Empir Software Eng (2006) 11: 203–225
DOI 10.1007/s10664-006-6402-8

# Integrating agile software development into stage-gate managed product development

**Daniel Karlström · Per Runeson**


**Editor:** Marvin Zelkowitz

**Abstract** Agile methods have evolved as a bottom-up approach to software development. However, as the software in embedded products is only one part of development projects, agile methods must coexist with project management models typically of the stage-gate type. This paper presents a qualitative case study of two large independent software system projects that have used eXtreme Programming (XP) for software development within contexts of stage-gate project management models. The study is comprised of open ended interviews with managers as well as practitioners, followed by a structured, fully traceable, qualitative analysis. We conclude that it is possible to integrate XP in a gate model context. Key issues for success are the interfaces towards the agile subproject and management attitudes towards the agile approach.

**Keywords** Agile methods · Stage-gate project management · Qualitative study · Extreme programming · Case studies

## 1. Introduction

Agile methods have evolved as a new approach to developing software products (Agile Manifesto, 2001). The methods have been successfully implemented in small to medium sized projects, and the principles of the agile methods have inspired software developers in wide-ranging application domains (e.g., Vanhanen and Kähkönen, 2003; Fuqua and Hammer, 2003; Rasmussen, 2003; Schuh, 2001; Grenning, 2001; Poole and Huisman, 2001; Murru et al., 2003; Sharp and Robinson, 2004; Karlström, 2002). The agile methodologies offer down-to-earth approaches to software development that focus on simplifying and improving the software development process, making the customers, the developers and the final product the most important (Agile Manifesto, 2001). A summary of the main principles involved in the agile methods as formulated by the agile manifesto is given in Table 1.

D. Karlström (✉) · P. Runeson
Department of Communication Systems,
Lund University, Box 118, SE-221 00 Lund, Sweden
e-mail: daniel.karlstrom@telecom.lth.se

**Table 1** The agile manifesto (Agile Manifesto, 2001)

1. Individuals and interactions over processes and tools.
2. Working software over comprehensive documentation.
3. Customer collaboration over contract negotiation.
4. Responding to change over following a plan.

In real-life, however, software development projects are not isolated activities. They usually exist as sub-projects in an environment composed of hardware development, marketing, production planning etc. which all must be managed and coordinated concurrently (Cooper, 2001). Furthermore, software development seldom starts from scratch; legacy software constitutes the core of new products to which new functionality and other types of customer value are added, or legacy software is moved to new platforms. The traditional approach when managing a complex development project situation is to use a project management model (PM) such as Cooper's Stage-gate model (Cooper, 2001), see Fig. 1. The PM gives support not only for the communication within the project, but also for decision makers sponsoring the project or acquiring the outcome of the project. The concepts of a PM are not connected to any specific domain; these models are sufficiently generic to be applied to any kind of project. However, PMs currently tend to have a basic sequential structure, i.e., distinct phases of pre-study, feasibility study, execution and conclusion. From a management point of view this is quite natural. Between each stage there is a decision gate at which the project must meet predetermined demands such as producing certain documents. The project sponsors can then make the decision whether the project is to proceed to the next phase, stay in the previous phase or be abolished.

Gate oriented PMs are compatible with maturity-oriented software process improvement work, using for instance the CMM framework (Paulk et al., 1993). Both PMs and the maturity models aim at defining structures, roles and responsibilities within the organisation, but do not intend to actually dictate how the actual technical project work is to be performed. The agile methodologies can be considered a reaction to these maturity-oriented approaches within the software development process field, but we do not believe that the two schools are contradictive. Software projects using agile methods must be able to exist within

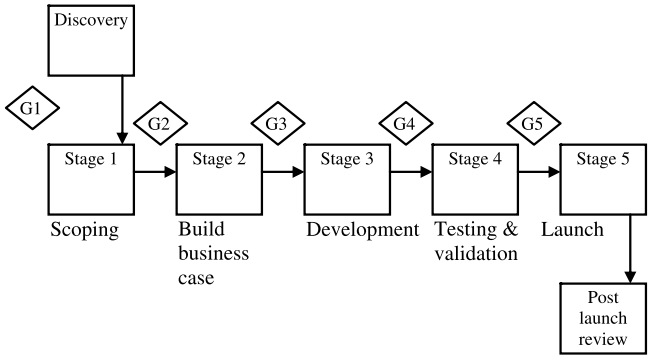


**Fig. 1** Stage-Gate model™ (Cooper, 2001)

environments structured using gate oriented PM. Mature organizations add not only rigour and order to a chaotic environment, but also certain characteristics similar to a successful agile development project (Paulk et al., 1995; Paulk, 1999; Paulk, 2001).

Gate oriented PM models have already been integrated with more iterative development models in general (Gilb, 1998) and the Rational Unified Process (RUP) in particular (Royce, 1998). A proposal on how to integrate an instance of a gate model with various development methodologies, has been presented (Wallin et al., 2002), although only at an overview level. The research presented by Wallin et al. presents the overall concepts and problems involved in combining a project management model with several different software development processes, including RUP and eXtreme Programming (XP). The presentation, however, is lacking in detail of how the interfaces are defined and how the different processes need to be modified.

Software projects using agile methods have indeed been executed in traditional PM environments, but the main characteristics of the PM are not originally designed for the management of software projects using agile methods, and their advantages are therefore not utilised in the management of the project. The agile development has in these cases had no effect on the project management and overall management of the project despite demonstrating success on the development level.

Rainer et al. present interesting work on "buy in" for software development process improvement (Rainer et al., 2003). This is an especially relevant issue for agile methods as these are often introduced bottom up in contrast to top down for most other software development processes. They present the importance of showing local effectiveness of the process to developers in contrast to metrics from other locations.

Boehm and Turner (Boehm and Turner, 2004) discuss contrasts between plan driven software development and agile approaches at length in their book. Gate models in general are a form of plan driven models, defined on a higher level of abstraction than those discussed. Those used by the companies involved in this study, however, are more specialised. Interestingly one of the conclusions drawn in the book is that future projects will need both agility and discipline, which can be implemented by containing the agile development methodology within the gate model.

