Agile Customer Engagement: a Longitudinal Qualitative Case Study

Geir Kjetil Hanssen
SINTEF ICT
S.P. Andersens vei 15
NO-7465 Trondheim, Norway
+47 73597081
ghanssen@sintef.no

Tor Erlend Fægri
SINTEF ICT
S.P. Andersens vei 15
NO-7465 Trondheim, Norway
+47 73593007
tor.e.fegri@sintef.no

ABSTRACT
In this longitudinal case study we have followed a small software product company that has turned from a waterfall-like process to evolutionary project management (Evo). The most prominent feature of the new process is the close engagement of customers. We have interviewed both internals and customers to investigate the practicalities, costs, gains and prerequisites of such a transition. We have gathered data from a period of two years covering four consecutive release projects using the new process and analyzed the material in detail. Our findings implicate that close customer engagement does give certain benefits but that it comes with a cost and needs careful attention to management.

Categories and Subject Descriptors
D.2.9 **[Software Engineering]:** Management – *productivity, programming teams, software process models, software quality assurance.*

General Terms
Design, Human Factors, Management, Measurement, Performance.

Keywords
Customer Engagement, Agile Process, Stakeholder Management, Process Transition.

1. INTRODUCTION
Software engineering is a highly versatile practice and can take many forms from small tasks to huge projects, lasting for years. Technologies, complexity, criticality and business domains may vary greatly – however, one piece is always a part of the game - the customers. It may be a single person, an organization or a larger diverse group. As there are many theories, strategies and methodologies for software engineering there are many ways of engaging the customers. Nevertheless, doing it right is one of the main success factors for software engineering [1]. The fundamental challenge is to understand and capture expressed and/or un-expressed requirements and needs from the customers and to answer to them as completely and correctly as possible. Traditional plan-based approaches tend to favor a large up-front investment in requirements analysis and design having low flexibility in customer dialogue in later phases [2]. As a reaction to this, and other characteristics of traditional approaches, agile methodologies suggest a different approach, emphasizing close and continuous cooperation with the customers throughout a larger part of the development cycle. There is only scant evidence holding sufficient scientific quality on agile practices [3-5]. Furthermore, we were particularly interested to determine the effects of agile development in a product organization, i.e. a supplier of packaged software. Most published work on agile processes focus on single-customer development, mainly in IS focused forums. We believe this is due to the stronger tradition of empirical research within the IS domain. To investigate practical agile customer engagement we have followed a small sized Norwegian software product company during two years and four product releases in their transition from a plan-based development process to an evolutionary software development process. The agile method being used in this case is Evo [6]. The basic feature of Evo is that it is evolutionary, however it can also be classified as an agile process as it shares many of the concepts defined in the Agile Manifesto (www.agilemanifesto.com).

The aim of this paper is *to present a real case of agile customer engagement showing prerequisites, benefits, costs and risks in a software product setting.* We focus here on the effects on the people participating in the software process; we do not discuss potential effects on the resulting product as such or how the process affects other stakeholders than the customers, developers and product management. We find the focus on customer engagement to be particular interesting within the context of agile processes as this is one of the four foundational values of the agile manifesto.

Our findings are based upon a qualitative analysis of data collected using interviews with customers, product management and developers. Our findings indicate that implementing some of the basic principles of agile methodologies do work, however they come with a cost and an absolute premise of disciplined management. The continuation of the paper first gives a brief overview of the state of the practice with respect to customer

ISESE'06, September 21–22, 2006, Rio de Janeiro, Brazil.

engagement to identify gaps in current knowledge (section 2). Then the study context is explained (section 3) followed by a description of the study method (section 4). Further on the results and an analysis is presented (section 5). Finally we give some conclusions (section 6).

2. STATE OF THE PRACTICE

Professional software product development deals with the effective and efficient construction of software artifacts that bring value to the customer organization through benefits to the users, efficiency gains in work processes and implementation of miscellaneous company goals. Most people will agree that this is a non-trivial endeavor. Two parallel knowledge transfer processes constitute the platform of software engineering: Firstly, the knowledge of the particulars about the problem domain, the business functions, and the inherent qualities of these functions must be transferred from the customer organization to the supplier organization. Secondly, pragmatic knowledge related to the realistic fulfillment of these criteria must be transferred from the supplier back to the customer (and other stakeholders). This is basis for setting sensible expectations to the collaboration. In practice, the schedules of these processes are typically firmly embedded within the software development process. Although there might be a vast number of stakeholders in both organizations that have significant interests in these processes (politically or otherwise) and the resulting product we focus here on the issues related to the practical development and use of the product. Thus, developing a software system that will ultimately satisfy its users is critically dependant upon effective knowledge transfer between developers and users.

As a profession, software engineering is in a maturing state. This is evident from the high failure rate for IT projects. As referred to in [7], average cost overruns lies close to 34% reminding us about the need for further improvement. It is claimed that problems related to the customer relationship is the most prominent cause [8]. Researchers and practitioners seem to agree that increased user engagement is the answer to the problem [9, 10], exemplified by agile development that embraces user engagement. Value number one of the agile manifesto advocates "individuals and interactions over processes and tools". The positive effects that are credited to user engagement include: improved product quality through better understanding of the users needs, improved knowledge of customers' organization, reduced risk of producing unnecessary or unacceptable functionality, improved ability to negotiate expectations among users, improved ability to resolve conflicts regarding the design of the system, increased feeling of ownership among users, reduction in the natural resistance towards change in work practices, remedies lack of decision capability in management, improved project performance and an increased willingness to experiment and improvise in search for solutions [11-14]. Furthermore, and of strategically high importance, user engagement might help the supplier organization to embrace innovation by improving the ability to capture users' ideas and needs [9, 15].

