Taming the Big Animal – Agile Intervention in a Large Organization

Timo Punkka
Helsinki University of Technology, Software Business and Engineering Institute
timo.punkka@fi.schneider-electric.com

***Abstract*— New Product Development (NPD) today is mainly managed with plan-driven, relay race type, processes. Today's dynamic markets and new technology used in NPD result in a possibility of benefiting from more adaptive, rugby style, methods. In the software community methods applying this approach are commonly called agile methods. In this study I present experiences from a bottom-up initiative to apply agile methods for embedded system development in a large organization using the Stage-Gate® project management framework. The purpose of the study was to find out whether agile methods have a chance and how much this type of organization could benefit from agile methods. Study supported the earlier research showing that implementing incrementally some agile practices the project organization was able to improve the team spirit, motivation level, and additionally to increase productivity. The affect of agile practices to the Stage-Gate® framework was minimal. Through wider buy-in and commitment, the agile development increments could be synchronized seamlessly to the Stage-Gate® milestones. However as described earlier in literature, top-down commitment and changes in organization culture are needed in order to succeed in process change. The philosophy of agile development is quite contrary to so called traditional NPD approaches. These changes in organization culture are also needed in order to fully harness the power of agile development.**

***Index Terms*— Agile process, bottom-up initiative, embedded system development**

I. Introduction

Is the new product development (NPD) experiencing a chronic crisis? Gibbs diagnosed software's chronic crisis in 1994 (Gibbs, 1994). He had noticed that while the complexity of software to be developed keeps growing the development methods used do not meet the challenge. Today electronics and microcontrollers are getting so cheap that even the simplest gizmos will have complex systems built in them. (Graaf, Lormans and Toetenel, 2003) found out that hardware and mechanical perspectives were driving the embedded system development in 2003, but the role of embedded software was growing. A recent 2005 survey in Embedded Systems Programming magazine[1] reveals that software effort is dominating the co-design of embedded projects in all three categories; time, manpower and spending. This implies that software methods should be considered more thoroughly in this traditionally electronics driven environment. Product development and management association (PDMA) claims that new products currently have a success rate of only 59 percent at launch, up only one percent since 1990. Cancelled or failed (in the market) products consumed an estimated 46% of product development resources (Cooper, 2001). At the same time the time-to-market demands are getting tougher because of more intense rhythm of every-day life. The current practices in new product development are not going to sustain these new project requirements.

Relay race type processes rule currently NPD. The process is often very bureaucratic. Quality assurance is managed by formal reviews. Organizations try to control the development work by planning the project up-front and then do anything to stick to that plan. Long analysis and specification periods precede the design effort to ensure that the actual development would get the product right at the first time. This is pure fiction today; the customer does not know what he wants at the pre-study phase of a project. Even if he has an idea at the beginning, the markets and technology are at high risk to change during the development phase. The required documents are not updated during the development and they fall out of synchronization with the actual development. This means that the hard work at the beginning is wasted anyway, not to mention the cost of missed time to market. In the worst case markets are reached late with a wrong product.

To tackle these difficulties, a different approach is establishing a foothold in contrast to conventional heavy process models. This rugby style (Nonaka and Takeuchi, 1984) approach embraces the change and instead of being plan-driven it is adaptive. In the software community methods applying this approach are commonly called agile methods. The term *agile* was taken into use in 2001 when a group of methodologists and developers of then called lightweight methods formed the *Agile Alliance*[2]. The core values and practices of agile development are written down as the *Agile Manifesto*[3]. Table I lists the common agile development principles (Abrahamson et.al., 2002). Agile methods can be

[1] Embedded Systems Programming, July 2005, Vol.18, number 7

[2] Agile Alliance (http://www.agilealliance.org)

[3] Agile Manifesto (http://www.agilemanifesto.org)

described as emergent process models focusing on proceeding with actual development with short increments after just enough pre-planning. Planning is called for when experimenting is expensive. Compilation power enables rapid incremental development in software development. Lowered cost of experimenting in electronics and mechanics development enables us to fail once or twice before getting it right also in embedded system development. Stefan Thomke calls this an era of "enlightened experimentation" (Thomke, 2001). This new possibility to experiment opens the window for agile methods. Principles from agile methodologies in embedded system development are worth studying.

Table I

Common Agile Development Principles (Abrahamson et.al., 2002)

Development Principle	**Description**
Incremental	Small releases, with rapid cycles
Cooperative	Customer and developers working constantly together with close communication (collaboration)
Straightforward	The method itself is easy to learn and to modify, well documented
Adaptive	Able to make last moment changes

Agile intervention, or agile enablement, is a term used in literature to describe the process of changing an organization to be more agile in its development practices.