XP has been structured into *values*, *principles* and *practices*. Values are at a high level of abstraction and are not specific to software engineering or XP (Beck and Andres, 2004). The fundamental values presented by Beck are Communication, Simplicity, Feedback, Courage and Respect, but it is also contended that a team needs further values to function. The principles are intended to bridge the gap between the abstract general values and the detailed specific practices. The principles, in contrast to the values will differ depending on the domain that you are in. The principles presented in XP are humanity, economics, mutual benefit, self similarity, improvement, diversity, reflection, flow, opportunity, redundancy, failure, quality, baby steps and accepted responsibility. For details on the specifics of the values, principles and practices of XP, please refer to Beck (Beck and Andres, 2004).

XP is most frequently presented as the 12 detailed practices that are common knowledge in the Software Engineering community by now. Presenting XP as composed of 12 practices is similar to describing a car as composed of four wheels. While the description is true and the wheels really are an important part of the car, there are a many crucial details omitted. Simply quoting these practices might give an

insight in what the team members were doing, but would not describe the major factors that make XP work.

In this paper we investigate the experiences from integrating agile teams in traditional project management models. Two cases are studied, both within large system development companies, comprising both software and hardware development. Ericsson Microwave Systems (EMW) is in the domain of airborne defence radar systems and ABB in industrial control systems. In both cases the agile method used was XP (Beck and Andres, 2004) while each company used their respective corporate variant of the gate model. We present and motivate a feasible approach for this investigation, using qualitative research methodology (Seaman, 1999). As a result of this research approach, many interesting observations regarding the specific effects of Stage-Gate management as well as agile methodologies, and their combination are made. These are presented in the paper in an as structured fashion as the methodology allows. A shorter, more practice oriented presentation of the cases together with an additional case has recently been published (Karlström and Runeson, 2005).

This paper is organised as follows. In Section 2, the study scope is presented. In Section 3, background information about the companies and development teams involved is presented. The study design is presented in Section 4 and study validity in Section 5. Finally, the results are presented in Section 6 and a summary is provided together with the conclusions in Section 7.

## 2. Study Scope

The study aims to investigate the effects of introducing an agile software development process within a traditional stage gate model style project management system. The study specifically aims to identify the main problems with this combined approach as well as key success factors. The specific projects studied were using XP within their own variants of the gate model. To narrow the scope of the investigation, it was decided to focus on the decision point gate 3 (or G3 as shown in Fig. 1), i.e., deciding on start of full-scale development, as this is considered a crucial point in projects. Here it is decided whether the project should proceed into full-scale development and a large increase in resources is to be allocated. The teams that were investigated however were far beyond G3 and as such could only describe this as they remembered it. At the time of the investigation, the EMW team had completed their implementation, had subsequently been split up and had its members reassigned to various assignments. The ABB team was late in the implementation stage of the product. Furthermore, specific focus was set in the study on the interfacing between the XP team and the surrounding environment as this is where contrasts between the two were thought to appear most clearly.

## 3. Case Contexts

The context of the case study is presented according to the recommendations by Kitchenham et al. (Kitchenham et al., 2002) stating that the experimental context needs the three elements *background information*, *discussion of research hypotheses*,

and *information about related research*. The two former are presented below, while the latter is presented in the introduction.

### 3.1. Background Ericsson Microwave Systems Case

Ericsson Microwave Systems (EMW) is a part of Ericsson global communications company and is developing systems for radar and microwave communication. The particular section that is studied in this research constructs airborne military radar systems. The company culture is influenced by the military application domain, as they are used to requirements on document-driven development processes. EMW has about 2000 employees and Ericsson as a whole has around 50 000 employees.

The organisation of EMW is of a matrix type, with line management and functional management. The functional management is still structured according to the physical parts of the radar system when it was constructed completely in hardware. The system is moving more and more towards software solutions and as a part of this, how best to structure the functional organisation is becoming unclear.

The team that was studied completed two projects using agile methods. The first was a target tracking function within the radar system. This function is relatively isolated related to the overall system and communicates with relatively few other parts of the system. The second function was a data integration function, communicating with virtually all parts of the system. The team size varied between 6 and 10 people during the course of the projects and during a part of the project the team was split into two.

When the team was formed in order to commence the first project they were eager to try a new methodology. At the time XP was frequently mentioned in software engineering media and the team thought it would suit them well. Management were indifferent to the suggestion and the team decided to go ahead. The team used no external resources to introduce the method. Instead they studied the literature available in a workshop format and applied the methods to their specific situation.

The project management model used at Ericsson is known as PROPS (Mulder, 1997). This model is defined on the corporate level, is very similar to Cooper's Stage-Gate model (Cooper, 2001) and allows for a large degree of flexibility in selecting the actual technical development method. They also have a development process on a more technical level integrated in the gate model, which is company-tailored. It was, however, unclear and unplanned how it would work together with XP.

The EMW context can in summary be characterized by its military background and the corporate level gate model. The introduction of XP was bottom-up, without explicit management support.

### 3.2. Background ABB SCADA Case

ABB SCADA is a part of Asea Brown Boveri (ABB) global company with more than 100 000 employees in total. The development facility studied consists of more than 100 developers in total and produces hardware and software for automated industrial control systems. The organization studied was integrated in ABB through acquisition

some 10 years ago. The company culture has its roots in a small, but steadily growing software development company.

The project studied is a part of the matrix organisational structure with both functional and line management. ABB also has a system of technical and market product managers for each product that work in cooperation with the developers. The studied development team consisted of eight people, but only six of these were part of the XP group. The other two were located off-site and used a more traditional development approach to complete tasks for the project as they were only on site about once a month.

The project members and the project leader back in 2002 were inspired by the local SPIN[1] (Software Process Improvement Network) on XP and decided to try the method. An external consultant was hired for the initial introduction of the method. The compliance with the ABB gate model, was not accounted for at the start, but a study by ABB corporate research was conducted and resulted in a part of the results presented by Wallin et al. (Wallin et al., 2002). The ABB stage gate model is also similar to Cooper's Stage Gate model (Cooper, 2001), although more integrated with the technical processes.