There is only a small amount of evidential knowledge that provides support for these claims. Most studies report both varied [16, 13, 17, 18] and somewhat contradictory results [19]. We believe that this is due to a lack of knowledge of the inherent complexities that must be addressed by the software organization embracing user engagement [9]. This helps to explain why user engagement is still not widely used in practice [11]. So, what does the evidence tell us?

Arguably, personal, team-oriented skills are more important for developers working within an agile software process compared to plan-based processes [4]. Some factors that are consistently associated with benefits are of the psychological kind, relating to the personal qualities of the people and the environment in which the communication occurs. Examples of factors include the ability to accept a variety of responsibilities, discipline in communication, appropriate office designs in addition to incentives for both personal and group related achievements [11, 16, 20, 12]. Trust, of key importance to good communication, must be built across all tiers of the cooperation between the supplier and the customer organization [14, 1]. This includes the constituting, contractual and development level. Difficulties in the two former tiers are likely to affect the cooperative nature of the relationship between the developer and the end users.

Although communication is a vital process it will only benefit the knowledge transfer between supplier and customer if it is employed mindfully. Too much, and it will most likely degrade the performance of the developers [19]. Naturally, if there is too little, developers face the risk of developing the wrong software. The risk associated with the development effort should be used to tailor the collaboration between developers and end users [21, 22]. This is particularly important during development of software with high complexity [18]. Sound tradeoffs must be made here – it is important that developers are given time to focus sufficiently on the engineering problems. At the same time, an important aspect is to maintain the trust and confidence among the developers that the relevant problems are solved, and solved in the right way. This requires attention to the continuity of the collaboration. Tasks related to user engagement consume a significant effort from the developers [23], so maintaining this continuity should not be left to the developers, as it will only disturb their work routines further. As we will discuss later in this paper, discontinuity may have devastating effects on developers' confidence and performance.

Engaging users takes on many forms. Users might be *involved*, meaning that they are mentally part of the development process but might not have any specific tasks or responsibilities. Examples of user engagement are customer surveys or user forums. On the other extreme, users might be *participating* in the process – implying that they are expected to play certain roles and perform certain tasks. Examples include participatory design [24], Soft Systems [25] and ETHICS [26]. Users are occasionally promoted to decision makers – essentially directing the course of development [27]. Although this might be good for projects lacking sufficient knowledge for good decision making within project management [11] this would pose problems for a product company that depends on the adherence to product development plans, technology strategies etc., established within the supplier organization. The underlying reason is the fundamental problem of management without the appropriate mandate. However, the majority of research on user engagement has focused on custom development projects, very few have investigated user engagement in packaged product development [28, 9]. We regard the lack of empirical evidence in this field is a sound motivation for this work.

3. STUDY CONTEXT

3.1 Case company profile

The case company, CompNN, is a medium-sized Norwegian software company that provides a packaged software product for marketing and customer surveys called ProdNN. (All names are made anonymous). The product comes in two variants; an ASP-solution and a stand-alone server installation. They have a wide customer base including some of the world's largest market research agencies. The company was established by three friends in 1996 and has since grown steadily to the present situation with about 80 employees with offices in Norway, UK, Vietnam, and USA. There has been a gradual shift from building custom-made applications to a packaged product. Today, ProdNN licenses constitute most of the revenue.

3.2 The situation prior to Evo

The development process had from the start matured from ad-hoc to a well defined non-iterative waterfall-like process. This was documented in a web-based process guide and included the ordinary phases from requirements engineering through design, implementation and testing. Initially, it seemed to address their current problems with lack of control and quality management [29]. But it also introduced some new problems. CompNN struggled with changing requirements that incurred high costs due to little flexibility in the process. Much emphasis was given to extensive and thorough requirements engineering upfront but with little effect.

Product planning was at this stage a collaborative effort between R&D and management. A CompNN representative explained: "Development was mainly driven by casual discussions between R&D and management. Gradually, the need to structure the work better and allocate responsibilities for the research ahead of product planning materialized." Customers also raised the question of closer collaboration with CompNN. One customer said: "Input to the development plans was very informal. We only saw the products released by CompNN." The somewhat ad-hoc product planning resulted in a failure to address many existing business requirements from end-users. A CompNN representative said: "We were really only focusing on features. Non-functional qualities were added last, and only in very vague terms." Another representative commented "By focusing primarily on features, we somehow forgot to think about how these features should be used, and the user-perceived quality of these features."

The first point of customer feedback came late – at the phase of acceptance testing – leaving little time to improve the software before delivery. Also, effort estimates had been too optimistic, forcing developers to cope with intense amounts of overtime near the release date. As a result the product was delivered with many errors.

Coincidentally, the Quality Assurance (QA) manager and the Chief Technical Officer (CTO) of CompNN met Tom Gilb at a software engineering research forum in November 2003 where he gave a presentation of Evo. CompNN's representative found the ideas appealing and within a month CompNN decided to apply some key Evo practices to the next product version, already in the initial phase and due for release May 2004.

3.3 Evo described

Evo [6] – short for 'Evolutionary project management' – is an agile process in which the product evolves iteratively as new requirements are discovered, constructed into the code, and subsequently tested. Stakeholders are expected to give feedback on each of the iterations of the product in order to guide the development of the next iteration. Finally, suites of integration tests are performed before the final system test. Evo is a rigorous process that shares some aspects of the risk-driven Spiral model [30] but is ultimately driven by the highest possible stakeholder value in each iteration together with risk validation and reduction.

