casper Lassenius
Helsinki Univ. of Technology
P.O.Box 9600
FIN-02015 HUT, Finland
Casper.Lassenius@hut.fi

Abstract

This paper presents a longitudinal case study at Smartner Information Systems, a small software product company operating in a dynamic and uncertain environment. Smartner successfully combines flexibility and control in their product development process. Flexibility is gained with monthly sprints, after which new decisions about project scope can be made in planning the following sprint. Control is achieved through mapping the sprints to management decision points, where the management team makes decisions conserning the whole project portfolio. The development team and other stakeholders of the product participate in sprint planning facilitating communication of business/customer needs to development. Product roadmapping and sprint demonstrations give visibility of development plans and progress to the whole organization. Freezing the development scope for a month at a time helps in giving the development team a chance to work on their assigned tasks and creates a more relaxed atmosphere.

1. Introduction

The majority of companies developing software are small. According to the US Census Bureau's "1995 County Business Patterns", over 90 % of the software companies in the United States have fewer than 50 employees, but still literature and existing research have mostly overlooked software engineering in the small [11]. Small companies find it hard to use existing software process improvement models and standards, such as the Software Engineering Institute's Capability Maturity Model for Software [6], because of lack of resources and fear of excessive process overhead as a result [5]. Small companies need effective software engineering practices tailored to their size and type of business [11], but practices designed for large businesses do not scale down easily [15].

Another shortcoming in most existing software process improvement models from the point of view of a small software product company is that they have their background in the customer project business or software service business, which differ from the software product business in their underlying business models [17]. The business aspects may place constraints on the software process. For instance, the processes and practices needed to develop and maintain a software product for thousands of users should differ from the ones needed for developing a customer-specific solution.

In contrast to the traditional plan-driven software development processes, several agile process models have been proposed, e.g. [3,12,18,22]. These are said to be suitable for small teams in companies operating in dynamic and uncertain environments [13], but scientific empirical evidence of the effectiveness of agile models is lacking [1]. However, empirical studies have shown that many companies, both large and small, in the Internet software and PC software businesses use flexible processes [8,9,10], and such flexible processes have also been found to lead to increased customer satisfaction [16].

Agile models address what should be done at the "floor level", but their connection to a company's product planning and strategy processes is not clearly addressed. For product-oriented companies product portfolio planning and roadmapping have been found to be crucial to long-term success [8]. Thus, from the point of view of a company in the software product business it is important to understand how to link these two areas of concern.

In this paper we present a longitudinal case study of how one small Finnish software product company, Smartner Information Systems Ltd. (Smartner) improved its product development process. The company successfully combined practices from multiple agile process models and integrated long-term product and business planning into their product development process, combining flexibility and control. The purpose of this paper is to describe what Smartner did, why they did it, and how it has influenced the process and product. We

also hope that the paper can inspire other small software product companies to improve their product development processes.

The rest of the paper is structured in the following way: the next section presents the research methods used followed by a description of the background of Smartner and its product development process. Then we describe the process improvement initiative and the improved product development process followed by the findings of our study. We conclude the paper with a brief evaluation of the study and outline our future research.

2. Research methods

This paper is based on a single longitudinal industrial case study [25]. The first and third author (researchers) attended meetings concerning the process improvement initiative at Smartner between September 2000 and March 2002 in the role of participant-observers. The role of the researchers at this point was consulting. The researchers provided a high-level framework [20] they had constructed and wanted to validate and develop further for the company to use, as well as their knowledge and experience in working with similar companies. The responsibility for implementing process improvement actions was held by the company at all times.

Meetings were held more frequently in the first six months, approximately twice a month. We collected data in the form of process-related documents that the company personnel wrote and notes made at the meetings. The meetings' agenda was always similar. Company personnel presented how they had implemented or planned to implement process improvements and the researchers gave feedback. The meetings ended in discussing further improvement ideas and planning action points to be completed until the next meeting.

Between April 2002 and February 2003 Smartner independently continued improving and deploying the process led by the second author (R&D team leader, process owner and process improvement champion) and in March 2003 the researchers revisited the company to collect data on the progress and success of the process improvement initiative. At this time data was collected by reviewing relevant documents, reviewing defect tracking and version control system data, conducting taped semi-structured interviews with key stakeholders and discussing with the R&D team leader. This is described in more detail in section 5.

3. Background

Smartner Information Systems Ltd, founded in 1999, is a software product and professional services company enabling mobile business services. Smartner offers its mobile technology for operators and application service providers who need tools for building mobile services and solutions for enterprises. Customers include Radiolinja, Swisscom Mobile and Vodafone Ireland. In March 2003, Smartner employed about 30 people with expertise in mobile solutions, software development and business management. Smartner is based in Helsinki, Finland. Smartner's main product is a mobile office product, a mass customized enterprise software.