The ABB context can in summary be characterized by its small and growing company background, the corporate gate model with a technical focus. The introduction of XP was supported by the management and facilitated by using external consultants.

## 4. Study Design

The case study performed is of the applied research type, implying that the purpose is to understand the nature of and the problems present within a certain context in society (Patton, 2001).The study takes a broad view within the area of integrating agile methods into gate models, the scope of which has been described in Section 2. The units of analysis are individual members at different positions in the organisation, the teams that these people are organised in and the organisation as a whole, including its process descriptions. The study is focused on the experiences gained during the time that the teams were using agile methods in the gate model context compared to previous experiences within the same organisation without agile methods.

### 4.1. Subjects

The subject sampling strategy was to interview a sample of the teams involved and their management in the immediate surrounding. A few of the people, however, were unavailable at the time of the study. All participants attended the interview voluntarily and all data were treated strictly confidentially after the interview occasions. In total, nine people were interviewed as presented in Table 2. Among the development teams, both pioneers and adopters of the methodology were interviewed.

[1] SPIN-syd, South Sweden Software Process Improvement Network, http://www.spin-syd.org.

**Table 2** Interview subjects in the study

| ABB | EMW |
|---|---|
| Department manager | Department manager |
| Subproject manager | Developer |
| Developer | Developer |
| Developer | Developer |
| Developer | |

## 4.2. Research Strategy

Based on the low epistemological level within software engineering in general, and specifically in the area of integrating agile methods in software engineering, we approach the research using a qualitative research strategy (Patton, 2001; Robson, 2002; Yin, 1994; Seaman, 1999). Inspired by Patton's classification of Farming Systems Research and Development (FSRD) (Patton, 2001), we elect to study the software engineering environment as a complete dynamic system based on the following observations. Software engineering research and practice...

... is a team effort.
... is interdisciplinary.
... takes place in the field.
... is collaborative.
... is comprehensive.
... is inductive and exploratory.
... begins with qualitative description.
... is sensitive to context.
... is interactive, dynamic and process oriented.
... is situationally responsive and adaptive.

These are the characteristics defined up by Patton for another area of research, from this we apply the same observations on software engineering and thereby conclude a similar research strategy.

## 4.3. Research Methods

The main source of information in this investigation is the interviews performed with the development team members and the management in proximity to the teams in the studied organisations.

The interviews were performed as semi-structured interviews (Robson, 2002), more in the form of a discussion, using the interview instrument as a guide of areas available to discuss. Interesting facts mentioned were followed up immediately with each subject instead of strictly following the instrument. The interview instrument was constructed by two researchers and was adapted slightly as the interviews progressed. Adaptations were primarily made with the purpose of gaining further information about statements made by previous subjects. The interview instrument, provided in Appendix A, consists of the following main sections:

- Introduction
- Personal, group and company background
- General experiences of introducing XP

- Experiences of combining XP and gate models
- Ending

The interviews were approximately one hour in length with two researchers interviewing one subject. Some notes were taken during the interview but the main form of documentation was the sound recording intended for transcription as a part of the analysis.

### 4.4. Analysis

Following is a complete description and discussion of the analysis steps taken in the study, an overview of which is given in Fig. 2.

During the course of the analysis, information takes on several forms at different levels of abstraction. The subjects act in and observe the reality within their organisation. Their reflections and observations on the area of research are discussed in the semi-structured interviews and recorded as an audio file. These recordings are then fully transcribed and interesting quotes are identified. Next, quotes are coded and grouped to support each other and finally individual results are identified from these groups. The grouping was conducted using plain tables in a word processing product.

The different forms that the information takes in this investigation are numbered 1 to 7 in Fig. 2. The first form is the actual event in the study context. The second form is the subject's perception of the event, and so on as labelled in Fig. 2. Each change in form is due to some kind of treatment and we refer to this change in form as a transition. The transitions are denoted α through ζ in Fig. 2 and the subsequent tables. Each transition introduces additional validity threats to the results, depending on the treatment performed in the transition. Treatments and the validity threats introduced are summarised in Table 3. Countermeasures to these validity threats are addressed in Section 5. These countermeasures can also be seen in part throughout the study design, as the one of the goals of a good study design is to minimise the effects of threats. Some of the transitions in the analysis were performed redundantly by two researchers independently of each other. The results of the individual analyses in these transitions were compared and aggregated into a common analysis result. This is indicated by a star next to transitions involving redundancy in Tables 3 and 5.

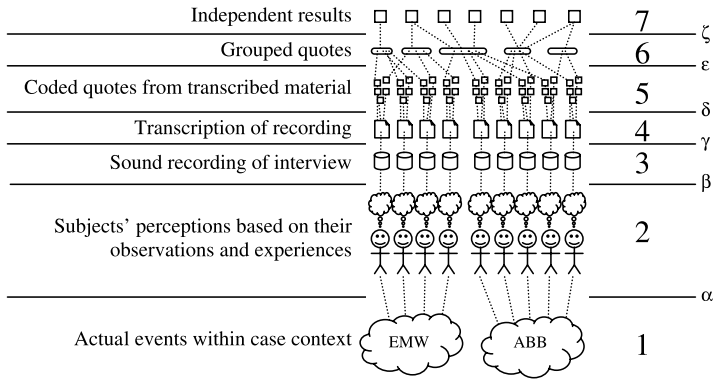


**Fig. 2** Forms and transitions in the research analysis process

The analysis performed in the study is of the inductive type, implying that patterns, themes and categories of analysis come from the data itself in contrast to being imposed prior to the data collection, as is the case in deductive analysis. All approaches used in inductive analysis according to Patton have been considered in the analysis, except for the synthesising of studies (Patton, 2001). This is however currently under preparation as further work is being planned. The main categories of analysis identified in the study were:

- Gate model and its adaptations
- Communication – horizontal, vertical, internal
- Planning and estimating within the project
- Continuous feedback
- Quality aspects
- Technical aspects
- Attitudes

The coded data were stored in tables together with references to the individual interview statement that the data was based on in order to ensure full traceability. These tables were then used to identify results across interview subjects and companies and draw the final conclusions.