In this paper I present results from an action study of an bottom-up front-line employee initiative to apply agile practices in NPD co-design activities in a large organization. The objective is to find out if agile practices can help NPD and how agile practices can be integrated into this environment. I describe what we did in the case company, why we did it, and how it influenced the NPD project. Another objective is to share experiences with people struggling with similar problems and to give everyone something to think about.

This paper is composed as follows. In the next chapter I review the existing related studies. In the third chapter I present justification for the research method. After that a brief introduction to the study's setting is given, followed by a chronological description of events. The final chapter wraps all together and presents the key findings of the study with discussion.

II. Related Studies

Agile methods are originally targeted for small, non-critical, software projects with vague, or rapidly changing, requirements (Beck, 2000). The current knowledge on agile methods created in industry experiments proof the applicability to wider range of situations than originally targeted. Studies on using agile methods to develop embedded software have been published in recent years (Grenning, 2002, Grenning, 2004, Manhart and Schneider, 2004 and Greene, 2004). Many of these analyses focus on extreme programming practices (Beck, 2001). Highsmith has defined a more abstract agile project management framework suitable for general new product development (Highsmith, 2004). There has also been founded a Agile Project Leadership Network[4] (APLN). APLN has defined a modification of the Agile Manifesto more suitable for general project management. This is called a Declaration of Interdependency[5] (DOI). There is an on-going project in ITEA (Information Technology for European Advancement) program to define an agile framework for embedded systems development. This project is called AGILE-ITEA[6]. Large experiment showed that agile methods can be anchored to the large organizations main Stage-Gate® process (Karlström and Runeson, 2005).

However, only very limited empirical evidence can be found on applying agile methodologies or even methods or practices on co-design level required in embedded system development.

III. Research Method

Quantitative research is without a doubt efficient in many fields of research. Certain situations however are not appropriate for such an approach. Agile intervention is unlikely to offer possibilities for rigor research planning and gathering of quantitative data. Qualitative research is more powerful for such an event. Robert K. Yin defines the case study research method as an empirical inquiry that investigates a contemporary phenomenon within its real-life context; when the boundaries between the phenomenon and context are not clearly evident, and in which multiple sources of evidence are used (Yin, 1984). This research is conducted as an action study, a sub-class of case studies. Action research aims at solving specific problems within a program, organization, or community. As a result, the distinction between research and action becomes quite blurred and the research methods tend to be less systematic, more informal, and quite specific to the problem, people, and organization for which the research is undertaken. There is no intention, typically, to generalize beyond those specific settings. (Patton, 2002)

For the primary data I conducted all together 13 open-ended interviews, approximately 45 minutes each with active participants of the agile intervention. Interviewees consisted of developers of different engineering disciplines, line managers and project managers. The interviews were recorded and summaries were written. Each interviewee reviewed the summary. Modifications were made if necessary until the summary was agreed. Additional qualitative data was collected as a 22 page (A4) journal by the author. The journal included notes from daily observations and team meetings.

[4] Agile Project Leadership Network (http://www.apln.com)
[5] Declaration of Interdependency (http://pmdoi.org)
[6] AGILE-ITEA project (http://www.agile-itea.org)

The analysis was done using the ATLAS.ti qualitative analysis software package. Quotations from interview data were created and collected under categories using the software. These categories were used as a basis for the key findings reported in this paper. I increased the study's validity by having a final workshop where group of 9 persons reviewed the identified categories and dependencies between them. All categories and quotes under them are fully traceable to individual interviewee statements.

IV. The Setting

A. Organization

Control Offer Development (COD) is the R&D function of installation electronics development in Product&Technology Division of Schneider Electric. COD develops building automation and mains network control devices. Turnover of this business area is about 150M€. The R&D activities are mainly conducted at 2 sites (Denmark and Finland) and involving 53 pairs of hands. Market area is global, but also contains multiple brands that are very strong in their own area. In year 2004 roughly 20 new products saw daylight. This means a multi-project setting, but usually single major project dominates the effort. Products typically include one or more low-end microcontrollers.

B. Process

Modification of Stage-Gate® (Cooper, 2001) was the house process for new product development, but the process was new and none executing it had sound understanding of it. The front-line had no formal training to process. Stages and gates are presented in Fig. 1.

