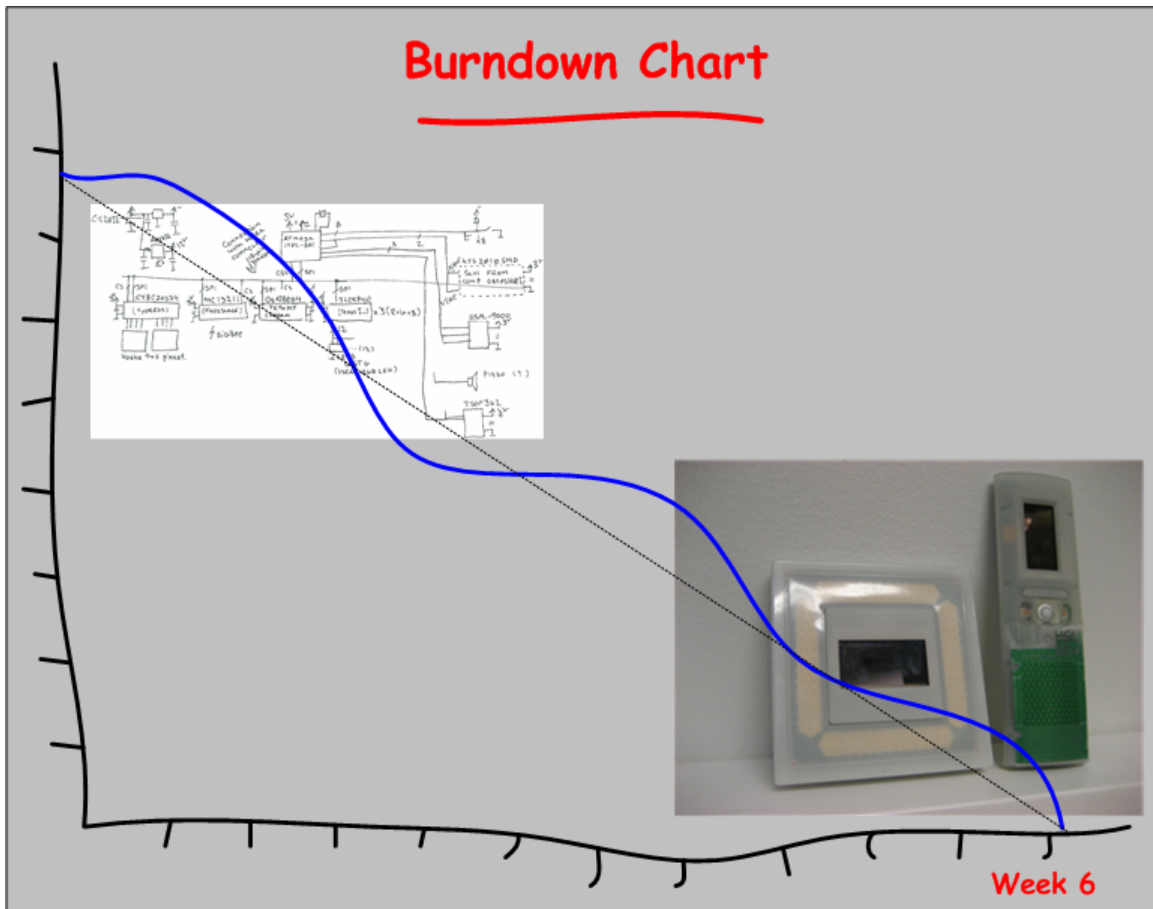


Spike Up Your Coctail

Reliable Proof of Concept with Rapid Development & Agile Practices - A Story

Timo Punkka
Schneider Electric
April 10th 2007



Abstract. Agile development is a term used for wide variety of lightweight software development methods following shared values¹. Many of these methods and practices however can be applied to more general new product development. This paper describes a six week project using some of the practices from agile development. These practices included self organizing team, collective ownership, continuous integration, iterative planning, iteration demos, team retrospective meetings, wall work queue, information radiator etc. The project crystallized a vague idea and a draft of electronics schematic into two fully functioning prototypes. It is shown that meaningful functionality can be developed in just six weeks. This is remarkable when reflected against the effort needed in so called traditional process models with formal analysis/design and theoretical proof of concept with heavy review processes. This is possible because of today's advanced development tools and prototyping technology. Stefan Thomke (2001) calls this an era of enlightened experimentation. Building the working prototype is more cost effective way of reliable proof of concept. Using this approach already in the fuzzy front-end phase of the project would result in huge savings in overall project schedule and budget. It was also noticed that all stakeholders of this project appreciated the approach and considered it as "common sense". The positive side effect was the team building effect this period had.

Introduction

Rapid iterative prototyping is considered to be a way of proactive risk management and greatly accelerate product development and lead to high-quality, defect-free quality (Clay and Smith, 2000). It has been proposed that a working prototype can work as specification, and further that it is more reliable and cheaper approach than traditional sequential process model, with up-front analysis and design. In this approach the specification can be considered as an output of a process, rather than input. Agile is a term used for wide range of incremental development methodologies and frameworks². In order to be agile the process needs to be collaborative, incremental, adaptive, and straightforward (Abrahamson et al., 2003). Examples of agile development methodologies are eXtreme Programming, Scrum, FDD, DSDM, and Crystal Family. Agile development is strongly driven by the software community, but these ideas are adapted to general new product development as well (Highsmith, 2004a and Smith and Reinertsen, 1998).

Spike is a term used in agile development literature to describe a time-boxed development activity to gather knowledge of some aspect formerly not known well enough for detailed planning or estimation. Agile projects often start with technology spikes to ensure enough competence to derive the actual initial project plan.