In addition to the interview data, other information sources in the study include archival analysis of internal documents, including process description documents as well as experience documents, written by the teams to summarise their use of the new methodology, and informal information gained through conversations outside the interviews.

### 4.5. Validation

The preliminary results from the study were presented back to the teams in a workshop and their opinions were collected and any misinterpretations were resolved. Final conclusions were based on all the gathered information.

The interview study was performed in a short time space, three days in total, but the companies in the study are involved in a long-term cooperation with the research

**Table 3** Transitions between information forms in the research process (*indicates redundancy in transition)

| Transition | From form | To form | Treatment description | Threats |
|---|---|---|---|---|
| α | 1 | 2 | Native observations | Misconceptions, misunderstanding, lack of objectivity |
| β* | 2 | 3 | Interview | Misunderstandings, omissions |
| γ | 3 | 4 | Transcription | Audibility of source |
| δ* | 4 | 5 | Identifying quotes and coding | Misunderstanding, misinterpretation of intent |
| ε* | 5 | 6 | Grouping | Incorrect grouping due to effects of previous threats |
| ζ* | 6 | 7 | Identifying specific results | Incorrect results due to effects of previous threats |

group and there is therefore much tacit knowledge already present, both generally, concerning the overall company organisation, products and working climate, as well as specifically the technical development details within the teams concerned.

## 5. Study Validity

The qualitative strategy selected involves using the researcher himself as the instrument for the research. A short discussion of the effects of the particular researchers involved is therefore appropriate (Robson, 2002).

The main author has been involved in industrial product development since 1993 and has been performing research within the area for over four years. The main focus of research has been software process and software process improvement through involving developers more in how they work and improve their work. He has followed the agile movement as it has developed from a small community to a global phenomenon. Methodologically, he has used both empirical experiments and qualitative case studies extensively in his research.

The second author has worked with a traditional process modelling and quality assurance perspective, both in industry and research. The main focus is processes for verification and validation, although the whole process is within the interest scope. He has gradually adopted ideas from agile methods, through critical analysis of its effects. He uses a wide range of research methodologies, both qualitative and quantitative.

To reduce the threats to validity of the study, countermeasures are taken in the study design and during the whole study. Further, an analysis of threats to the validity and corresponding strategies are presented below, using the Lincoln and Guba model (Robson, 2002), see Table 4. The model defines six strategies which address three types of threats to validity, namely *reactivity, researcher bias* and *respondent bias*. The strategies may reduce the type of threat, increase it or have no effect, as defined in Table 4. Reactivity refers to that the presence of the researcher may impact on the studied setting, in particular of the behaviour of the studied people. Researcher bias refers to the preconceptions of the researcher, brought into the studied situation, which may impact on how the researcher asks questions or interprets answers. Respondent bias is based on the respondents' attitudes towards the study; it may be suspicion, leading to withhold information, or in the other end of the scale, companionship, leading to attempts to give the answer they think the researcher wants. Specific countermeasures to the validity threats brought up in Section 5 are summarised in Table 5 and are elaborated below.

*Prolonged involvement* refers to that researchers have a long-term relationship with the studied object. In this study, the researchers have a long-term research involvement with the case study companies, while the specific case study observations are conducted during a short period of time. This is assumed to provide a balance between the researcher bias threats and the respondent and reactivity threats.

*Triangulation* refers to having multiple sources for the study information. In this study, it is attained in four different ways, which further increases the validity of the study. A summary is provided below:

- *Data triangulation.* Multiple data sources were used in the study, interviews, archival and informal.

**Table 4** Strategies for dealing with threats to validity (Robson, 2002)

| Strategy | Threat to validity | | |
|---|---|---|---|
| | Reactivity | Researcher bias | Respondent bias |
| Prolonged involvement | Reduces threat | Increases threat | Reduces threat |
| Triangulation | Reduces threat | Reduces threat | Reduces threat |
| Peer debriefing | No effect | Reduces threat | No effect |
| Member checking | Reduces threat | Reduces threat | Reduces threat |
| Negative case analysis | No effect | Reduces threat | No effect |
| Audit trail | No effect | Reduces threat | No effect |

- *Investigator triangulation.* Interviews were performed by two researchers together. Important analysis steps were performed by two researchers independently.
- *Theory triangulation.* Multiple perspectives are represented in the subject set as well as in the feedback phase.
- *Methodological triangulation.* Multiple methods were used, both qualitative interviews and qualitative archival analysis and a few quantitative measures to investigate relevant metrics.

*Peer debriefing* means that researchers involve independent peers to review the research. It is not used to any larger extent in the study, except in the very late stage of the analysis. In order to be effective, a peer from a different research angle should be used, and this was not available. *Member checking* refers to involving the interviewed persons in giving feedback to the researchers. This is used continuously, by feeding back both transcripts and analyses to the respondents.

The two researchers have conducted several steps independently from each other, as mentioned previously, as investigator triangulation. This approach also provides *negative case analysis*, i.e., trying to find another explanation to the observed phenomenon, than the initially hypothesized.

An extensive approach to *audit trail* is used in the study, i.e., keeping a full record of the study, from data collection all through the analysis. The interviews are tape

**Table 5** Countermeasures to validity threats in the study

| Transition | Threats | Countermeasures |
|---|---|---|
| α | -Misconceptions<br>-Misunderstanding<br>-Lack of objectivity | -Use several sources with similar perspectives.<br>-Identify inconsistencies between interviews. |
| β* | -Misunderstandings<br>-Omissions | -Feedback to subjects before final results.<br>-Multiple researchers during interview. |
| Γ | -Audibility of source | -Use high quality microphone and recording equipment. |
| δ* | -Misunderstanding<br>-Misinterpretation of intent | -Feedback to subjects before final results.<br>-Multiple researchers in quoting and coding. |
| ε* | -Incorrect grouping due to effects of previous threats | -Previous countermeasures.<br>-Multiple researchers in grouping. |
| ζ* | -Incorrect results due to effects of previous threats | -Previous countermeasures.<br>-Multiple researchers in conclusions. |

recorded and transcribed, and the analysis is conducted with rigour. Each statement of interest in the transcribed material is encoded and summarized in tables. The results are the then derived from the tables, still keeping the full traceability back to each statement in the interviews, which the results are based on.