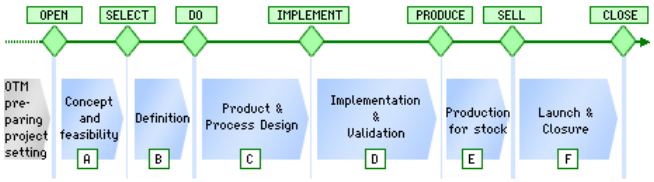

Fig. 1. Schneider-Electric Stage-Gate® process for NPD.

The Schneider-Electric process answers more to business, development resource and budgetary questions than provides guidance to daily rhythm of frontline development. (Wallin, Ekdahl and Larsson, 2002) recognized the limitations of these two different perspectives and documented the integration of business and software development models in ABB. (Rautiainen, Lassenius and Sulonen, 2002) defined an agile process framework to close the gap between business decisions and everyday development.

C. Project

Stardust is a sub-project of *Supernova* program. Supernova is developing new design series for light switches and socket outlets. Stardust is developing electrical switches to fit this new design series. Stardust project is the biggest development effort by COD so far in terms of money, effort and by far complexity. The project is developing a range of modules for mains control to gain a large number of end products from limited number of modules. Modular structure also offers a system upgrade path for the customer. A final end product is composed of load controlling unit, a user interface unit and optionally a wireless connection unit with rf technology. At the time of the agile intervention the functional requirements were nearly missing and even the number of hardware modules to be developed was changing monthly, and even daily. We can say that this was a turbulent environment enough for agile methods.

V. The Story

This chapter tries to capture the essentials of this experience in chronological order. At the time of writing this paper 7 increments were conducted. The time span and team evolution are illustrated in Table II.

Table II

Team Evolution Over the Experience

Sprint #	Sprint Duration	Team Size	Description
1	7	5	1 SW Developer 2 Line Managers 1 RF Developer 1 Marketing Manager
2	10	5	1 SW Developer 2 Line Managers 1 RF Developer 1 Power Electronics Designer
3	20	10	1 SW Developer 2 Line Managers 1 RF Developer 2 Electronics Designers 2 Mechanics Designers 2 PCB Layout Designers
4	10	7	1 SW Developer 1 RF Developer 3 Electronics Designers 2 Mechanics Designers
5	20 (Cancelled)	7	1 SW Developer 1 RF Developer 3 Electronics Designers 2 Mechanics Designers
6	22	Team1: 6	2 Electronics Designers 2 Mechanics Designers 2 PCB Layout Designers
		Team2: 3	3 SW Developer
7	22	Team1: 5	5 Electronics Designers
		Team2: 5	5 SW Developer

A. The Story So Far

Because it was impossible to have an all-at-once approach within the bottom-up initiative, I wanted to evolve the method from few very simple practices. I chose Scrum (Schwaber,

2002 and Schwaber, 2004) as the base. One hour presentation of agile principles and Scrum methodology in more detail was given to all developers participating in agile development.

The first two short increments (Sprints in Scrum method terminology) were used to figure out the overall work do be done, and to draw the skeletons for the needed process documentation. There were two line managers in the team. They could not commit to development work enough to be able to actually work on an agile team. The good result from that period is that we have two line managers with hands-on experience on how agile working feels like. As one of the line managers said;

"…reality really hit me; you can accomplish very little in a week…"

The third increment was scheduled to be 17 working days, and the team size grew to 10 developers. In the sprint planning stakeholders were present from line management, project management and marketing. After negotiation a prototype was decided to be developed and assembled during the increment. Progress started from day one in a cross-disciplined team (see Table II). Small stand-up technical meetings happened regularly after the official part of daily Scrum meeting. There was a whiteboard available outside the office where the daily scrum was held and small groups drafted ideas there and decided how to continue each day. This is important when people are cross-disciplined. When developers go deep into details of their own area, others would just waste their time listening.

In the review were line managers, project managers and people from marketing organization represented. A working prototype was presented. Developers described their part of the design on quite low technical detail. Developers only explained the customer value that their work added, and did not go into much technical detail. The work accomplished was highly appreciated:

"I'm surprised on performance of the team. I expected you to have a prototype but this is great. This looks like product from production."

"This is impressive"

"I'm super impressed. This gives a feeling what the product would be. It has to be a visual model of the actual product for a person like me. A bread board proto with wires hanging out would not have value for me."

The objective for 2 week increment number 4 was defined as polishing the paper deliverables for the Stage-Gate® gate. We continued to have daily status meetings and we delivered the paper documentation at the end of the increment.