The core process of Evo is the requirements management process that serves two purposes per iteration. Firstly, selected stakeholders evaluate the achievements in the previous iteration in terms of a hands-on test. This helps to assure that progress is in line with stakeholders' expectations. Secondly, based on stakeholders' feedback the next iteration is planned – what requirements needs further improvement and what new requirements should be targeted? Requirements are not specified as functions but as product *goals* – addressing real customer business concerns. Great emphasis is given to identify and associate metrics to each goal. CompNN uses Impact Estimation Tables (IET) as the key project management tool. Each of the projects within a release has an IET. The IET is essentially a spreadsheet that lists goals vertically and project iterations horizontally. Each of the iterations is broken down into one or two solutions (think of them as design proposals) that are believed to address the goals. The structure helps to separate requirements from solutions. By documenting both requirements and project progress (measured results) per iteration, the IET becomes a combined plan and historical record of the project. Before release initiation, selected customers are invited to give input to product plans with initial goals for the release. During project initiation, spanning from one to two weeks, the most promising solutions to address the initial goals are captured and laid out for a number of future iterations in the IET. CompNN's CTO explained: "This plan helps us ensure adherence to an overall technical strategy for product development."

Evo at CompNN is a highly scheduled process where weekly iterations follow a fixed schedule (called the "Evo-week") that defines each role's responsibilities per day. For example, on Fridays there is a management design review meeting for iteration N, on Mondays stakeholders uses (tests) the product of iteration N-1.

As an extension to Evo, CompNN has included an additional role; the product management team (PMT). This is a group of five persons with responsibilities within marketing, sales, dedicated support and key account management, thus being the role having best access to knowledge of the market and the customers [9]. In the release projects they have the responsibility of appointing customers and maintain the relationship throughout the release.

During the second Evo-release, CompNN extended the adoption of Evo by introducing the green week. This is a week dedicated to error correction and improvements responding to the requirements/needs of other stakeholders, thus not directly engaging customers. This is according to the Evo methodology

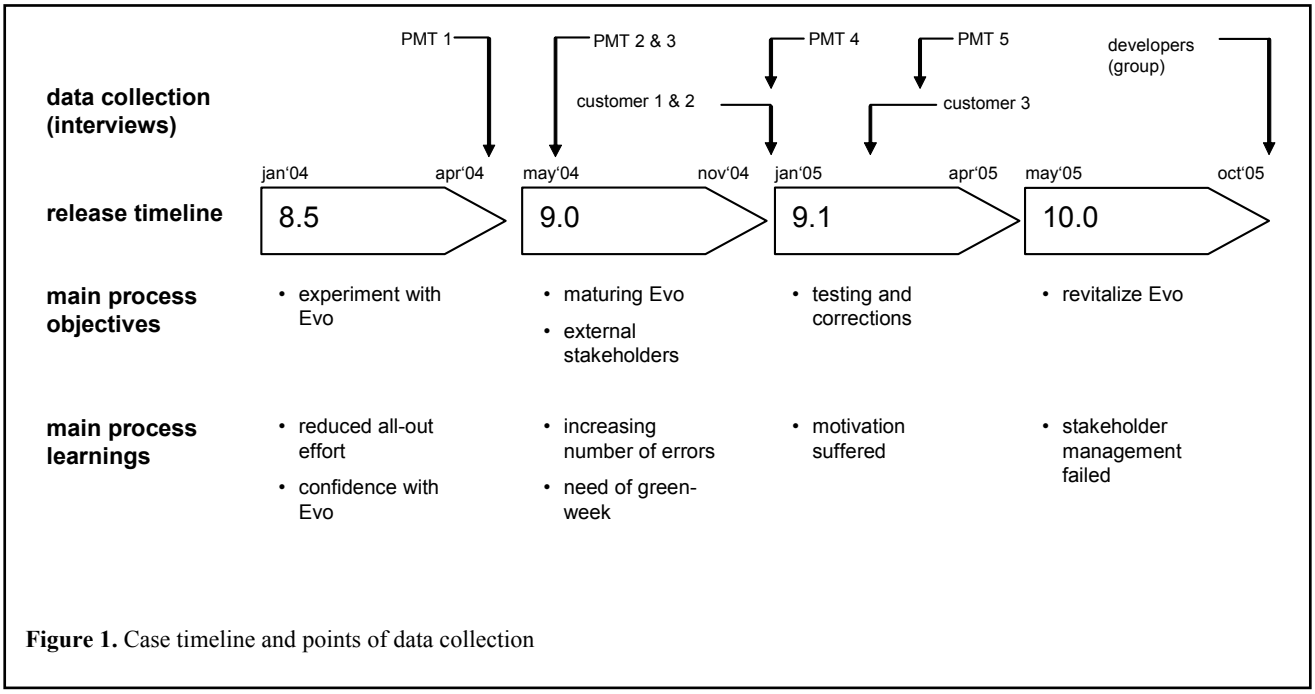

Figure 1. Case timeline and points of data collection

which recommends using about 20% of the time to improve qualities not directly related to end-users.

Not a part of Evo but nevertheless a great aid is the Continuous Integration framework (CI). This is a set of well integrated tools for code management, automated tests and builds that enable swift production of product versions ready for stakeholder testing.

3.4 The four Evo-releases studied

Our case study of the introduction, use and maturing of Evo covers four ProdNN releases over two years. This section briefly describes each release to explain main process innovations and experiences. See figure 1 for a brief timeline.