The large number of projects currently being coordinated using stage-gate models (Cooper, 2001) implies that, at least from the perspective of using such a model, *transferability* of the study should be high, although a case study always is coloured by its specific context. Using two different cases with two different sets of application domain, company cultures etc. also increases the transferability of the study.

In summary, reasonable countermeasures to validity threats have been implemented in the study. It is therefore contended by the authors that the validity of the study is sound.

## 6. Results

This section presents the results identified in the case studies through the analysis procedure described in the paper. The results are presented individually and aggregated conclusions are presented in Section 7. We present the findings structured into five major groups in Section 6.1–6.5:

- General
- Process
- Communication
- Timing
- Values

Further, among the observations, there are three contrasting perspectives on the observations; agile vs. non-agile teams, management vs. engineers, and EMW vs. ABB. The observations are discussed with respect to these perspectives in Section 6.6.

### 6.1. General

#### *6.1.1. EMW*

During the project, management was negative to the agile method used at the engineering level due to the fact that they felt uneasy with a different method. They felt less in control of the project and they missed the ability to squeeze extra functionality into the project without something else being removed. The developers however, felt a strong sense of control in the project and were very pleased. The opinion was that the information that the management required existed, they just did not look in the right places.

In retrospect of the project, the results are causing positive reactions within management at the company. The product quality achieved in the project is significantly higher than other parts of the organisation, although the quality measurement system does not allow follow up at this level of detail. There is now an interest to continue investigating the potential of agile approaches.

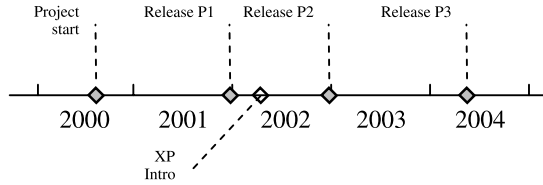


**Fig. 3** Significant events in the ABB development team

### *6.1.2. ABB*

The project was started during quarter 3 in the year 2000 using the standard development process at ABB. XP was introduced in quarter 1 of 2001 and product releases were made at the end of 2001 and 2002, as well as spring 2004. These events are shown on a timeline in Fig. 3.

The team has observed a measured increase in quality with significantly less defects in similar parts of the product as well as much fewer problems in integration. The defect data recorded in the defect management system at ABB has been extracted and is shown in the graph presented in Fig. 4. Note that the data for 2000 and 2004 is only for the part of the year during which the project was active.

Currently, the team has been dispersed as it has completed the project ahead of schedule and members have been relocated to help other projects that have not able to comply with their schedule.

## 6.2. Process

### *6.2.1. Gate Model Structure*

A general observation made is that the ABB stage gate model (Wallin et al., 2002) defines more low-level technical development details than the Ericsson PROPS model (Mulder, 1997). In the ABB model, the technical process model is more

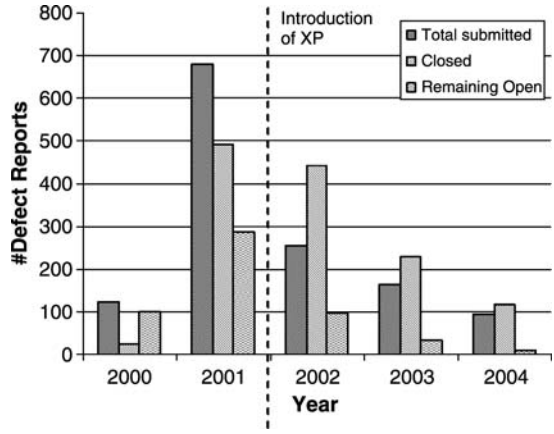


**Fig. 4** Defect report statistics from the team at ABB

tightly integrated into the management process model, while in the PROPS case, the connections can be more easily adapted. Still there is a resistance towards making the necessary adaptations. Nevertheless, the PROPS model therefore seems to be more flexible and insensitive to the underlying development process.

#### *6.2.2. Gate Model Adherence*

The gate models in the companies are not strictly adhered to in any of the companies, with or without XP. This is especially the case when the projects came under time pressure approaching gate decisions. An example of the type of deviances includes not producing the documentation required for a gate transition due to that resources are committed to solving technical issues. The advantage of XP in this situation is that it emphasises these known issues, which are often not as visible when using the companies' standard process models. The gate models are too inflexible to accommodate software development in any form. The solution must be to make the models more tailorable. This is particularly valid in the ABB case where the project management model is highly connected to the technical process model.

For instance at the time a complete requirement specification is required by the gate model, the XP team will be sitting on a functional basic version of the system. One key success factor to integrating agile teams in a gate model context is to map the artefacts required by the project management model to what is produced by the agile team, and vice versa.

#### *6.2.3. Documents*

The traditional document structures, used within the gate based project management models, need revision to fit an XP project. In the studied cases, informal documents were created as the projects progressed. These contained much of the information management needed, but this was not realized by management. The formal documents required for gate decisions were fed into the XP planning game as tasks, and treated like any other work in the project. Here again, XP emphasizes a known problem in the development process. Documents are inflexible and difficult to keep updated. Suggestions for improving this are using more flexible document formats and automatically generated documents. In the ABB case, design specifications were successfully generated automatically at any time that management asked for it.

### 6.3. Communication

#### *6.3.1. Level of Abstraction*

In the agile teams, communication takes place at a lower level of abstraction than in the non-agile teams, especially between the customer (project/product manager) and developers. This phenomenon results in that detailed functional issues are identified earlier than in non-agile teams and can be resolved earlier. Also technical problems are raised earlier leading to a risk that these hinder progress. On the other hand we

observed frustration from higher-level management about having to bother about technical issues so early, particularly in the EMW case, where the customer role was not so clearly defined.