Because process gates were synchronized very loosely to actual development progress the project team initially had only 5 weeks scheduled for development phase. Based on this information the increment 5 was started by specifying and designing the products on paper. One of the designs was decided to be developed to a prototype if time permitted, but the focus was on delivering paper. The team had worked hard with documents for the last two weeks, and continuing this without any additional experimenting was frustrating. The daily meetings got insignificant, and there was very little collaboration. After all, formal documentation is the opposite to collaboration and everyone was working with their own piece of document. During increment 5 the next gate got postponed and we cancelled the increment. During these 7 weeks there was very little value added to project, no new knowledge created, and thus the risk in project remained untouched.

As the process gate got postponed we were able to plan for a full 30 day increment. In planning for increment 6 we decided to shift the focus back to the prototypes. We also tried multiple teams for the first time; we had separate teams for firmware and hardware development. Documentation was not mentioned specifically as a task in our backlogs. Paper deliverables still emerged at the steady phase. In increment 6 review we had the biggest participation so far. There were roughly 20 people in the electronics lab. Stakeholders from management, project management, marketing, and ramp-up production site were represented. As a conclusion the review was considered to be a "good show".

"This is where you really see where the project is going. The picture you get from reports and e-mails is very different"

Increment 7 was conducted in a similar way as the previous one. We had two teams, one for hardware and one for SW/firmware. Number of new hardware platforms arrived based on design work done in the previous increment. This made HW designers occupied with assembling and debugging the designs. There was little new creative design activity. Because of this the HW team was having a status meeting only 2 times a week, but following the three Scrum questions. Towards the end of the increment the practices in the hardware team started to fade out. Hardware team still committed to putting out the review by scheduling tests in order to have working prototypes for software. In the review there were 4 completely new prototypes presented and 2 more modules with revisited hardware. Because of simple design and efficient configuration management the SW team was able to reuse much of the SW architecture and source code to implement the basic functionality for all new modules as well. While the participation on this review was not such a success as the previous one, the progress was considered "impressive".

VI. Findings And Discussion

This section presents our findings from the research. The most meaningful result is that agile methods can be applied also in embedded system development, and they benefit the co-design problems as well. Adaptation to methods is needed, but agile methods themselves seem to guide the team in doing this.