Smartner's organization is divided into five teams: Sales and Marketing (S&M, 4 people), Research and Development (R&D, 11), Product (PT, 3), Professional Services (PS, 8) and Administration (A, 5). Sales is responsible for negotiating and closing deals and managing customer relations. Marketing is responsible for public relations and communicating Smartner's message to partners and customers. Professional Services is responsible for consultation, product support, and management of delivery and integration projects to the customers. R&D is responsible for developing and maintaining the software products, and Product team is responsible for planning and envisioning what the products should be. The company's management team consists of people from the operational teams and is led by the CEO.

The first draft for a development process was written in the beginning of 2000. It resembled the traditional waterfall model [21]. The uncertain environment made planning in advance very difficult and the process was changed in June 2000 to be iterative and incremental. At the same time a product management view was adopted with productization milestones from kick-off to product launch. This resembled the Stage-Gate™ model for new product development [7] and showed product development as a part of productization and made its link to business clearer. However, the constantly changing product requirements and architecture made product development difficult to control and schedules were overrun, according to the R&D team leader.

At this time Smartner contacted the researchers and joined a research project aimed at the development of a framework for managing software product development in small software product companies by ensuring a fit between software development and management practices and the business model(s) of the company. Smartner was the first company to implement a preliminary version of the framework developed in the research project.

Smartner's first product release was made in mid-July 2000. At the same time development instructions for R&D were made explicit, covering the overall process and details about specifications (requirements, functional and technical), implementation work (unit test scripts and reports, Javadoc documentation, and code review notes), testing (different types of test cases and reports were identified) and the operational product release process (how to make product builds for releases). In an internal

R&D team feedback session the developers expressed happiness with the clear roles and responsibilities within R&D. The issues that raised most dissatisfaction were file and document management (information could not be found easily), schedule overruns, and the stakeholders' inability to freeze the specifications.

In November 2000 some problems in product development were identified by the company, namely lacking communication between R&D, Product Management (called Product team today) and Professional Services, unclear division of roles and responsibilities between these groups, unclear priorities between different products and their features, and unclear real customer/business needs for the products. These issues together with the objectives to make the product development process visible and understandable for all stakeholders, and to combine flexibility and control were the main process improvement goals Smartner presented to the researchers.

4. The process improvement initiative

The R&D team leader was the champion of the product development process improvement initiative at Smartner and also wrote his Master's thesis about it. Before we move on, we need to clarify the two main concepts used at Smartner, the product *release process* and the product *development process*. The product release process is the productization process, in which all stakeholders contribute to making a product release. The product development process is part of the product release process and comprises the software engineering efforts done by the R&D team. Other processes closely connected to these two processes were improved by other people, but a closer examination of these is out of focus of this paper. Here we will focus on the product development process and its most important connections to the product release process.

4.1. Overview

The product development process improvement initiative at Smartner is ongoing. Figure 1 shows a timeline with process improvement milestones and major product releases. The major process improvement milestones are marked M1-M4, and are referenced from this point on in the paper. The requirements for the product development process were gathered by the R&D team leader by interviewing key people and arranging a workshop to discuss the factors that define the company's operational environment and the requirements those factors pose to product development. Also, the current product development process and its practices were discussed to understand why they had been chosen, what worked and what did not. The main requirements identified for the improved process were: schedule oriented (schedules should be met, adjustments done by dropping functionality), fast reaction to change (there must be a structured way to change plans often), customer oriented (key customers participate in planning and testing the products), managed requirements (long-term planning as well as short-term specifications), and extensive testing. For a more detailed discussion, see [23].

Due to the nature of the requirements for the process, four software process models were chosen for evaluation: Rational Unified Process (RUP) [14], eXtreme Programming (XP) [3], Scrum [22], and Microsoft's "synchronize-and-stabilize" [8]. RUP was chosen as a more traditional iterative/incremental process model. XP was chosen because of its emphasis on the development work. Scrum was chosen as an agile process model with a project management perspective. Microsoft's process was chosen because it seems to have worked well in an uncertain and dynamic environment.

None of the process models could satisfy all the requirements, but put together they succeeded in covering the requirements. This led to combining parts of all the process models in Smartner's improved product development process, which is described in the next section.