Within this experiment two prototypes were developed in just six weeks using the ideas of rapid development, agile practices, and the idea of spike. Development activities were

¹ Agile Manifesto <http://www.agilemanifesto.org>

² Agile Alliance <http://www.agilealliance.org>

started with very vague requirements. The idea was barely sufficient for a vision to drive the work. After just six weeks all the technological uncertainties were tackled. All the stakeholders are confident that further development towards end product would be both possible and well predictable. Further development of these products is however out of the scope of this paper.

Data is gathered as author's observations, pictures, and version control repository. A retrospective reflective workshop with 2 developers, including the in-house consultant, was held near the end of the project.

There is no objective for this paper, but to tell a story.

Setting

The Project

The aim of the project was to develop two prototypes working as platforms for innovating user interface concepts; one for wall mounting and one for remote control. The HW to-be included several different sensors and indicators; oled graphical display, RGB led array, piezo buzzer, push-buttons, capacitive trackpad, temperature, ambient light, pressure, and humidity sensors. The unit is equipped with RF communication capabilities, but this is provided as an existing module with I2C interface. The hardware for both units was similar, but remote control unit had for example an additional trackball element.

The device was to be controlled with Atmel AtMega128 uC. Developers had no earlier experience with the microcontroller or its development environment. Obviously most of the indicator and sensor components were also new.

Initially the project was scheduled to four weeks, but was later extended by two weeks "until the end of the year" (see pg. 5). The overarching vision was to develop all the HW drivers enabling later application development.

Developers

The embedded firmware team consisted of 4 developers; three internal developers and one consultant working in-house. Outsourcing was used in two different ways during the project; one pair of rented hands and an outside non-agile team (in another city in Finland). The outsourced team had one firmware developer providing parts of software to agile team. Electronics, plastic, and PCB were developed by another 4 developers. They were not members in the agile development team, but worked in close interaction with firmware team. They also attended the increment demonstration reviews and planning sessions. There were no formal project meetings or status meetings.

None of the developers had solid experience on iterative development, nor agile development. Internal developers had some experience, but no sound capabilities of doing disciplined agile techniques, like TDD, pair programming etc. Developers also chose to work in separated rooms throughout the experience. So, this experiment focused more on project management side than engineering practices.

Story

"I have to say I'm amazed about the team's accomplishment."