A. How Agile Were We?

Table III

Description of Used Agile Practices

Practice	Description of the practice	Success and value of practice
	AGILE PROJECT MANAGEMENT PRACTICES AND TECHNIQUES	
Sprint Planning/ Planning Game	The planning game has two participants: development (those involved in building the system) and business (the people responsible for determining what the system will do) (Schuh, 2005). Using the data from earlier increments the two parties negotiate the scope of the forthcoming increment.	Planning for the next 1-5 weeks was done in planning meetings and the Sprint backlog was agreed with all stakeholders. In an embedded project it is often difficult to find the customer that would be capable of prioritizing the functions. Large part of development effort may be necessary before anything visible to end user level is achieved (monolithic system). Even while the developers were the ones to prioritize most of the technology task, the required work was made visible to other stakeholders.
Sprint Review	A meeting at which the team demonstrates to stakeholders what it was able to accomplish during the current increment. Only completed product functionality can be demonstrated.	We learned to keep the technical description of features abstract so that all stakeholders could follow. However finding interesting features at this level was sometimes difficult. Embedded systems are often developed for single relatively simple primary purpose. We however succeeded in this, and reviews were appreciated and participation was stable through the experiment.
Metaphor	The project team writes a simple statement that says what the system should do after the next increment This statement is used to keep the team focused on the original target.	When the metaphor is set correctly it is highly valuable in making the decisions on tasking inside the increment. It was however difficult to find the working metaphor.
Daily Scrum Meeting	Daily meeting is held to synchronize the information. Each team member answers to three questions in turn; 1) What I did since the last meeting, 2) What I am going to do before the next meeting, 3) What impediments do I have keeping me doing my work?	Daily status meeting was the single most effective practice working as a catalyst for developer collaboration. Daily meetings were held during the whole experiment. In increment 7 the HW team reduced number of status meetings to only 2 per week.
Customer Collaboration	Agile methods replace much of documentation with intense collaboration with customer. Extreme Programming requires a full-time on-site customer.	In an embedded system project the role of customer might be filled with different stakeholders during the different phases of development. In this study engineer collaboration between different disciplines worked well due the team work. Marketing collaboration was also at time very intensive. There is constant pressure to update specifications based on new knowledge and communicating this formally with paper documents is difficult and inefficient.
Product and Sprint backlogs/ Burndown charts	The trend of work remaining across time in a Sprint (increment), a release, or a product. The source of raw data is the Sprint Backlog and the Product Backlog, with work remaining tracked on the vertical axis and the time periods (days of a Sprint or Sprints) tracked on the horizontal axis (Schwaber, 2004).	Sprint backlog(s) and burndown charts were updated during the development. They were mainly used as task lists by developers. Technically competent line managers were able to follow the progress with backlog, but their use in administrative tasks was limited. There should be a method for this high level reporting which should be achieved easily from backlogs. It was not seen satisfying just to fit backlog into gannt chart as suggested by (Schwaber, 2004).
Retrospective Meeting	A meeting at which the team discusses the just concluded Sprint and determines what could be changed that might make the next Sprint more enjoyable or productive (Schwaber, 2002).	After each increment a retrospective meeting was held as a workshop for improving the methods. This is one of the key issues in agile development making the continuous improvement possible. It is even more important to have the meeting after a failed increment. This is the place where the difference between a coincidence and chronic pain is made.
Information radiator	Information important to the daily operation of the team should be posted on a wall near to the team, in plain view, and kept up to date (Schuh, 2005).	Information radiator was experimented, but only occasionally. The information mainly consisted of backlogs and burndown charts. Without full-time team leader this information was usually quickly out-dated and useless for team members.
	AGILE SOFTWARE DEVELOPMENT PRACTICES AND TECHNIQUES	
Collective ownership	Collective ownership means shared source code, and also shared responsibility for the source code. Anyone on the team can improve any part of the system at any time (Beck, 2000).	Source code was controlled with CVS and each software developer had own working copy. We did not assign responsibilities to certain parts of code. Knowing the whole code base seems to ease the development. The consequence of a change to any part of the code is known to person making the change.
Pair Programming	Write all production programs with two people sitting at one machine. Pair programming is a dialog between two people simultaneously programming (and analyzing and designing and testing) and trying to program better (Beck, 2000).	Pair programming was not done systematically. However developers were quickly introduced to new parts of code by coding the first feature with someone familiar with that part. In addition pair debugging of complex defects, and thus verbalizing the problem, proved to be highly efficient.
Unit Tests	Not really specific to agile development. Unit tests however enable practices like test-driven-development and automated regressions tests, which are strongly promoted by agile methods.	There is no out of the book recipe for unit testing low-end firmware, even while some papers exists (Van Schooenderwoert and Morsicato, 2004). We gave unit tests and test-driven-development some thought. Some unit tests were written and a simple build and test server was set up to do continuous integration.
Continuous Integration	Small bits of functionality are integrated as they are completed into an up-to-date and clean codebase (Scuch, 2005).	Getting a hard real-time requirement right may be time consuming and the solution may affect a large part of the software. This causes serious delays in CVS commit rhythm, and these delays can cause very expensive to resolve problems. It is important to identify and schedule this work into beginning of a new increment when everyone else still has time to adapt to new situation.
User Stories	User story is an agile way of defining requirements. They are similar to use cases, but differ in amount of detail. A proper user story is just three phrases describing the customer-visible functionality.	In an embedded system there is very little end-user visible functionality. If we extend the definition of customer to for example engineer from other discipline or technically competent manager, and/or take into use concept of developer story, we can use stories to prioritize the work and be able to better execute the *planning game*.

This experiment tested more the agile thinking in abstract level than any specific agile method. The experiment was started with the Scrum framework, but we did not have chance to have continuous 30 calendar day increments as defined by Scrum, or to have a fully committed Scrummaster. Nor was it possible to execute the sheepdog's role as described by Schwaber (Schwaber, 2004) to protect the team, since the experience was bottom-up initiated and the management commitment was lacking. Being an agile team does not mean that the team has to have all the practices of extreme programming fully applied. The team or organization can be agile in its way of thinking. This is where the agile practice of "art of possible" has a major role. Literature presents a number of practices for agile development. For example Peter Schuh lists all together 31 agile practices and techniques in his book (Schuh, 2005). Table III describes the agile practices that we tried during this experiment and how they succeeded.

B. Key Findings

Table IV lists the seven key findings of this study based on observation and interview data. These findings were identified as the root issues causing the phenomenon experienced.