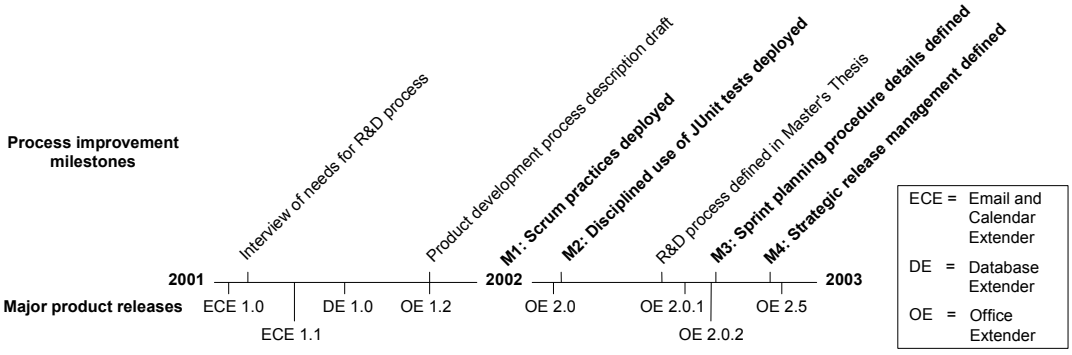

Figure 1. Timeline of process improvement milestones and major product releases

4.2. The improved product development process

Figure 2 shows an overview of the improved product development process. It is depicted based on a high-level framework (Four Cycles of Control, 4CC [19]) for managing software product development in small companies.

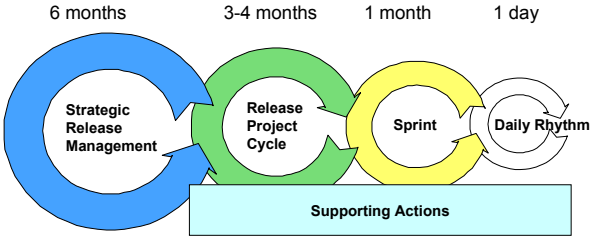

Figure 2. Overview of the improved product development process

Table 1 shows the details of the process, which are explained in the following sections. The findings from deploying the process are described in section 5.

4.2.1. Strategic release management. Strategic release management aims for setting the direction for product development by aligning the product development efforts with the business and technology strategy of the company. This is accomplished by setting a vision for the products (one-year planning horizon), scheduling upcoming product releases into a product roadmap and prioritizing features to be implemented in those releases. Strategy updates are done biannually and the product roadmap is updated after each product release.

The concept of product vision is applied from synchronize-and-stabilize and RUP combined with Scrum's concept of backlogs. All ideas for features in a product are collected in a product backlog. The product vision crystallizes the main scope of the upcoming releases and explicates the main product features from the product backlog to be developed in each product release. A rough effort estimation is done to these features, which are then grouped into preliminary release backlogs. The releases are scheduled into the product roadmap. Based on the roadmap, high-level resource allocation can be planned. Even if resource allocation in a small company is sometimes done on a daily basis, making high-level resource allocation between different projects such as new product development, product delivery, product customization and support is necessary to avoid or at least spot overcommitment of people.

Strategic release management is done by the management team of the company and the sprint board. The sprint board consists of the most important stakeholders of the product: the head of the product team, the product manager, the R&D team leader and the head of professional services. The sprint board prepares the issues for the management team, which makes the

Table 1. Details of the improved development process

	Strategic Release Management	Release Project Cycle	Sprint	Daily Rhythm	Supporting Actions
Goals	Set product vision Define the main content for product releases Schedule product releases Resource R&D projects	Update product vision and refine release backlog Schedule sprints Produce a tested product	Define the sprint backlog and goals Produce a stable product increment Show sprint results to all stakeholders Get feedback for further development of the product	Steer the development towards the sprint goal Produce high-quality code Synchronize and communicate work of development team members	Help development team to concentrate on achieving the sprint goal Form a common basis for working with tools and other infrastructure
Activities	Collect feature ideas and feedback on products Update product roadmap Decide on resource allocation for the release projects in the roadmap	Set dates for sprint reviews Select and prioritize features for the release Define testing strategy for the release	Sprint planning Sprint management Sprint review	Design Code Test Demonstrate working software Scrum meeting	Configuration mgmt Requirements mgmt Defect tracking Daily builds Automated testing Maintaining test and development environments
Participants	Management team Sprint board Customers and partners	Sprint board Customers and partners Development team	Sprint board Scrum master Development team Any stakeholders in reviews	Scrum master Development team	System administrator Product manager Scrum master Development team

final decisions. Customers and partners participate by providing and evaluating ideas for future product releases.

Smartner used simple roadmapping on a six-month planning horizon before the process improvement initiative, but as mentioned in the end of section 3, there were problems in making the plans visible and understandable to all stakeholders. Explicating a product vision and defining preliminary release backlogs one year ahead were thought to remedy at least some of these problems. The product backlog provides a systematic way to collect feature suggestions continuously from all stakeholders, compared to doing that only before each release project earlier. This is thought to help in considering user feedback/needs in long-term product planning.

4.2.2. Release project cycle. The main purpose of the release project cycle is to produce a product release. The length of the release project cycle is 3 months, after which a beta release of a product is ready. To complete all the work needed for a market launch, an additional market release sprint is needed, in which the product is thoroughly tested and the accompanying product documentation is finalized. The sprint board is responsible for managing the release project cycle.