-Line manager

Achievement

The project was initiated with only a draft of electronics schematic, Fig. 1. The overarching vision was to develop two similar devices that would allow innovative development of user interface concepts.

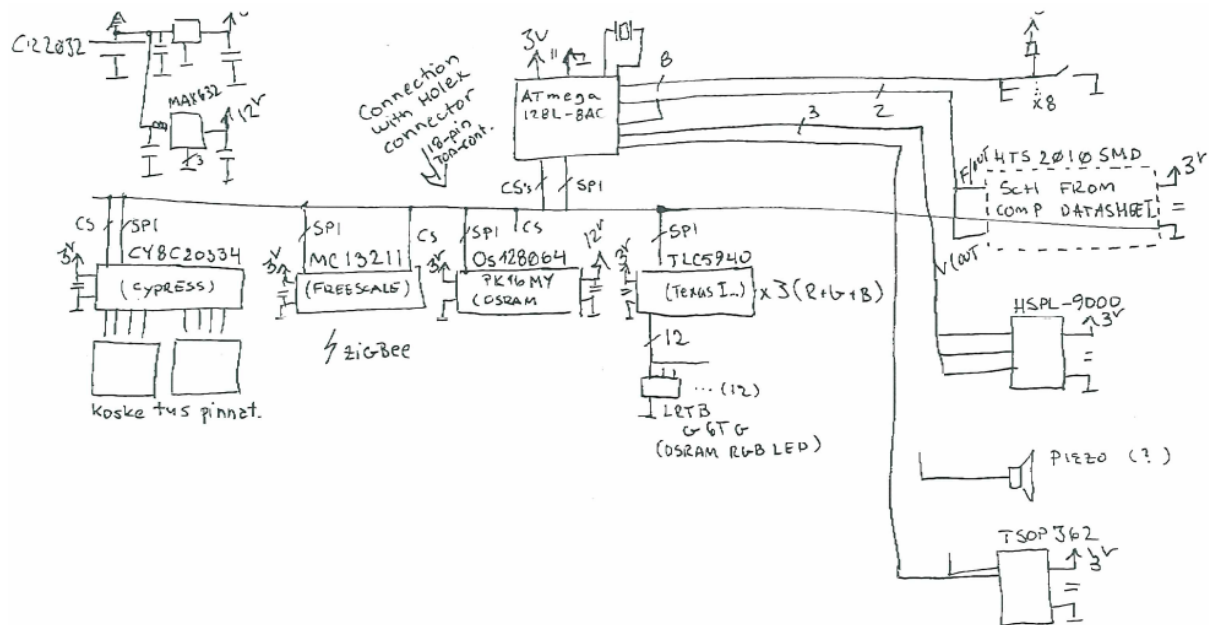


Fig. 1 The initial electronics schematic used in the beginning of the project.

During the six weeks these vague ideas evolved into two complete designs seen in Fig. 2.



Fig. 2 The developed prototypes, results of six week project.

No written documentation was used for communication during the six weeks. However, in addition to working prototypes the team provided sufficient software documentation enabling further development of software by developers outside the original project.

Timeline

The picture below illustrates the life-cycle of the project. On top, in blue font, are the major deliveries of firmware outside the team. Milestone indicators with an angle are the deliveries of actual PCB. On the bottom is the summary of each weekly iteration SW backlog.

After three weeks the consultant contract was prolonged from original four weeks for other two weeks resulting in full six week project duration. The last week was Christmas, so it actually added only three days. Based on this additional time the integration of remote controller unit was also added to local team's backlog.