Table IV

Key Findings

Finding	Description	Recommendations
Missing or Vague Requirements/Specification	Missing, changing and vague requirements especially in the beginning of a project is the number one cause of developer confusion, and thus project delay.	When an agile framework is fully utilized it should have a vision creation activity in the beginning. This vision guides the creation of initial release plan, and is used to justify the prioritization of tasks and features to be implemented. The incremental planning makes the affect of change visible.
Daily Scrum as Catalyst	Embedded system development requires co-design effort. Cross-disciplined teams need to share lot of information, and this need is frequent. Daily status meetings foster this perfectly.	It is important to have discipline to follow the 3 Scrum questions. The status meeting can only be successful if it is used for information synchronization and not problem solving. A fulltime agile team coach with broader knowledge on agile methods should be available especially when the team is just taking the agile practices into use.
Lightweight Method	The Scrum framework is putting only a light control over the chaotic development. Practitioners did not feel a need for formal education in agile development, but were able to gain from these practices from day one.	The team coach needs both, the knowledge of agile methods and skills to offer them to team without putting emphasis on the theory or philosophy behind the methods.
Incremental Planning and Development	Incrementally planning the activities and developing the products put focus to important issues first instead of wondering around.	Project team should put more focus to release planning (i.e. for next 3 months) and balancing up-front design. This was not possible in this research because of bottom-up initiative.
Lacking Stakeholder Commitment	Large stakeholder commitment was lacking during the whole research. This is mostly because lack of information. The agile intervention was bottom-up initiated and there was no full-time person to share information about the new method.	Visible benefit from this experience has awoken interest outside the team. In the next agile pilot the overall buy-in to use agile methods should be easier to achieve.
Missing Administrative Interface	In a large organization different views to project progress are needed. During this experiment building these bridges between different perspectives was not achieved. In the case organization the Gannt chart was the selected format to report, but during the experiment this was not used at all.	If a Gannt chart is needed, then agile team progress reporting can be easily adapted to this. This requires a competent project manager to have experience in both worlds to be confident in defending the need for different views.
Unsynchronized Development and Stage-Gate Model	The Stardust project was only a part of a bigger program. At times timing of milestones, gates, seemed unsynchronized with the actual development progress. This often led to fire-fighting type project steering decisions.	In the overall project plan the incremental development needs to be anchored to business the Stage-gate model. This anchoring is easy to achieve and the Stage-gate model, and any other process, can be used to prioritize the development tasks. Increment rhythm also needs to be appreciated by the organization. During this study other tasks violated the agreed Sprint review dates, and created unnecessary delays in progress.

Missing or Vague Requirements/Specification. Stephen Balacco, embedded analyst at VDC (www.vdc-corp.com) was quoted in December 2002 issue of SD Times[7] as "*Embedded [developers are] frustrated by inadequate or changing specifications during product development*". In the article it is mentioned that two-thirds of the 400 respondents cited changes in specification as the number one cause of the delays. Every developer in this research, when interviewed, stated missing or vague requirements and specification as main project pain and cause of confusion. This confusion is directly related to project delays.

Daily Scrum as Catalyst. Daily Scrum meetings were working extremely well as a catalyst for collaboration. Co-design with cross-disciplined teams is typical to any modern embedded system development process model. Yet, while the effectiveness of concurrent development is acknowledged the close collaboration within the engineers resulting from very light agile practices seems to accelerate the effect even more.

From the very beginning trough the whole experience the developers were very easy going with the idea of daily meetings. In every retrospective meeting daily status meetings were mentioned as the single most valuable practice. The designers coming from harder disciplines like electronics, mechanical or printed circuit board designers, proposed maybe 2-3 meetings per week to be enough. The most experienced developers were the only ones resisting the daily meetings. It seems that they don't feel like gaining anything in

[7] http://www.sdtimes.com

daily meetings as they know how to develop these products. It would however be very important to have them involved in order to synchronize the project information, generalize created knowledge and maybe even challenge old habits with new information.

Multi-project environment made occasionally the daily planning difficult as the tasks that developers committed to could not be completed because of other emergent responsibilities.

Lightweight Method. Using Scrum as a framework enabled us to start practicing agile methods from day one without a formal education. This "Just-in-Time" (Hutchings et al., 1993) type of training was seen sufficient by all attendees. Furthermore, a more detailed training on agile development, the theory or the philosophy behind it, was not seen interesting by most of the people. When new practice is introduced this way, the need for team leader or coach with broader knowledge on methodology was seen evident. One manager level interviewee described:

> "It is OK to be a follower, if you have someone to follow."

Deeper knowledge is needed to be able to keep the team on track. During this study the team leaders were wearing multiple hats, and this was seen as a problem. The bigger picture is easily left without notice when the person leading the team participates also in development work.

Incremental Planning and Development Incremental approach to development really helped us in getting the focus to the task at hand. While the scope of the project was vague, but the development effort was needed, the incremental planning helped us in prioritizing the development in an easy, almost effortless, way. We were able to justify the development effort to be spent on things that we thought were the most important or the most unlikely to change. Similar situations have happened before, and normally the start of development effort has been delayed.