The activities in the release project cycle can be divided into three parts: project planning (scheduling the sprints and refining the projects' scope), decision-making at release process gates, and product release.

The most important sprint scheduling issue is to plan the dates and times for the sprint demonstrations. These dates are published to the whole company so that people can include them into their personal schedules. Another important use of sprint scheduling that can also influence the content of the product release is giving selected customers a possibility for early access to the new product version, e.g., for trialing the product or joint testing. As described in synchronize-and-stabilize [8], this early external testing is an important source of feedback about the product and can be used in refining the scope of later sprints. Improvement suggestions are included in the product backlog and planned into future product releases.

The main task in refining the projects' scope is reprioritizing the features in the preliminary release backlog defined in the product roadmap. New features can be added to the release backlog, meaning that other features must be excluded. The reprioritization of the features is the responsibility of the product team. It is done in different sessions with internal experts and selected customers and partners. The internal sessions are 1) a technical planning session where architectural issues are discussed and considered in the feature prioritizations, and 2) a session for defining the testing strategy for the release, which covers testing whose effort needs to be included in the release backlog, such as performance testing, usability testing and testing on different platforms (e.g. operating systems).

In October 2002 (M4) a release process with gates in which the management team makes decisions about the release projects were defined in detail by the head of product team. The definition included a description of all different stakeholders of the product and their responsibilities and deliverables in making a product release. As mentioned earlier, this resembles the Stage-Gate™ model for new product development [7]. After each sprint, the management team gets a status report (including an update of effort estimates) from all the projects, and go/kill decisions can be made considering the whole project portfolio.

The biggest differences to the earlier release project practices is the introduction of Scrum terminology and practices in project planning and including the responsibilities and deliverables of all product stakeholders to the release process description. At the same time the release process control points (gates) have been mapped to the development process control points (sprint reviews). These are thought to help in increasing the visibility of the process and avoiding lacking communication between stakeholders. Also, adding a stabilizing market release sprint after the beta release of the product is thought to help in achieving release-quality software and preventing schedule overruns.

4.2.3. Sprint. The purpose of sprints is to develop stable increments to the product. The concept of sprint is adopted from Scrum. The length of a sprint is approximately one month. A scrum master—also a concept adopted from Scrum—is responsible for the sprint.

The sprint starts with sprint planning and ends with a sprint review and a demonstration of what has been achieved in the sprint. Sprint planning has two goals: defining the sprint goals and creating a sprint backlog. This is done in 5 steps taking up to 1½ days to go through (2-5 hours for the development team and 4-8 hours for the sprint board, according to the R&D team leader):

1. *Defining the issue backlog* is done in a sprint board meeting. Inputs are customer commitments, the product roadmap and unfinished tasks from previous sprints. The output is a list of issues to be tackled in the following sprint.
2. *Designing tasks in the issue backlog* is done as groupwork by the development team, where the developers define the tasks that need to be done to complete the issues in the issue backlog. If any issue is unclear, it is immediately discussed with the sprint board for clarification.
3. *Estimating efforts of the tasks and checking availability of resources* is done as groupwork by the development team.

4. *Selecting issues to complete and postpone* is done by the sprint board. The list of issues is typically longer than can be accomplished in a sprint, so choices must be made, what to include in the sprint backlog and sprint goals, and what to leave to future sprints. This was made explicit by defining the sprint planning procedure in August 2002 (M3).
5. *Committing to the sprint goals* is done by the development team. It reviews the choices made by the sprint board and accepts the tasks.

A sprint is managed using a project burndown graph and daily scrums, as suggested in Scrum. Work estimates for tasks are updated and new tasks are added by the developers during the sprint, if necessary, and the estimate of remaining work is shown in the burndown graph. An indicator of problems in development is that the estimate of remaining work goes up instead of down, which often happens when the work is better understood or something unexpected occurs. At that point the scrum master can decide to contact the sprint board to agree on adjustments to the sprint tasks and goals.

The sprint ends with a sprint review session, where anyone from the company can participate. The scrum master is responsible for coordinating the session. It is an informal meeting including a sprint overview, where the highlights of the sprint are presented. The sprint goal and sprint backlog are compared with the actual results in the sprint and reasons for deviations are discussed. An architectural overview is given to describe the main technical points and their relation to product functionality. The product increment is demonstrated feature by feature in a simulated production environment to concretely show what has been accomplished. Questions, suggestions and discussion are encouraged throughout the sprint review.

All these practices are new. Previously iterations were planned in an almost ad-hoc manner, contributing to too much flexibility and lack of control. The whole development team was not directly involved in the planning, so the business goals and requirements were not communicated to them. Task effort estimations were only done sporadically, if at all, which contributed to unrealistic goals. No sprint review sessions were held, so the visibility of project progress was very limited.