This problem is solved in one of the study objects through the direct involvement of the technical product management in the planning game of the team to resolve these issues as quickly as possible. Automatic and manual tests were continuously used to promote technical communication and coordination with good results.

A significant advantage of the early communication is that the customer needs are made very clear early in the project.

#### *6.3.2. Team Isolation*

The XP team is considered more isolated than other teams in both cases. This results in a separation of concerns between developers and management, and breathing space to 'get on with the tasks at hand.' An enabler for this is an increased reliance on communication through unit tests.

The increased isolation, however, did on some occasions increase the amount of suspicion and speculation about what was really going on in development. The important customer communication must not be affected by this isolation, which was the case in the EMW case. This can be helped by ensuring that the customer role is clearly defined as in the ABB case, even if several people assume the role. Also, an increased emphasis on this relationship may help. Especially management seem to be in need of an attitude change here.

#### *6.3.3. Customer*

A clear customer role is essential to keep the XP team running smoothly. This must therefore be provided from the project management model. In both our cases the customer role a requirements document was used as a communication interface towards the gate model. In the EMW case, the deliverables were sometimes "put on the shelf" which had a negative impact on the progress. In the ABB case, the customer role player—a technical product manager (TPM)—was continuously testing the deliverables, giving feedback, and thus providing a driving force in the development and a continuous interest in the developed software.

The customer role ensures fast, continuous and flexible feedback. The main problem seems to be identifying a suitable person or persons to assume the role. The most important factors appear to be that the customer is active and present within the team planning game and acceptance of continuous releases. Project managers, technical product managers and marketing product managers are examples of suitable customer candidates.

#### *6.3.4. Internal Communication*

Communication within the agile development team is perceived to be much better than in traditional teams. This is believed to be due to continuous pair programming, daily stand-up meetings and bi-weekly planning games and

deliveries. One important factor affecting communication is the physical office environment. When the EMW team was spread over two floors, communication was severely impeded. Positive factors regarding internal communication include increased competency spreading, and increased project status awareness. The only negative factor reported was that of possible increased team isolation due to this, the results of which are discussed in Section 6.3.2.

## 6.4. Timing

### *6.4.1. Iteration Frequency*

The XP teams use a much higher iteration frequency compared to the gate model, even in the EMW case where an incremental development process is used. XP iterations are in the magnitude of weeks while the increments in the main projects are in the magnitude of many months. The higher frequency provides much faster feedback both for developers and management. In the studied cases, the developers adjusted to this easier than management did. The problem with this shows itself in the form of a mismatch between the information required by the gate model, and the activities dictated by the model as well as other teams. Other teams that are dependent on the work performed by the XP team operate at a different iteration frequency and, for instance, will ask for complete requirements document at a specific stage, while the XP teams delivers requirements, design *and* implementation for a part of the product. This must be solved through clearly defined interfaces with the rest of the organisation and gate model. In the ABB case, the technical product manager, playing the customer role, constituted this interface very successfully.

### *6.4.2. Good Micro Planning*

The XP teams perceive that they have much better plans for the coming weeks compared to working in common teams. The XP teams are totally focused on the work at hand and everybody is certain about what they are responsible for. The problem that became apparent was that management was more concerned with macro planning and status reporting. The solution must be to combine the successful XP micro planning with more traditional macro planning. In the ABB case, the customer also bridged between the macro plans of the overall project, and the micro plans of the XP teams. Another plausible improvement is to try to change management attitudes to the importance of micro planning.

## 6.5. Values

### *6.5.1. Flexibility*

The XP teams see no problems with continuously changing requirements in contrast to how they worked previously. This flexibility works well both when the requirements are clear and when they are more diffuse. The flexibility supports the extreme

work breakdown required for the micro planning and increased incremental delivery.

The only problem associated with this is a perceived difficulty in adhering to current quality and safety standards that the companies are certified to as these are not always flexible enough to accommodate various development methodologies. Neither of the cases had done any deeper investigation about quality or safety standard compliance, but they delivered the required documentation, which was defined as any development task in the planning game.

### *6.5.2. Under Control*

All people involved in the projects have a strong feeling of being in control, with the exception of management, which of course is a major threat to success. This is in contrast to the feeling of uncertainty that both teams experienced, hampering progress in previous projects. The teams have increased confidence in the code due to the automatic test suites, the incremental delivery in small steps with continuous feedback give more confidence in the functions that are developed and the micro planning gives control of time allocation.

Management perceived that they lost some control, as they did not recognise their usual planning models. Neither did the process model satisfactorily describe the ongoing work. Management also complained that when they asked development to squeeze in new functionality; they were in return asked what they wanted to be replaced, since the developers were so good at micro planning. Most of the management reactions were due to the change to agile methods as such, and in retrospect when they judge the results, they seem to be less critical of the methodology itself.

### *6.5.3. Quality*

Both engineers and managers consider product quality being higher than when using traditional development methods, although the quality measurement systems cannot provide quantitative measurements on that level of detail. The advantage of increased quality showed itself in the fact that there was no big-bang integration phase that previously took several months. This in turn meant that the projects adhered better to the macro plan, even finishing ahead of schedule. Part of this was perceived to be caused by engineering resources being allocated more efficiently to tasks.

### *6.5.4. Attitude*

XP was in both the observed cases introduced bottom up by motivated engineers. This appears to scare management, although several managers express a need for a change in the way they work with software products. The advantages of a bottom up approach include that the engineers feel empowered in their environment and are extremely motivated. The management attitudes must change in order for bottom up process

change and responsibility to work. One way of creating such an attitude change is management training. Further, successful pilot projects, like the ones studied, may provide evidence for the gains of the approach.

### 6.6. Contrasting Comparison

In this section we summarize the observations with respect to three contrasting perspectives on the observations in the study; agile vs. non-agile teams, management vs. engineers, and EMW vs. ABB.