It is common that during the development new ideas emerge, or new technologies are discovered. Ideally bureaucratic relay race type processes tackle this by making change so difficult through change management that change would be avoided. In practice changes sneak in and their affect on project schedule and budget is left without analysis. In incremental agile planning the effect is immediately visible and any needed re-scheduling can be initiated.

Stakeholder Commitment. The experiment was started after a bottom-up initiative. During the whole experience the top management knowledge of agile methods was low or even non-existing. There was no over all project synchronization to increments or respect towards the team's meetings or practices. This is probably due being unaware of incremental development and the importance of the focus. The empowerment was semi-achieved as the project management was convinced that agile methods work well as the motivator for the team. Collocated line and project managers tried to give the team the possibility to self organize inside the increment. It was still impossible for them to fully block the requests coming from higher levels of project organization and those needs interrupted the development for several days on several occasions without much prior warning.

Without broad commitment the Sprint planning meetings during this research were only partly functioning. Agile methods prefer a single full-time customer. As the second best approach is presented that "customer has to speak with one voice". During this experimentation we had active customer representative from marketing department. He however needed also fulfill the needs of legitimacy system. Practices needed by agile development were experienced like doing the same thing twice even though agile way was seen more effective. Further it is typical for embedded projects to have difficulties identifying the customer. In the case organization the situation is even more complex because of different market areas. Key customer candidates in global organization also spend lot of their time traveling. Senior developers with domain experience typically fill in the role of customer in the beginning of a project, but this misses the non-technical perspective in planning. Without effective prioritizing it is difficult to harness the full power of an agile team.

Administrative Interface. In a large organization different views to project reporting are evident. In the case organization Gannt chart created with MS Project was normally used. Unfortunately Gannt charts as normally deployed are not useful for realistic agile team progress reporting. Incremental development can be mapped to Gannt chart (Schwaber, 2004) even without totally faking it, but defending these two different perspectives requires a competent project manager to defend the need for it. In the case project the Gannt chart was not used at all for reporting the progress during this research. This was however seen as a problem reducing the level of visibility experienced by the administrative stakeholders.

Another key obstacle for rapid development was a lack of decision making power. Developers were not fully authorized to make decisions, but a project steering committee was responsible for decision making. In practice the steering committee however failed to make technical decisions and the waiting resulted in wasted time.

Synchronization of Development and Stage-Gate Model. The development phase in the project was initially completely irrationally scheduled for only 5 weeks without any development planning. After postponing the prior gate the team had only few weeks for developing the "working prototypes" needed by the process guide. Even in that case it was preferred to use the given time for creation of paper deliverables.

Lack of information and understanding of incremental development often led to fire-fighting type of project steering.

Certain individual tasks were prioritized without overall planning. These new steering actions normally interrupted the current development increment and made the incremental planning un-motivating - or even impossible. The whole organization including higher management needs to be informed to understand the consequences of such violations in rapid incremental development.

Incremental development needs to be anchored to other processes that the organization follows. These other processes can be seen as one of the customers when identifying and prioritizing tasks for incremental planning. As seen in the case of sprint 6 it seems to be possible to emphasize the design work and still get sufficient documentation. The positive side-effect is a more committed and motivated team.

C. Conclusion

This study supported the findings in existing literature. Agile methods can be anchored to the Stage-Gate® model in embedded NPD activity with very little effort, as reported in (Wallin, Ekdahl and Larsson, 2002 and Karlström and Runeson, 2005). It is recommended that some time is spent in the beginning of agile intervention to train and involve all stakeholders at early stages and to create reporting abstraction layers for different levels. It was also experienced that taking light agile methods into use, the developer team was able to accelerate the progress and at the same time achieve a higher team spirit and motivation level.

Some friction was identified preventing fully applying agile methods, for example lack of full commitment. Some of these obstacles can not be resolved completely in the setting of this study. In a large organization there are mandatory processes that can not be easily overcome, but product development has to satisfy these needs as well. As such the agile methods can not be applied at their purest form and can not be claimed to be the cure-all or a silver bullet to development problems. However agile thinking and incremental and iterative development in more general seems to benefit cross-disciplined embedded systems development. It is recommended to continue gathering experience on applying the agile practices, but also to share the information in order to get wider buy-in. This is required as the first step in solving the difficulties identified.

Future research can include studying how the agile methods could be modified and adapted to the Stage-Gate® model, large organization project reporting, and how these different models affect each other. Several missing issues on this study need more attention, such as usability engineering and requirements engineering, balancing the up-front design, full release planning, estimation techniques, incremental testing techniques etc. Other future research should include studying ways to distribute the created knowledge and experience in a large organization. (Kähkönen, 2005) presents a life cycle model for software process improvement project that deploys an agile method to a single project in a large organization.