4.2.4. Daily rhythm. The purpose of daily rhythm is to coordinate day-to-day activities in product development towards the sprint goal. The main activities are design, code, test, synchronize, demonstrate and adjust.

Design has two main principles: keep it simple and keep the end-user in mind. Refactoring is an essential element of coding, as well as collective code ownership and a common coding standard. Pair programming is not used extensively, only when developers deem it useful for solving a problem or sharing information. In daily rhythm the focus in testing is on unit testing and doing it in parallel with coding using JUnit as a tool to help automize the tests. Having unit tests in place for the components encourages developers to refactor the code, because errors introduced should be picked up by the tests. These ideas and practices are adopted from XP.

Synchronizing the effort of development team members is done in two ways: a daily scrum (from Scrum) and a daily build-test cycle (discussed in both XP and synchronize-and-stabilize). The daily scrum works as a status meeting, lasting a maximum of 10 minutes. A build is done automatically every night, taking the latest code versions from the version control system. The unit tests are run automatically against the new build and a report is generated and available the next morning.

Short weekly demonstrations are made by the developers to show what they have accomplished during the week in terms of running software. Adjusting the project is done frequently in minor ways, such as moving tasks between developers, changing the implementation order of tasks or refining tasks when unforeseen dependencies emerge, and updating the effort estimations.

Although some of the practices above were used earlier, the definition of the daily rhythm practices is more specific and considered to improve teamwork capabilities.

4.2.5. Supporting actions. Supporting actions include tools and infrastructure for helping product development achieve its goals and form a common basis for working. These include tools for configuration management, requirements management, defect tracking, daily builds and automated testing. Maintaining test and development environments is also part of supporting actions.

5. Results

In March 2003 the researchers revisited Smartner after a period of 11 months, during which the company had independently continued to deploy and improve their processes. The researchers reviewed relevant documents concerning the process improvement, such as process descriptions and minutes from feedback sessions, dating back to the beginning of 2000 in order to get an overall view of what had happened. Defect tracking and version control system data was gathered and reviewed in order to see how the product had evolved. Hour reporting system data was gathered in order to see how much time had been used to process improvement. Backlogs were reviewed and compared to realization data from the hour reporting system in order to have an objective measure of how the process has worked.

Semi-structured interviews with key people were performed about their experiences with the new process and the ongoing process improvement efforts in order to get a subjective opinion on how the process has worked.

The people interviewed were the head of product team, the head of professional services, the chief architect and a developer. They were all asked to tell about their roles in the process and the process improvement, to evaluate from their perspective how the process works now compared to before, how product quality has been influenced if at all, how the communication works between the different stakeholders, how the process improvement has succeeded from their perspective, what still needs to be improved and the reasons behind all these. The interviews were conducted by one of the researchers. They were taped and transcribed in mindmaps.

All the data was analyzed and discussed together with the R&D team leader. Although this adds risk of bias to the results, it also gives a richer interpretation of the data. This is a trade-off we were willing to make. The following sections present the main findings.

5.1. Process

All interviewees reported that the main purposes of sprints, i.e. freezing the requirements the development team works on as well as giving the development team a month without external interruptions, were not fulfilled when sprints were first deployed bottom-up by the R&D team. Customer projects took precedence over product development, causing sprint goals to be missed, because only part of the time allocated to development could be used. This happened partly because the new process was not yet understood by all, partly because management was not fully committed to the new process, and partly because the customers were prioritized ahead of product development at that stage of Smartner's life cycle.

All interviewees reported independently of each other that the major breakthrough in process improvement happened when a top-down approach was taken and the management processes for roadmapping, product releases and sprint planning were detailed and deployed in M3 and M4. When asked about the increased formalism and bureaucracy this improvement entailed, the chief architect and developer told that they were very happy with the changes and that the changes made the process "more sturdy", but at the same time "more relaxed". Now they have the opportunity to work with their assigned tasks for the duration of sprints, which they claim has helped them get the work done in sprints from M4 and on. They also claim that they have learned to estimate task efforts better, which has helped in setting more realistic goals for the sprints.

When comparing the hours of work done in sprints to the estimate of effort left undone at the end of sprints (Figure 3), we can see that the numbers to some degree corroborate the above claims. However, the numbers are neither representative nor conclusive, since we could not compare realized hours with the original planned hours, and how many tasks had been dropped during the sprints as corrective action. Also, the estimated work left undone cannot be trusted, since estimation was made less systematically before M3 and has improved since. The missing data in Figure 3 relates to the time when two major customer projects were done. The R&D team leader estimates that during this time the percentage was much higher.