Release 8.5: The first Evo-release started soon after a one-day workshop covering the main Evo practices, held at CompNN's main office, given by Tom and Kai Gilb. As this was the first Evo-project CompNN wanted a gentle start. It was decided not to involve any customers directly. The product management team would act as surrogate customers. In cooperation with customers PMT put up a list of improvements and innovations for the upcoming release. A release plan was defined dividing the release into four projects, each addressing a specific product module. Each project had an R&D-team of 2-4 developers and each team was appointed one PMT-representative to act as a surrogate stakeholder. All projects followed the schedule of the Evo-week, used the CI-framework and an IET. CompNN, especially R&D, soon began to see some first effects of this new practice. The amount of overtime at the end of the release period was almost eliminated and interviewees from R&D reported an increased level of comfort. Version 8.5 of ProdNN was released in April 2004.

Release 9.0: The second release started in May 2004. The main experience was positive, and CompNN decided to include real customers as stakeholders. PMT made a thorough judgment on whom to invite to ensure stakeholder competence and motivation to contribute. Three stakeholders, all with an existing and close relationship with CompNN, were asked to participate in the development process and after a brief introduction of Evo they accepted. The stakeholders were allocated to individual projects within the release. Stakeholders and developers soon became accustomed with the feedback loop which for the most part worked according to the description given in the section above. However, the development teams experienced that the fast pace due to the weekly iterations reduced the time for testing, thus resulting in an increasing number of errors. As a corrective action midways in the release CompNN introduced the green week to get time to catch up. Release 9.0 ended in November 2004.

Release 9.1: Release 9.0 was considered to be a success with respect to the feedback from customers and its effect on the perceived product quality; however CompNN realized that too many errors were introduced – even if the green weeks had helped to some point. CompNN decided to run an interim release to correct and improve ProdNN. Release 9.1 was started in January 2005. Only one external stakeholder was involved, the other projects focused solely on internal improvements. The release ended in April 2005.

Release 10.0: The fourth release being studied here was started in May 2005. After the interim release CompNN now set a strong focus on Evo – including stakeholders – and soon recovered the spirit from the 9.0 release. However, it shortly became a problem with customers dropping out of the projects for several reasons, the summer holidays being one of them. This partly brought the developers back to old practice as they were left alone. Lacking corrective directions from customers, developers included too many requirements and the release ended as an exhaustive all-out effort. The 10.0 release was completed in October 2005.

4. STUDY METHOD

4.1 Rationale

At the time this study was planned there was a great amount of uncertainty at CompNN with respect to the new process. The decision to try Evo as a development process was made promptly, based on a brief introduction of Evo. The motive to perform such a radical change in process was funded on a feeling of the need of moving on, in this case from a cumbersome and inflexible waterfall like process. No clear improvement goals or overall objectives were defined – CompNN just wanted to try a new and promising process. As seen from the researcher's point of view we had no specific process performance goals to examine nor any baseline to contrast any qualitative data and therefore decided to follow the introduction broadly to document the details of the process innovations, the process values, costs and prerequisites. With this background we decided to perform this as an open case study [31]. To collect a widest possible foundation of data we prepared for a longitudinal study, covering several releases. This has given us the opportunity to study how CompNN has used gained knowledge from one release to the next to optimize the customer engagement practice [9]. From the start of the study in January 2004 to the end in December 2005 the study covers four consecutive releases of ProdNN described in section 3.4.

4.2 Data collection

As the main source of data we have interviewed several persons filling the three most central roles with respect to Evo; the developers (6 persons), participating customers (1 person from each of the 3 companies) and all members of the product management team (5 persons). Additionally we have had numerous discussions with the QA-manager and the CTO. The timing of the interviews is depicted in figure 1. All three customers involved in the second Evo release were interviewed using an interview guide after the release had completed. All five members of the product management team were interviewed also using a guide. The first was interviewed late in the first release, the two next early in the second release, the fourth in the start of the third release and finally the fifth near the end of the fourth and latest release. Six developers from R&D were interviewed in a group interview after the completion of the fourth Evo-release. During this interview the group first drew a timeline for all the four releases on a whiteboard. Then we organized a postmortem analysis [32] asking the developers describe their positive and negative experiences with Evo. All interviews, nine in total, were taped. We sought to support the interview data with quantitative data on costs and effects of the process being studied. It turned out to be hard to obtain this type of data during or after the four releases, thus making this study a solely qualitative study.

4.3 Data analysis

All taped interviews were transcribed to form a basis for a thorough and detailed textual analysis. Interview results per interviewee-group were analyzed using constant comparison [33] to find lessons learned from the introduction of Evo at CompNN with respect to practical issues (how to), effects, costs and prerequisites. The analysis was conducted with the help of NVivo™, a software tool for analyzing textual data by tagging data with meaning and grouping analytical fragments into larger constructs. The results from the analysis of each group of interview results were used as the basis for the analysis presented in section 5.

4.4 Bias and limitations

There are several factors restricting the credibility and generalizability of this study. First of all we have collected and analyzed data from one single company.

However, this opens up for an in-depth study having been harder to perform involving more case companies (with the same amount of resources). Besides that, we only know – and have access to – this one company taking into use this particular software development method. Further on, restricting a study to a single organization puts limitations on the generalizability of the results to other organizations as there are no contrast cases. However, by providing contextual information a reader may be able to see which experiences that may be transferred to another software development organization. The most prominent contextual features of our case organization is the size and the fact that they are a provider of a standard software product with an established user group – as opposed to project or consultancy organizations. Finally, there is a threat with respect to the completeness of our study. We have not interviewed all parties affected by the process transition but we believe we have selected the most central ones that could give us the most reflective and nuanced view. To summarize; the reader should have these limitations in mind when reading the results of this study, however, our conclusions should be useful to those that find a similar context and at least they could serve as a basis for further research.

5. ANALYSIS

CompNN's motivation for increasing customer collaboration throughout the development cycle was described in section 3.2. In short, CompNN saw a need to better respond to changing requirements and to improve the attention to customers' perceived quality of use of the product.