REFERENCES

(Abrahamson et al., 2003) Abrahamson Pekka, Warsta Juhani, Siponen Mikko T. and Ronkainen Jussi, New Directions on Agile Methods: A Comparative Analysis, Proceedings of the 25th International Conference on Software Engineering(ICSE'03), 2003.

(Beck, 2000) Beck, Kent, Extreme Programming Explained, Addison-Wesley 2000.

(Cooper, 2001) Cooper, R.G., Winning at New Products, 3rd ed., Perseus Publishing, Cambridge, Mass., 2001.

(Gibbs, 1994) Gibbs, W., Software's Chronic Crisis, Scientific American, Sept. 1994.

(Graaf, Lormans and Toetenel, 2003) Graaf, Bas, Lormans, Marco and Toetenel, Hans, Embedded Systems Engineering: State of the Practice, IEEE Software, November/December 2003.

(Grenning, 2002) Grenning, James W., XP and Embedded Systems Development, 2002 Available at http://www.objectmentor.com/home

(Grenning, 2004) Grenning, James W., Progress Before Hardware, 2004 Available at http://www.objectmentor.com/home

(Highsmith, 2004) Highsmith, Jim, Agile Project Management: Creating Innovative Products, Addison Wesley Professional, 2004.

(Hutchings et al., 1993) Hutchings, Tony, Hyde, G. Michael, Marca, David, and Cohen, Lou, Process Improvement that Lasts: An Integrated Training and Consulting Methods, Communications of ACM, Vol. 36, No.10, October 1993, pp. 105-113.

(Karlström and Runeson, 2005) Karlström, Daniel and Runeson, Per, Combining Agile Methods with Stage-Gate Project Management, IEEE Software, May/June 2005.

(Kähkönen, 2005) Kähkönen, Tuomo, Life Cycle Model for Software Improvement Project Deploying an Agile Method, ICAM 2005 International Conference on Agility, 2005.

(Lindvall et al, 2004) Lindwall, Mikael, Muthing, Dirk, Dagnino, Aldo, Walling, Christina, Stupperich, Michael, May, John, Kiefer, David and Kähkönen, Tuomo, Agile Software Development in Large Organizations, IEEE Computer Society, December 2004.

(Manhart and Schneider, 2004) Manhart, Peter and Schneider, Kurt, Breaking the Ice for Agile Development; An Industry Experience Report, Proceedings of the 26th International Conference on Software Engineering (ICSE'04), 2004.

(Patton, 2002) Patton, Michael Quinn, Qualitative Research & Evaluation Methods, 3rd Edition, Sage Publications, California, 2002.

(Pierce, 2004) Pierce, Dan, Extreme Programming Without Fear, Embedded Systems Programming, March 2004.

(Rautiainen, Lassenius and Sulonen, 2002) Rautiainen Kristian, Lassenius Casper, and Sulonen Reijo, 4CC:A Framework for Managing Software Product Development, Engineering Management Journal Vol.14 No.2 June 2002, pp.27-31.

(Schuh, 2005) Schuh, Peter, Integrating Agile Development in the Real World, Charles River Media, Inc. 2005.

(Schwaber and Beedle, 2002) Schwaber, Ken, and Beedle, Mike, Agile Software Development with Scrum, Prentice Hall 2002.

(Schwaber, 2004) Schwaber, Kent, Agile Project Management with Scrum, Microsoft Press, 2004.

(Takeuchi ja Nonaka, 1986) Takeuchi, H. ja Nonaka I., The New Product Development Game, Harvard Business Review, Jan-Feb 1986.

(Thomke, 2001) Thomke, Stefan, Enlightened Experimentation: The New Imperative for Innovation, Harvard Business Review, Feb 2001.

(Wallin, Ekdahl and Larsson, 2002) Wallin, Christina, Ekdahl, Fredrik and Larsson, Stig, Integrating Business and Software Development Models, IEEE Software, November/December 2002.

(Van Schooenderwoert and Morsicato, 2004) Van Schooenderwoert, Nancy and Morsicato, Ron, Taming the Embedded Tiger – Agile Test Techniques for Embedded Software, Agile Development Conference (ADC'04), 2004.

(Wolf, 1994) Wolf, Wayne H., Hardware-Software Co-design of Embedded Systems, Proceedings of the IEEE, Vol. 82, No. 7, July 1994.