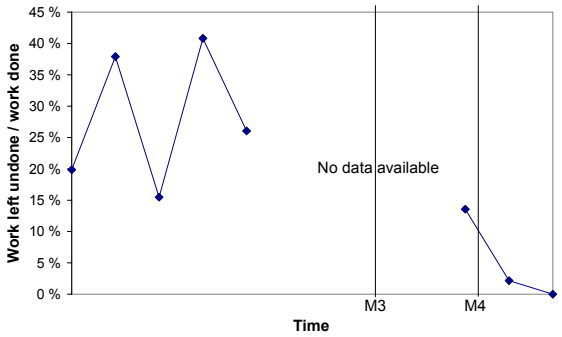

Figure 3. The percentage of work estimated left undone at the end of sprints in proportion to work done in sprints

According to the chief architect and developer, another benefit of the sprint planning session is that the dialog between the development team and the sprint board has clarified the business objectives of the product to the development team. Also, since customer inputs and commitments are now considered during sprint planning, there have been fewer surprises that take away product development resources during the sprints, which has contributed to better success in reaching the sprint goals.

The release process gates and the estimation and specification corresponding to the gates together with monthly sprint demonstrations has helped the management team follow the progress of development and make more informed decisions, according to the head of product team. He told that before the gates were taken into use, the management team could not understand all the implications of their decisions and had to rely more on the expertise of the sprint board. Also, now the plans are more detailed in the product roadmap and development more predictable, so when customers ask for something, answers about schedules of future releases and their content can be given with high confidence, which has satisfied the customers. The head of product team and the head of professional services both reported that everybody now realizes that there is no need to make adjustments to the product within less time than a sprint and that the possibility to make changes to the release content each month in sprint planning is flexible enough.

5.2. Product

The product has evolved and improved since January 2002. Looking at incident reports from customer delivery projects, the head of professional services says that problems attributed to the product have decreased significantly. This can be partly explained with a maturing product, but both the head of professional services and the chief architect claim that the improved testing practices with, e.g., automated unit testing have also made the difference. Other data does not fully support this. Although the amount of automated unit test code has grown, the amount of source code has grown even more as the ratio shows in Figure 4. This can partly be explained by the fact that user interface components are not tested with automated unit tests and their share of the code has grown. The big jump in amount of code in Figure 4 is explained by moving a large code base from one tree to the main tree in CVS.

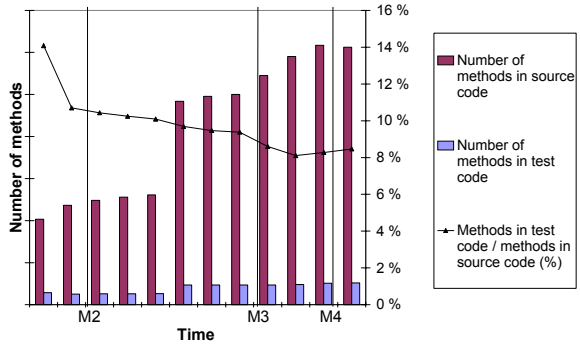

Figure 4. The evolution of the main code base and unit test code base

When comparing product components that have been developed after M2 with components developed earlier, only a fraction of defect reports from customers have been attributed to the newer components as can be seen in Figure 5. In this period 4 major customer deliveries have been done.

Looking at the amount of serious and critical defects reported by customers (Figure 6), the trend has been steadily going down during 2002. The data does not show what the exact reason for this is. The Chief Architect estimates that the new product that is ready in April 2003 will have even less defects. He speculates that this is because there has been more time for testing after M4, and that better testing tools are available. Another reason he gives is that the product architecture has improved to entail more and better defined components, which can easily be tested separately.

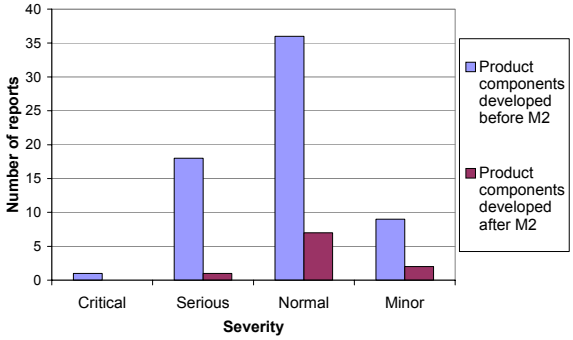

Figure 5. Defects reported by customers

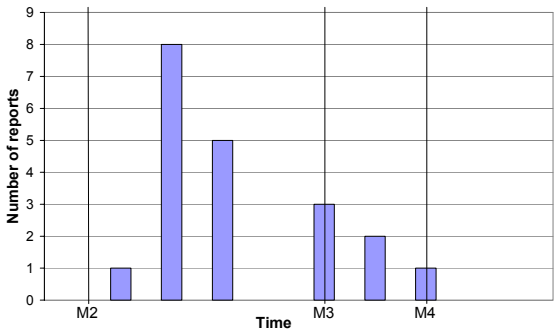

Figure 6. Serious and critical defects reported by customers per month

5.3. Success factors

The R&D team leader's opinion was that the high-level framework (4CC) had helped him envision how product development could be organized in the company. It gave him the idea how to combine flexibility and control and made choosing practices from the agile process models easier.