Comparing the first contrasting perspectives of agile versus non-agile teams, summarized in Table 6, we can identify that the gate model is in practice not adhered to on the technical level, but this is accepted and worked with in the agile situation and ignored in the non-agile situation. Similarly, the need for documents and information are continuously worked with in the agile case, where the cost of creating documents becomes apparent as these are planned and scheduled as regular tasks, making cost/benefit analysis and direct prioritization between different types of activities possible. The focus on the product and technical issues in agile teams means that communication is performed on a lower level of detail than in regular teams. Communication with the world outside the team is more limited in the agile cases compared to the teams using their previous approaches. This is positive in that the team can get on with the work at hand undisturbed, but negative in that the team appears isolated from the rest of the organization. Communication issues regarding dependencies arise and must be solved by other means than in the non-agile case. The agile teams, however, perceive the communication within the group as much better than previously in their non-agile teams, partially due to pair-programming and team isolation. The iteration frequency is much higher in the agile teams, even compared to other incremental approaches. This is perceived as effective as it increases feedback and keeps the team on track with both the schedule and the stakeholders' requirements on the product. The agile teams have no problems handling changes in requirements whereas non-agile teams have to go back to the early stages of the

**Table 6** Summary of observations on the contrasts between agile and non-agile teams

| Finding | Agile vs. non-agile |
|---|---|
| Gate model adherence | Model not adhered to in any case. Agile team emphasises need for more flexibility. |
| Documents | More informal documents used in agile teams. Document writing planned as normal agile tasks. |
| Level of abstraction | Agile teams communicate on a lower level of abstraction. |
| Team isolation | Agile teams work, for good and for bad, more isolated. |
| Internal communication | The agile teams perceive the internal communication better than in non-agile teams. |
| Iteration frequency | Agile teams have higher iteration frequency, also compared to other incremental approaches. |
| Flexibility | Being flexible to e.g., changing requirements is an integrated part of the agile teams. |
| Quality | The resulting product quality is considered better for agile teams. |

development process and redo a lot of work. In retrospect, the products created by agile teams have been considered to be of significantly better quality than that of non-agile teams, although this was not quantitatively assessed.

Looking at the second contrasting perspective, management versus engineers, (Table 7) we can identify the following interesting observations. Management dislikes handling detailed level issues. It is believed that the engineers should resolve these, whereas engineers need support to resolve detailed issues early before they turn into bigger issues. Concerning the iteration frequency, it was observed that the engineers adjusted to the higher frequency easier than management. This could be due to that the way that management plans at a high level today is not optimised for a highly iterative way of working making the integration difficult for management. Management expressed concern with the good micro planning at the engineering level, as the engineers were unable to slip things into the schedule without taking something else out. This schedule control is perceived as one of the reasons of the success of the agile teams and, in contrast to the perceptions of management in these cases, should increase control of management as well, if properly understood and managed. Well functioning micro-planning seem to lead to better adherence to the macro-plans, which management is responsible for. The engineers experience this as an increased control of their situation while management feel that they are loosing control, as they are not adapting to the new situation. This leads to positive attitudes from engineering compared to negative attitudes from management, at least during the course of the projects. As accounted for previously, management is in retrospect interested in the methods due to quality increases and adherence to plans and schedules.

The third contrasting perspective, EMW versus ABB, is summarised in Table 8. In general ABB had a more positive approach to the introduction of the methodologies. Support was provided from management in the form of resources for external consulting and time for education. This meant that the management was more involved in the whole initiative. At ABB, however, the gate model is more fixed and closely related to the actual technical work making it harder to accommodate a different methodology of this kind. Neither company managed to accommodate the agile team with complete success; indeed the successes described were achieved without changing the gate models at all from the management's perspective. For the customer role, ABB appointed this to the TPM, making this role very clear and available to the engineers, while this role was very unclear at EMW.

**Table 7** Summary of observations on the contrasts between management and engineers

| Finding | Management vs. engineers |
|---|---|
| Level of abstraction | Management dislike handling the detailed level issues. |
| Iteration frequency | Engineers adjusted better to the higher frequency than management. |
| Good micro planning | Management finds sometimes, paradoxically, good micro planning as a problem, since they are concerned with macro planning. |
| "Under control" | Engineers find agile project work "under control," while managers feel they are loosing control. |
| Attitude | Engineers are positive towards agile methods while managers are sceptical. |

**Table 8** Summary of observations on the contrasts between EMW and ABB

| Finding | EMW vs. ABB |
|---|---|
| General | ABB was more positive towards the agile team than EMW. ABB has a more flexible company culture than EMW has. |
| Gate model structure | The ABB model structure is more fixed than the EMW structure, but neither company managed to adapt the model to accommodate the agile teams. |
| Customer | ABB appointed a customer role to the TPM, while the customer was less clear in the EMW case. |

## 7. Summary and Conclusions

The results of the study show that it is indeed possible to successfully integrate project management models of the stage-gate type with XP projects. This has been achieved through complete isolation of the engineering teams and attempting to make these look as similar to a regular team as possible. Based on the results we identify certain key issues that are believed would make a more effective project management, involving agile methods. These include:

- involving developers early in the product development, to quickly identify and eliminate technical issues and clearly outline plausible solutions;
- adapting the project planning to accommodate for agile micro planning in combination with macro project planning;
- identifying critical feedback loops and make these as short and fast as possible;
- striving to make an early version of the actual product as quickly as possible to exemplify solutions and use for communication;
- using technical tools for technical coordination, such as for example automated testing of technical dependencies;
- making the customer-developer roles and interactions as clear end effective as possible;
- working chiefly with management attitudes to accommodate uncertainties due to them not feeling at ease with the new processes.

An observed benefit due to the increased quality of the developed code was the lack of "big-bang" integration problems, making the overall project much more predictable. The increased control felt both by the customers and developers is also considered to be an advantage in the cases studied. This is in contrast to the contradictive lack of control felt by the management.

In summary there are several interesting benefits of integrating agile teams into traditional stage-gate project management models and we have identified some key issues regarding these. There are also some key issues that must be addressed in order to be successful, some of which have been identified in this paper, in particular a clear customer interface, strategies for artefacts in the gate model, balance between micro and macro level planning, and management understanding of the agile principles.