Through the introduction of Evo, CompNN has performed an organizational change that enables the engagement of a selected set of customers more tightly into their software development process. Subsequently, CompNN has experienced effects on both of the two key knowledge transfer processes mentioned in section 2. Generally speaking, CompNN's ability to transfer knowledge from the users to the developers has improved significantly. As our following analysis will explain in greater detail, CompNN has obtained a rich set of experiences related to this process. Likewise, CompNN's ability to clarify expectations towards the involved customers has improved. We also provide detailed discussions on this issue.

5.1 Engaging customers

The way CompNN has chosen to engage customers can be characterized as advisory. Customers are not project leaders although each customer selected as stakeholder in a project has a strong voice in approving the quality of the developed software. The project leader is in control of the priorities among the goals in the IET. Another advantage in CompNN's approach is scalability. Each release involves 4-5 rather autonomous development projects with 2-5 developers. Therefore, maintaining an agile environment is relatively straightforward

and associated with modest overheads within the project group. In a bigger group, this might easily become a problem [34, 20]. Furthermore, CompNN restricts a project to only one external stakeholder (multiple stakeholders may be associated to a project during its lifetime however). The process does not restrict the number of stakeholders associated with a project however, so if it was deemed necessary, it would not be overly difficult to include more than one.

A very important task is to establish collaboration with the most appropriate customer within each project, i.e. selecting stakeholders [9]. What criteria should be used in the selection process? CompNN has allocated this task to the PMT role which must identify 4-5 customers that shall act as stakeholders. Priority is given to customers that already have a very close and long-lasting relationship with CompNN. A second criterion is customers classified as expert users. Expert users are likely to have a great direct interest in the product and the direction of its development.

Customers of the packaged product were not participating in the product development process previous to Evo. Thus, education of the relevant people is important [11]. In this case this was done quite easily and lightly. Customers were given a brief overview of the basic principles with emphasis on their tasks covering specification of desirable product qualities, evaluation and feedback. Based on our interviews it seems that this nimble approach has been sufficient. We think this is due to the customers in this case already being accustomed to software development. Additionally, an appointed customer must have a motive to participate; CompNN provides no form of payment or compensation to stakeholders. Stakeholders are not given any formal power either, the sole reason to get involved as seen from the customers' point of view is the *opportunity* to affect the course of development. All three customers interviewed agreed to this as their main motivation. Without any guarantees for control the value of the collaboration is moderate, compared to collaborations in traditional custom development projects [11]. One did even indicate that if he was to contribute more extensively, the contribution would have to be reimbursed. This is not unreasonable as taking on the role as an active stakeholder in the development project may imply an extensive resource usage. This particular customer invested only a few days during the project period. Another customer had a greater level of involvement, going so far as to set up a designated group within the organization and also consulting their own customers.

In practice, we found that as long as CompNN works directly with issues of special interest that hold a promise of business value customers will continue to evaluate consecutive releases and provide detailed and valuable feedback. We see this clearly from the developer's experiences; as long as the ongoing work affects the customer, the customer stay engaged. As soon as there is no direct interest the customer subsides. A fundamental effect is that customers have a strongly varying motivation to contribute. Thus, *monitoring and managing* the collaboration emerges as a vital task.

5.2 Stakeholders' practical roles

The stakeholder has two primary roles in a project. The first is to collaborate on the specification of detailed requirements in the form of measurable goals. These goals are subsequently added to the IET by CompNN. The second role is to test a series of product versions and provide timely feedback to CompNN. The use of ASP-based software delivery made this possible within short iterations and at low cost – no installation and setup is needed (ASP – Application Service Provider). Two customers used this option; the third did not use the product per se, but only got a list of features to comment on. In some cases other ways of exposing new features were used, for example video demonstrations. This worked well for testing end user functionality, but with the negative consequence that it became impossible to test certain system administration qualities of the product (that requires a local server installation). CompNN improvised when necessary however. One of the customers, being a server customer, requested and got a server installation, thus being able to provide this kind of feedback. It is important to note that there is a highly variable cost associated with testing and responding to the different qualities. Another premise to achieve this level of agility was the use of the CI framework for rapid building and partial, but automated testing of new versions of the product.

5.3 R&D experience

As mentioned above, customers loose interest quickly if the perceived value is reduced. Subsequently, it does not take long before the customer withdraws from the collaboration. This creates a void for the developers. Having various experiences through four release projects with strongly varying Evo compliance we see this effect clearly. Developers soon got to value the direct and informative dialogue with customers and felt an equivalent strong loss when a customer dropped out. Evidently, it just did not make any sense to provide a release when no one was there to receive it.

This close customer dialogue at a personal level had a value beyond detailed functional testing and feedback; it consequently also made the developers feel comfortable and more personally responsible for the customers. The fact that someone took a direct interest in their work was reported to be a huge motivational factor creating a new spirit that was missed before the process change. A PMT-member said: "Well, I felt that... hey - now some of the initial joy is back that we had during the first years where things was very different and where we had to work in an Evo-spirit without really knowing it" One developer even referred to it as being fun.