The developers have been in a key role in the process improvement initiative. It started bottom-up with picking ways of working that felt natural to the developers, which have then further evolved. Many of these were inspired by the practices of XP and adapted to suit Smartner's developers. But the process did not work as expected, until management committed to the process. This happened when the management processes for roadmapping, product releases and sprint planning were defined and deployed in M3 and M4, integrating long-term product planning into product development, as agreed by all interviewees.

All interviewees thought the incremental approach to process improvement was a success factor. As the chief architect put it: "We learned something new in every sprint and made small adjustments to the process in feedback sessions. I don't think we could have done it any

better or faster." Of course, the process improvement initiative needed a driving force or champion, who was the R&D team leader. He reported that without extra "pushing", the process improvement initiative might have stalled completely at several points in time.

To give an indication of the workload needed for process improvement—notwithstanding possible ineffectiveness due to a learning curve—the R&D team leader recorded 120 hours to process improvement during 2002, and the rest of the R&D team recorded all in all 49 hours. Data on the hours recorded by management was not available. The effort needed might vary significantly from company to company depending on the organizational culture and attitudes towards process improvement and how familiar the personnel already is with different practices and process models. At Smartner process improvement has been considered important since the company was established, which was mentioned as a success factor by the R&D team leader.

5.4. Further improvement needs

One concern identified by the head of product team is being able to manage simultaneously customer projects and product development according to the roadmap with limited resources. Outsourcing has been tried in product development earlier, with some bad experiences, mainly concerning quality of work and knowledge transfer between Smartner and the independent subcontractor. Recently, outsourcing has been tried in customer delivery projects and that has worked better. One reason mentioned is that it was easier to control, because it was done as bodyshopping. This should, however, be considered in the process.

The developer reported that dependencies between the tasks in the sprint backlog and the order in which to perform the tasks were not written down anywhere, but just discussed and agreed upon within the development team. In the last sprint there had been some problems with this and considering that some of the people work part time, this could be an issue for further improvement.

Effort estimation ability is always an issue in software engineering. The chief architect wanted this to be a personal mission for each and everyone to improve, although he also said that the accuracy had become much better lately. The head of product team expressed it from a different viewpoint: "We still get very busy at the end of projects, but now we can see it coming two months in advance."

6. Discussion and future research

In this paper we reported how Smartner—a small software product company in a very dynamic and uncertain market—succeeded in combining flexibility and control in their improved product development process. Flexibility is gained with the monthly sprints, after which new decisions about project scope can be made in planning the following sprint. Control is achieved through mapping the sprints to release process gates, in which the management team makes informed go/kill decisions based on the whole project portfolio. This is a long-known best practice reported in new product development literature [7]. The high-level framework, 4CC, helped in envisioning how this could be done. Another approach to integrating business and software development models has been reported in [24].

Freezing the development scope for a month at a time helps in giving the development team a chance to work on their assigned tasks and creates a more relaxed atmosphere. This, combined with development practices that give a daily rhythm are the key to more predictable development during sprints.

Smartner also successfully communicates the business/customer needs for the product to development through sprint planning, where the whole development team and other stakeholders of the product participate. This, combined with high-level product roadmapping and sprint demonstrations has given visibility of development plans and progress to the whole organization and facilitated communication between the different organizational units.

The case study shows that practices from different agile process models can be combined to make a coherent whole. When Smartner made a list of requirements for its development process and evaluated existing process models against these requirements, none of the evaluated models fulfilled all the requirements. The chosen combination of processes and practices, however, got the job done and suited the organization's culture. This goes to show that one size really does not fit all and is a lesson to all practitioners creating or improving their development process. They should take time to consider the requirements for the process, both from a business and a technical viewpoint, as well as cultural issues, in order to be able to adopt and combine processes and practices for their own needs. A risk-based approach to balancing agile and plan-driven methods when tailoring a development strategy has been presented in [4]. Combining agile methods is also not a new idea (see e.g. www.controlchaos.com/xpScrum.htm on combining XP and Scrum), but our findings contribute to the empirical body of knowledge on the subject.

The identified success factors of the process improvement initiative, namely using small improvement steps, getting management commitment, having a champion as the driving force, and involving all the stakeholders are also not a novel finding (see e.g. [2]). However, the importance of these factors should not be forgotten.

We believe the findings of our study contribute to encouraging practitioners in small software product companies to start improving their software development process by combining and adopting different processes and practices to suit their own needs. This study can be used as an inspiration of how one small company in a very dynamic and uncertain environment so far has succeeded in their process improvement. The process description could be used by other companies in a similar environment as a good starting point for tailoring their own process.