**Acknowledgments** The authors would like to thank all the people that were available for interviews at ABB and EMW. This work is partially funded by the Swedish Research Council under grant 622-2004-552 for a senior researcher position in Software Engineering.

## Appendix A: Interview Instrument

General (~10 min)

Explain about the study, what we are looking for, how they will benefit from the results and that they will be anonymous.

Subjects personal history in the company (keep short).

Describe how this project fits into the overall organisation structure and product portfolio.

Gate Model (~10 min)

About the corporate gate model (ask for any documents that can be provided).
What is required as exit criteria from G3?
Which documents are used at G3?
How is the G3 decision taken?

XP (~15 min)

XP introduction, how was this performed?
XP generally, how did it work, describe the most significant aspects.
What worked well?
What did not work as well?
XP experiences compared to your previous way of working.
Did you adapt XP to fit the gate model?

Adjustments to gate model 20 min

How was the G3 decision affected by you using XP?
Describe any adaptations of the gate model in general.
Any adaptations specifically due to XP?
How was it apparent to people outside your team that you were using XP?
How did you work with the documents required by the gate model?
Did you use the software product as a form of communication in itself?
How did the planning game affect your project?
How did you perform time estimation?
Could you start development at a different stage than in your previous projects?
How was the project affected by working on the most important feature first?
How did the automatic unit testing affect your project status reporting?
How was your delivery certainty versus the gate model affected?
What are the advantages and disadvantages of the gate model?
What are the advantages and disadvantages of the XP methodology?

Ending (~5 min)

Other observations? Anything we have not covered?

Any relevant observations with regards to your specific perspective in the project?

Finish with a brief summary, thank the subject, and confirm that it is ok to come back to the subject with questions in retrospect. Inform about the feedback procedure.

## References


Agile Manifesto (2001) http://www.agilemanifesto.org last accessed 051014

Beck K, Andres C (2004) Extreme programming explained (2nd edition). Addison Wesley

Boehm B, Turner R (2004) Balancing agility and discipline. Addison Wesley

Cooper RG (2001) Winning at new products (3rd edition). Perseus Publishing, Cambridge, MA

Fuqua AM, Hammer JM (2003) Embracing change: an XP experience report. Extreme programming and agile processes in software engineering, 4th International Conference, XP 2003. Genova, Italy

Gilb T (1998) Principles of software engineering management. Addison-Wesley

Grenning J (2001) Launching extreme programming at a process intensive company. IEEE Software 18(6):27–33

Karlström D (2002) Introducing extreme programming—an experience report. Proceedings of the Third International Conference on eXtreme Programming and Agile Processes in Software Engineering (XP'02). Sardinia, Italy

Karlström D, Runeson P (2005) Combining agile methods with stage-gate project management. IEEE Software 22(3):43–49

Kitchenham B, Pfleeger SL, Hoaglin DC, El Emam K, Rosenberg J (2002) Preliminary guidelines for empirical research in software engineering. IEEE Transactions on Software Engineering 28(8):721–734

Mulder L (1997) The importance of a common project management method in the corporate environment. Blackwell Publishing: R&D Management 27(3):189–196

Murru O, Deias R, Mugheddue G (2003) Assessing XP at a European internet company. IEEE Software 20(3):37–42

Patton MQ (2001) Qualitative research & evaluation methods, (3rd edition). Sage Publications

Paulk MC (1999, March) Practices of high maturity organizations. The 11th Software Engineering Process Group (SEPG) Conference. Atlanta, Georgia, pp 8–11

Paulk MC (2001) Extreme programming from a CMM perspective. IEEE Software 19(6): 19–26

Paulk MC, Curtis B, Chrissis MB, Weber CV (1993) Capability maturity model for software, Version 1.1, Software Engineering Institute, CMU/SEI-93-TR-24

Paulk MC, Weber CV, Curtis B (1995) The capability maturity model: guidelines for improving the software process. Addison Wesley

Poole C, Huisman JW (2001) Using extreme programming in a maintenance environment. IEEE Software 18(6):42–50

Rainer A, Hall T, Nathan B (2003) Persuading developers to 'buy into' software process improvement: local opinion and empirical evidence. Proceedings of the International Symposium on Empirical Software Engineering. pp 326–335

Rasmussen J (2003) Introducing XP into greenfield projects: lessons learned. IEEE Software 20(3):21–28

Robson C (2002) Real world research. Oxford: Blackwell Publishers

Royce W (1998) Software project management—a unified approach. Addison-Wesley Longman

Schuh P (2001) Recovery, redemption, and extreme programming. IEEE Software 18(6): 34–40

Seaman CB (1999) Qualitative methods in empirical studies of software engineering. IEEE Transactions on Software Engineering 25(4):557–572

Sharp H, Robinson H (2004) An ethnographic study of XP practice. Empirical Software Engineering 9(4):353–375

Vanhanen J, Kähkönen T (2003) Practical experiences of agility in the telecom industry. Extreme programming and agile processes in software engineering, 4th International Conference, XP 2003. Genova, Italy

Wallin C, Ekdahl F, Larsson S (2002) Integrating business and software development models. IEEE Software 19(6):28–33

Yin RK (1994) Case study research design and methods. Sage Publications, Beverly Hills, CA

**Dr. Daniel Karlström** is a research associate at the Department of Communication Systems, Lund University, Sweden. He received a Ph.D. in software engineering from Lund University in 2004, and an M. Sc. in Electrical Engineering in 2001. He has previous experience of software engineering in large organisations. Dr. Karlström's main research interests are developer involvement in software process improvement, agile methods, maturity frameworks for software development, software quality and project management.

**Dr. Per Runeson** is a professor in software engineering at the Department of Communication Systems, Lund University, Sweden, and is the leader of the Software Engineering Research Group since 2001. He received a Ph.D. from Lund University in 1998, and an M. Sc. in Computer Science and Engineering from the same university in 1991. He has five years of industrial experience as a consulting expert in software engineering. Prof. Runeson's research interests concern methods and processes for software development, with special focus on verification and validation. He has published more than 70 papers in international journals and conferences and is the co-author of a book on experimentation in software engineering. He is member of the editorial board of Empirical Software Engineering.