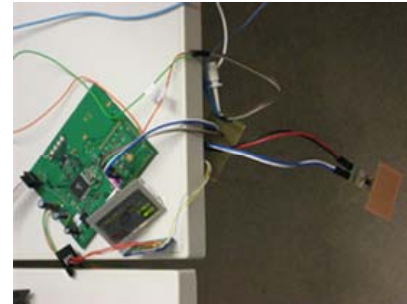
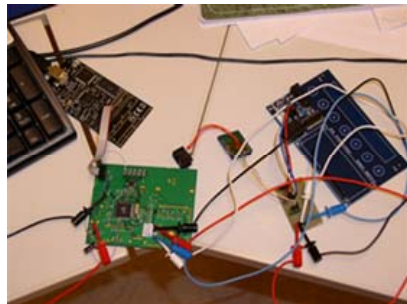
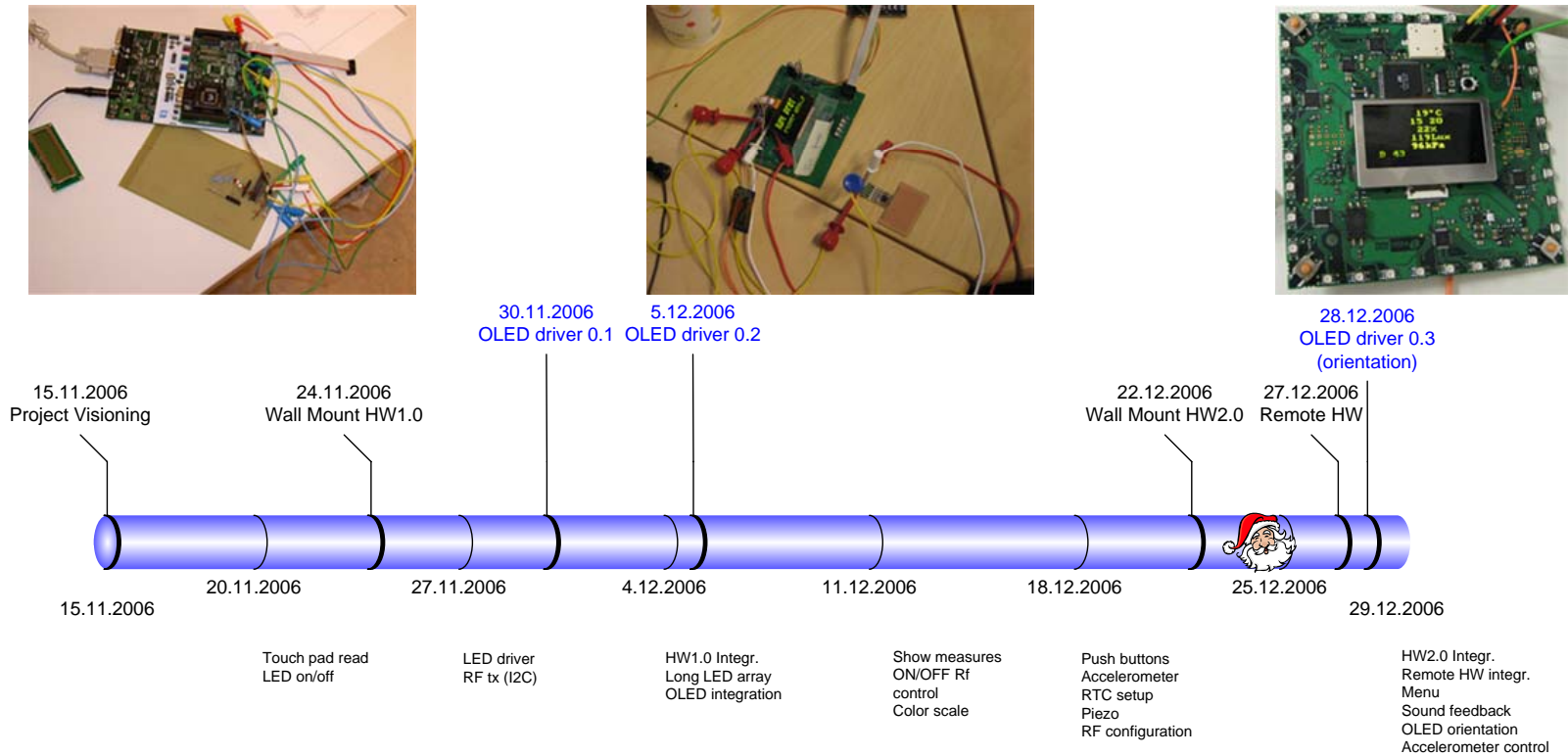


Fig. 3 Project timeline, and evolution of prototypes.