5.4 Specifying quality goals

One of the most distinguished features of Evo is the clear focus on quality requirements as opposed to functional requirements. A user is expected to express desired qualities of the system being developed such as "time to use" or "ease of use" and others. At the same time the customer also defines a measurable goal that can be verified through testing. In this study we see that both customers and developers report this to work as intended in some cases. On one side it gave the user the opportunity to talk "user language" not having to consider any design issues. As seen from the developer's point of view they got precise requirements with respect to performance giving them freedom do define how to realize the solution. Further on – after testing by customers – developers got clear and consistent feedback on the performance that they could use to either declare the goal as completed or needing further improvement in the next iteration. Developers appreciated this feedback as it became clear when to stop the improvement on a given quality

or feature, thus being able to move on. One of the developers commented: "...feedback from stakeholder minimizes the risk of developing something no one really needs". One concrete example may be found in Johansen's description of Evo at CompNN [29] where the goal "Usability.Productivity: Time for the system to generate a survey" used to have a performance of 7200 seconds. The goal was set to be 94 seconds and the result after one of the releases was measured to be 15 seconds. Having obtained this performance the goal was defined as complete. However this practice only worked when the premises were in place, that is, when the user was available and when the goals were clear and measurable. When this practice worked as intended, stakeholders described CompNN a "sensitive to their requirements" and greatly appreciated the close and direct communication during, some times, painstakingly detailed discussions. As this cooperative model involves just a few customers out of many it makes it even more important to select/invite stakeholders that may have requirements that other users potentially share. Including customers this closely into the development process may be a challenge as customers will see the process from their viewpoint. It is therefore extremely important that CompNN makes the final decisions on what requirements to focus on and how to resolve them as they need to have the total market in mind – both existing and future customers. We see from one of the customer interviews that the customer had a close and detailed dialogue with respect to a specific feature that was not included in the final release. In this case CompNN made a strategic decision that deviated from the customer's desires. For this particular example the customer understood CompNN's decision, however this type of trade-offs puts an extra dimension to the customer relationship management as development is rather by advice by customers – not by direction. To handle the practicalities and to be able to see the total development process in relation CompNN used IETs. Developers reported this tool only to be a visualization of progress to them. The project leaders used it more extensively as a tool to prioritize requirements. The IET is simply an excel spreadsheet, however some of the project leaders indicate a future need of having a more elaborate tool to handle this new amount of information better and more efficiently.

5.5 Testing

Evo, by principle, is a process that incorporates testing during the entire course of a development project; the developers performing unit-testing at each step and the stakeholder perform user acceptance tests. Thus, the system should already be tested completely at the end, just prior to the final release. However, the developers underline the need of a more comprehensive system test before market release as the system has grown in complexity and the fact that testing is never the same as real use. Further on, removing the need of a final system test requires that all aspects of the system have been tested under ways which actually is not happening due to time and resource limitations.

5.6 Experience with the weekly schedule

Both PMT members and especially developers commented that weekly iterations may be too short to efficiently cope with the duties prescribed by each step (develop, unit-test, build, deployment, user-test, feedback, feedback evaluation and iteration planning). A certain level of reluctance should come as no surprise, software development is mentally demanding work that depends upon deep concentration [1]. Additionally, it correlates strongly with the size of a projects development team – two developers may not be able to produce enough value for one iteration – biweekly iterations may be a better option for the smaller teams, as one of the developers directly suggests. Further on, developers refer to the internal testing to be time consuming – eating up the time available for developing new features.

As seen from the developers' point of view it was time consuming to maintain the CI-framework (primarily, this involved maintaining the tests). A developer commented that this problem probably will grow with the code size. The adoption of the green week is a clear example of extra overhead as all time are spent on finding and correcting errors and improving performance. However, this may be equally valuable as just adding new features.

We note the fragility of the process. All roles – developers, project leaders, PMT and customers need to stick to the week schedule and diligently tend their responsibilities. If any person or group fails to do so it will strongly affect others directly and thus hamper the whole process. One member of the PMT also underlines the need of process compliance and rigor. Another effect of Evo, only reported by the developers, is that the continuous focus on direct customer value weakens the focus on engineering handcraft such as thorough general design. This may lead to development shortcuts to reach short-termed (next week) goals potentially degrading the overall architecture of the total system. The prevention of architecture erosion is a fundamental challenge in agile development [35]. AT CompNN the CTO has an especially important role to monitor the software architecture and recommend initiatives to maintain a sufficient level of quality – thus being the counter pressure for architectural erosion.

5.7 Stakeholder engagement monitoring and management

Several of the interviewed developers emphasized the importance of PMT as monitors and managers of the stakeholder-developer relationship. As we noted in section 5.3 a project experiences the sudden absence of a stakeholder as highly frustrating and devastating. Ideally, when a stakeholder drops out, a PMT resource must either promote a new customer to the role or step into the customer role to keep the vital deploy-test-evaluate loop running. Otherwise, the continuity of the development process may suffer [1]. We see that PMT has been an overloaded resource as the five members also have a wider responsibility including sales, marketing, key-account management and overall product strategy planning. All of the five PMT-resources interviewed blamed a high level of stress and a lack of resources for this. For PMT the direct cooperation with the project leaders and the management of customers came on top of other tasks, this was described as stressful and as we see especially from the 10.0 release this caused PMT not to fulfill their role, bringing some of the projects into trouble. The developers stated directly and clearly that PMT should be extended with more resources. Although it was already identified as a potential problem in the first and the second Evo releases, they still failed to correct this in the last release. We cannot say for sure if this was directly communicated to management, but in interpretation of our data we think this is

explained in R&D's reluctance of exposing higher costs of the process to management. Besides selecting customers initially and potentially under way, PMT also have a responsibility of making strategic choices of where to direct the development. In cooperation with the project leaders and the CTO they make this kind of strategic decisions, relieving developers from this. However we saw that when a stakeholder left a project and the dedicated PMT resource was unavailable (due to other tasks) developers were forced to make decisions by themselves. This happened in the 10.0 project during the summer holidays and resulted in the inclusion of too many requirements, more than there were resources to handle. This became apparent only close to the release date and led to an intense period with a lot of overtime work to carry the project off.