There is risk of author bias in the results, but we believe that it is decreased with triangulation of data from multiple sources. Researcher bias is decreased by the fact that most of the work was conducted by the company on its own without researcher involvement. Given that there is only evidence from one case, the results may not be widely generalizable, but we believe the results give an indication of what can be accomplished in a small software product company in a dynamic and uncertain market.

In the future the researchers continue working with other case companies, improving the framework for managing software product development based on the gained experience. We plan to do a multiple case study in the future with all the companies participating in the research project to try to identify possible patterns of what works and what does not in different types of companies.

References

[1] Abrahamsson, P., O. Salo, J. Ronkainen, and J. Warsta, *Agile Software Development Methods*, VTT Publications 478, 2002.

[2] Baddoo, N. and T. Hall, "Motivators of Software Process Improvement: An Analysis of Practitioner's views", *Journal of Systems and Software*, vol. 62, no. 2, 2002, pp. 85-96.

[3] Beck, K., *eXtreme Programming eXplained*, Addison-Wesley, 2000.

[4] Boehm, B. and R. Turner, "Using Risk to Balance Agile and Plan-Driven Methods", *IEEE Computer*, vol. 36, no. 6, 2003, pp. 57-66.

[5] Brodman, J. G. and D. L. Johnson, "What Small Businesses and Small Organizations Say About the CMM", *Proceedings of ICSE-16,* (May 1994), pp. 331-340.

[6] Carnegie Mellon University, S. E. I., *The Capability Maturity Model: Guidelines for Improving the Software Process*, Addison Wesley Longman, Inc., 1994.

[7] Cooper, R. G., "Perspective Third-Generation New Product Processes", *Journal of Product Innovation Management*, vol. 11, no. 1, 1994, pp. 3-14.

[8] Cusumano, M. A. and R. W. Selby, *Microsoft Secrets*, The Free Press, 1995.

[9] Cusumano, M. A. and D. B. Yoffie, *Competing on Internet Time*, The Free Press, 1998.

[10] Cusumano, M. A. and D. B. Yoffie, "Software development on Internet time", *IEEE Computer*, vol. 32, no. 10, Oct.1999, pp. 60-69.

[11] Fayad, M. E., M. Laitinen, and R. P. Ward, "Software Engineering in the Small", *Communications of the ACM*, vol. 43, no. 3, 2000, pp. 115-118.

[12] Highsmith, I. J. A., *Adaptive Software Development: A Collaborative Approach to Managing Complex Systems*, Dorset House Publishing, 2000.

[13] Highsmith, J., *Agile Software Development Ecosystems*, Addison-Wesley, 2002.

[14] Kruchten, P., *The Rational Unified Process, An Introduction*, Second ed., Addison-Wesley, 2000.

[15] Laitinen, M., M. E. Fayad, and R. P. Ward, "The Problem with Scalability", *Communications of the ACM*, vol. 43, no. 9, 2000, pp. 115-118.

[16] MacCormack, A., R. Verganti, and M. Iansiti, "Developing products on "Internet time": The anatomy of a flexible development process", *Engineering Management Review*, vol. 29, no. 2, Jan.2001, pp. 90-104.

[17] Nambisan, S., "Why Service Businesses are not Product Businesses", *MIT Sloan Management Review*, 2001, pp. 72-80.

[18] Palmer, S. and J. Felsing, *A Practical Guide to Feature-Driven Development*, Prentice Hall PTR, 2002.

[19] Rautiainen, K., C. Lassenius, and R. Sulonen, "4CC: A Framework for Managing Software Product Development", *Engineering Management Journal*, vol. 14, no. 2, 2002, pp. 27-32.

[20] Rautiainen, K., C. Lassenius, J. Vähäniitty, J. Vanhanen, and M. Pyhäjärvi, "A Tentative Framework for Managing Software Product Development in Small Companies", *Proceedings of HICSS-35,* (January 2002).

[21] Royce, W. W., "Managing the Development of Large Software Systems", *Proceedings of Wescon,* (August 1970), pp. 1-9.

[22] Schwaber, K. and M. Beedle, *Agile Software Development with Scrum*, Prentice Hall, 2002.

[23] Vuornos, L., *Software Process Improvement in a Small High-Tech Company: Case Smartner Information Systems Ltd.*, Master's Thesis, Helsinki University of Technology, 2002. PDF file available at http://www.soberit.hut.fi/publications/lvuornos_thesis.pdf

[24] Wallin, C., F. Ekdahl, and S. Larsson, "Integrating Business and Software Development Models", *IEEE Software*, vol. 19, no. 6, 2003, pp. 28-33.

[25] Yin, R. K., *Case Study Research - Design and Methods*, Second ed., Thousand Oaks: Sage, 1994.