Practices

Project Visioning/Planning. Only a two hour project planning sessions was held with the developers. The initial backlog was created and themes for weekly iterations were drafted (Fig. 4).

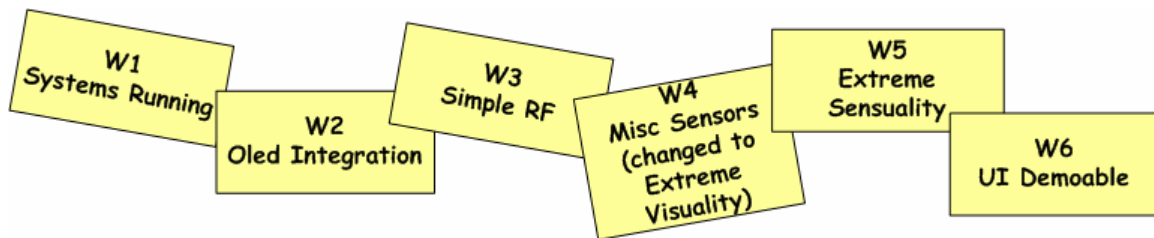


Fig. 4 Iteration themes.

Weekly Planning. Weekly planning was affected by several factors. We used the overarching vision of developing all the HW drivers as a main goal. We however needed to synchronize firmware development with outsourcing and hardware design. Technical line manager responsible for the project prioritized the development effort.

Weekly Demo. The team was able to pull out a meaningful demonstration every week. In the retrospective it was however pointed out that this may not be the case in more routine type embedded firmware system development. It depends on the project. Representatives in demos were mainly developers from other teams and technical line managers. The pressure of weekly demonstration may be too much for the sustainable pace. More experienced team might better handle this, but this is definitely not for beginners.

Wall Work Queue. Backlog was expressed as post-it notes on wall. Next to the notes were the evolving results of concurrent analysis and design. This was considered to be a good way of planning, and a wall queue was clearly preferred over electric tools, like Excel or web based project management tools.



Fig. 5 Low-tech wall work queue working as backlog.

Retrospective Meetings. We discussed briefly about possible changes in practices immediately after each weekly demo and planning session. A two hour retrospective was held near the end of the project. Objective of this session was to capture the gathered knowledge and experiences.

Information radiator. Electronics schematics, PCB layout designs, and mechanics drafts were posted to a wall where the team held daily Scrum meetings as they emerged. This helped the firmware developers to focus on HW while the actual design was still not available. Seeing the progress of other disciplines creates feeling of common goal and this works as motivator.



Fig. 6 Information radiator.

Daily Scrum. Daily Scrum meeting was highly appreciated for its knowledge synchronization affect. The meeting was following the three questions;

1. What did you work on since last meeting?
2. What are you working on before the next meeting?
3. What impediments you have?

Collective Code Ownership. Quality of code improved throughout the experience. During the first weeks the integration was a problem, and the version control baseline was broken at the time of two first weekly demonstrations. The consultant was new to this type of working, but did not see problems and agreed on the benefits of everyone knowing the whole code base, and the ease of continuous integration.

Pair Programming. Pair programming was practiced only in a form of opportunistic pair debugging. However, few times this happened, it was extremely effective in problem solving. Pair programming was not enforced at all during the experiment.