5.8 Development process visibility

In our interviews with the five PMT members they made a strong note of the improved visibility of the development process. They had previously been "outsiders" in this respect, playing a role in the initial rigorous requirements specification but left out after that. One PMT resource noted: "We did not see any visible results until the release was delivered to the market". Now, as the PMT has an active role in development projects they have both the opportunity to comment on results underway and to discuss and prepare customers on coming features and improvements in front of release. Developers refer to plans of utilizing this process visibility further by including the documentation writers underway, thus being able to complete correct documentation at the time of release.

5.9 Scoping and estimation

The fact that CompNN now makes scoping and estimation in each iteration of the Evo process, also means that product marketing can give less precise promises of the expected functionality in the coming release. A member of the PMT explains: "It is hard to promise what to deliver at a given date. We are able to give committing promises concerning coming features only a month before the release date. This is off course frustrating as we think it is important that the customers and market are updated". This is a fundamental aspect of evolutionary development as estimation is done per iteration and not up-front. However, PMT members commented that the benefit of a tangible product outweighed this disadvantage.

6. CONCLUSIONS

Through their process innovations CompNN has enabled us to reveal new knowledge concerning customer engagement in an agile software product development context. Our qualitative study helps to shed light upon the complex problem area of balancing the attention to customer demands while constructing software – arguably one of the world's most challenging endeavors. This study and our conclusions rely on one single company, however, we believe that the key lessons learned are valid and useful to others software companies, especially those with a product focus and a continuous development process to improve and develop the product.

It should be made clear that the process innovations in CompNN have not been a managed process as such. The introduction of Evo occurred due to a need for a change and the incidental encounter with Tom Gilb. However, through their incremental adoption of Evo and the introduction of the PMT role, CompNN is probably a typical example of process innovation in industry. Some things happen by intent and some things by chance. However, CompNN has both exploited existing competence but also actively embraced the exploration of new possibilities [36]. Mindful engagement of customers is a prerequisite for innovation in product development, it is not an option [37].

Our analysis identifies a number of prerequisites for succeeding with this approach. Proactive stakeholder management is the foundation. CompNN operates in a dynamic market in which long-term planning is unrealistic. This applies equally to the customers which are inherently unpredictable in their role as partner in the collaboration. CompNN depends upon mutual benefits in the collaboration to maintain motivation and contribution. As our analysis shows, lack of continuity has significant bad effects on the performance. In turn, achieving these mutual benefits depends upon selecting relevant customers with sufficient expertise. These are all volatile and difficult factors. Thus, the capability to constantly review and manage the selection of stakeholders is critical. Furthermore, our analysis shows that Evo is a highly demanding process with respect to personal discipline and professional behavior. Due to the frequent iterations there is little room for unrestrained activity. All roles must be meticulously filled. Additionally, some sophistication in technical infrastructure, such as CI, IETs and ASP-based software delivery, has been extremely valuable, and probably essential in CompNN's use of Evo.

Our analysis shows that CompNN has achieved a number of benefits as a result of Evo and the introduction of the PMT. First, close customer cooperation has a highly motivating effect on the developers. Second, developers' confidence has increased as a result of continuous settlement of expectations in that stakeholders assist in the prioritization of goals. The direct cooperation with users is a positive experience for the developers as it increases the quality of the communication and leads to improved understanding of the real business problems. Third, Evo has increased the visibility of the process internally in the organization and externally among the stakeholders.

Adopting agile customer engagement practices has incurred additional costs. Our analysis emphasizes the extra overhead in actually running the process and the human resources required. Essentially, each Evo week is a complete development process, spanning a number of phases that demands frequent changes of context and thus occupy significant resources. Also, the technical infrastructure, being a prerequisite, is costly. Furthermore, the analysis shows a significant cost incurred by the continuous need to maintain it.

It is natural to consider increased exposure to risk as a cost. First, short iterations with insufficient attention to management and process compliance increase the fragility of the software process. Our analysis shows that the 10.0 release in particular exposed CompNN to considerable risk in that the developers were left with much higher workload than anticipated – thus being a good example of lack of process compliance and the consequent effects. Secondly, engaging in this kind of strong cooperation with a small selection of customers also means a reduced capability to capture the needs of other, non-appointed customers. For example, it is critical that internal stakeholders are appointed in a conscious manner to take care of the full set

of quality drivers pertaining to the software product (specified in product strategies etc.)

Our study uncovers possibilities for further research in this area. It would be valuable to compare these findings with similar investigations in other companies. Combined, these findings might be used as a motivation for more specialized and rigorous studies of effects on customer satisfaction for example. Our study does not reveal information on any effects the process innovation might have on the non-included customers observed quality of the product. Ultimately, studies should be conducted to quantify the benefits and costs of agile customer engagement.

7. ACKNOWLEDGEMENTS

We are greatly thankful to the people at CompNN that has contributed to our study and even let us talk to their customers. We would also like to thank the Research Council of Norway for the funding of this work under Grant 156701/220.

8. REFERENCES

1. Reel, J. S., *Critical Success Factors In Software Projects*, in *IEEE Software*. 1999. p. 18-23.
2. Ceschi, M., Sillitti, A., Succi, G. and De Panfilis, S. Project management in plan-based and agile companies. *IEEE Software*, *22*, *3* (2005), 21-27.
3. Abrahamsson, P., Salo, O., Ronkainen, J. and Warsta, J. *Agile software development methods - Review and analysis.* VTT Publications 478, VTT Electronics, 2002.
4. Cohen, D., Lindvall, M. and Costa, P. *Agile Software Development A DACS State-of-the-Art Report.* Fraunhofer Center for Experimental Software Engineering Maryland and The University of Maryland, 2003.
5. Lindvall, M., Basili, V., Boehm, B., P., C., Dangle, K., Shull, F., Tesoriero, R., Williams, L. and Zelkowitz, M. Empirical Findings in Agile Methods. In *Proceedings of XP/Agile Universe (*Chicago, IL, USA, 2002). Springer-Verlag GmbH.
6. Gilb, T. *Competitive Engineering: A handbook for systems engineering, requirements engineering, and software engineering using Planguage*. Elsevier Butterworth-Heinemann, 2005.
7. Jørgensen, M. and Moløkken-Østvold, K. How large are software cost overruns? A review of the 1994 CHAOS report. *Information and Software Technology*, *48*, *4* (2006), 297-301.
8. Johnson, J. Turning chaos into success. *Software Magazine*, (1999), 30-34,39.
9. Keil, M. and Carmel, E. Customer-Developer Links in Software Development. *Communications of the ACM*, *38*, *5* (1995), 33-44.
10. Tait, P. and Vessey, I. The Effect of User Involvement on System Success: A Contingency Approach. *MIS Quarterly*, *12*, *1* (1988), 91-108.
11. Axtell, C. M., Waterson, P. E. and Clegg, C. W. Problems integrating user participation into software development. *International Journal of Human-Computer Studies*, *47*, *2* (1997), 323-345.
12. Boehm, B. and Turner, R. Management challenges to implementing Agile Processes in traditional development organizations. *IEEE Software*, *22*, *5* (2005), 30-39.
13. Lin, W. T. and Shao, B. B. M. The relationship between user participation and system success: a simultaneous contingency approach. *Information and Management*, *37*, *6* (2000), 283-295.
14. Mathiassen, L., Ories-Heje, J. and Ngwenyama, O., *Improving software organizations: From principles to practice*, in *The agile software developement series*. 2001, Addison-Wesley Professional. p. 368.
15. Rogers, E. M. *Diffusion of innovation*. Free Press, 2003.
16. Gallivan, M. J. and Keil, M. The user-developer communication process: a critical case study. *Information Systems Journal*, *13*, *1* (2003), 37-68.
17. Vredenburg, K., Mao, J. Y., Smith, P. W. and Carey, T. A survey of user-centered design practice. In *Proceedings of Conference on Human Factors in Computing Systems (*Minneapolis, Minnesota, USA, 2002). ACM Press New York, NY, USA.
18. Wong, E. Y. W. and Tate, G. A Study of User Participation in Information-Systems Development. *Journal of Information Technology*, *9*, *1* (1994), 51-60.
19. Heinbokel, T., Sonnentag, S., Frese, M., Stolte, W. and Brodbeck, F. C. Don't underestimate the problems of user centredness in software development projects - There are many! *Behaviour & Information Technology*, *15*, *4* (1996), 226-236.
20. Hansson, C., Dittrich, Y. and Randall, D. Agile processes enhancing user participation for small providers of off-the-shelf software. In *Proceedings of Extreme Programming and Agile Processes in Software Engineering, Proceedings* 2004).
21. Boehm, B., *Get Ready for Agile Methods, with Care*, in *IEEE Computer*. 2002. p. 64-69.
22. Glass, R. Matching Methodology to Problem Domain. *Communications of the ACM*, *47*, *5* (2004), 19-21.
23. Kujala, S. User involvement: a review of the benefits and challenges. *Behaviour & Information Technology*, *22*, *1* (2003), 1-16.
24. Bødker, K., Kensing, F. and Simonsen, J. *Participatory IT Design: Designing for Business and Workplace Realities*. MIT Press, 2004.
25. Platt, A. and Warwick, S. Review of soft systems methodology. *Industrial Management + Data Systems.*, *95*, *4* (1995), 19-22.
26. Mumford, E. The ETHICS Approach. *Communications of the ACM*, *36*, *4* (1993), 82-83.
27. Hartwick, J. and Barki, H. Communication as a dimension of user participation. *Ieee Transactions on Professional Communication*, *44*, *1* (2001), 21-36.
28. Dagnino, A., Smiley, K., Srikanth, H., Anton, A. I. and Williams, L. Experiences in applying agile software development practices in new product development. In *Proceedings of Proceedings of the Eighth IASTED International Conference on Software Engineering and*

*Applications, Nov 9-11 2004 (*Cambridge, MA, United States, 2004). Acta Press, Anaheim, CA, United States.

29. Johansen, T. Using evolutionary project management (Evo) to create faster, more userfriendly and more productive software. Experience report from FIRM AS. In *Proceedings of International Conference on Product Focused Software Process Improvment (*Oulu, Finland, 2005). Springer Verlag.
30. Boehm, B. A Spiral Model of Software Development and Enhancement. *IEEE Compuer*, *21*, *5* (1988), 61-72.
31. Yin, R. and Campbell, D. T. *Case Study Research*. Sage Publications Inc, 2002.
32. Birk, A., Dingsøyr, T. and Stålhane, T. Postmortem: Never Leave a Project without It. *IEEE Software*, *19*, *3* (2002), 43 - 45.
33. Seaman, C. B. Qualitative methods in empirical studies in software engineering. *IEEE Transactions on Software Engineering*, *25*, *4* (1999), 557-572.
34. Boehm, B. and Turner, R. *Balancing Agility and Dicipline - A Guide for the Perplexed*. Addison-Wesley, 2004.
35. MacCormack, A., Verganti, R. and Iansiti, M. Developing products on "internet time": The anatomy of a flexible development process. *Management Science*, *47*, *1* (2001), 133-150.
36. Dybå, T. Improvisation in small software organizations. *IEEE Software*, *17*, *5* (2000), 82 - 87.
37. Carmel, E. and Becker, S. A Process Model for Packaged Software Development. *IEEE Transactions on Engineering Management*, *42*, *1* (1995), 50-61.