Table I Agile practices tried.

| Practice | Execution | Recommendation |
|---------------------------|---|--|
| Project Visioning | Informal, few minutes in front of white board. | You need to put some emphasis on this, even if only doing technology development. The overarching vision is needed for team building and guiding the project tradeoffs. |
| Weekly Planning | Planning sessions were short, informal, and held in front of wall work queue using post-it notes. | When planning every week you must avoid too much overhead from this practice. |
| Weekly Demo | Team members demonstrated all the new functionally added in the past iteration. | When you are having a demo every week you have to be low on ceremony, and keep the technical level according the audience. |
| Wall Work Queue | Post-it notes in the wall with paper next to for additional notes. | Use wall instead of electronic work queue. Period. |
| Retrospective Meetings | Short discussions after each weekly demo. Longer two hour meeting at the end of experiment. | You can not have long meetings for this every week. Fortunately you don't need that much time. However having some time for reflecting is very important. |
| Information Radiator | Drawings from other engineering disciplines were posted on the wall next to work queue. | Often in discussions about the design it is needed to show it from for example schematic. Having the latest version on the wall makes this available immediately. Effort is needed in keeping the radiator up-to-date. |
| Collective Code Ownership | The source code was managed with CVS from the beginning. | There is no way you can skip this. As this type of project is a learning experience, this makes sure that all team members have at least some knowledge about all whole system. |
| Pair Programming | Only opportunistic pair debugging. When used it was very powerful. | This is difficult issue to enforce. Pair debugging is one way of showing people what can be achieved with pair programming. Pair programming should be promoted. |

Summary and Discussion

Key Positive Findings

HW Evolution. The team started with microcontroller starter kits. Demonstration unit of capacitive trackpad was connected to it via I2C bus. Gradually bread board prototypes of different sensors were added to it. On week two the first actual HW arrived. This hardware was designed fast to serve SW development needs and to experiment with drivers instead of long analysis/design period. The hardware was fully functioning with minor changes. HW and PCB layout designers said that most of the problems would not have been avoided with longer analysis.

Short term goal. This was accelerating the team formation. During the six weeks the phases of team formation were clearly visible. (DeMarco and Lister, 1987) have measured huge differences between developers with same education and experience. They further evaluated that experienced developers really shine when tasks are kept small.

Concurrent co-design. At the end of the project we started to assemble 3 pieces of each unit for the final demonstration. These were developed concurrently between HW designers and FW team. First one board was equipped with just the microcontroller and its mandatory external components. While the FW team tested reset and resonator circuitry, in circuit programming and things like that, another board was assembled into a little bit further. This continued until the final demonstration day when we had one of each unit fully assembled, and FW fully ported for the final HW version. This final integration and assembly was done in just two days!

Key Negative Findings

Lack of engineering practices. TDD, refactoring, and other more disciplined agile engineering practices should be used to avoid technical debt and to deliver quality source code.

Co-design. Team members should improve the synchronization of SW/HW co-design, and further improve the cross-disciplined team formation towards shared goal.

Lack of Context. Overarching vision helped to stay on track, but it may not have been shared effectively. Developers need to understand the context. We think that we would probably have had better results in this case if we had a clear vision of even a very thin application.

Discussion

(Clay and Smith, 2000) suggest that rapid prototyping can be used to accelerate product development. They claim that biggest savings can be achieved if these practices are already introduced at the fuzzy front-end of the project. The author shares this view, as this early prototyping results in:

- Team building,
- Risk tackling,
- Improved estimation capabilities,
- Better communication (using the prototype), and
- Higher stakeholder commitment

This type of approach recommends also incremental project funding model instead of giant all-at-once funding. Jim Highsmith explains this approach in his paper (Highsmith, 2004b).

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.

(Clay and Smith, 2000) G. Thomas Clay and Preston G. Smith, Rapid Prototyping Accelerates the Design Process, pp.166-171, March 9, 2000.

(DeMarco and Lister, 1987) DeMarco, Tom and Lister, Timothy, Peopleware - Productive Projects and Teams, Dorset House Publishing, 1987.

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

(Highsmith, 2004b) Highsmith, Jim, Agile for Executives: Improving Investment and Risk Management, Cutter Consortium, Business Technology Trends and Impacts Advisory Service, Executive Update, Vol.5, No.20, 2004.

(Smith and Reinertsen, 1998) Preston G. Smith and Donald G. Reinertsen, Developing Products in Half the Time: New Rules, New Tools, John Wiley & Sons, 1998.

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