Effective Communication About Software in a Traditional Industrial Company

an Experience Report on Development of a new Measurement Instrument

Jens Bæk Jørgensen, Henrik Lauf Christensen, Søren Thestrup Hansen, Bent Bisballe Nyeng
Mjølner Informatics A/S
Aarhus, Denmark
{jbj,hlc,sth,bbn}@mjolner.dk

ABSTRACT

We consider communication between software professionals and other stakeholders in a traditional industrial company. The stakeholders include senior management, product management and hardware engineers. They have in common that they have no or little knowledge of or limited interest in software development and software as such. We describe a specific project and identify and discuss seven lessons learned about communication. These lessons contributed to more effective communication than we have previously experienced in similar projects.

1 Introduction

Our employer, the software company Mjølner Informatics A/S, has many customers from the traditional industry where the role and impact of software is increasing over time. These customers' main products can be, e.g., wind turbines, pumps for industrial use, heating and cooling installations, and lifting systems for tables and beds. Often, in industrial companies, communication has been a challenge for us. Our problems with communication have their roots in some inherent properties of software.

In Brooks famous "No Silver bullet" paper [2] written around 35 years ago, he listed four essential difficulties: complexity, conformance, changeability and invisibility. Essential means that these difficulties will be there forever, independently of progress in various subfields within software engineering. These properties make software development radically different from development in other engineering disciplines. When we develop software in an industrial context, we are working in environments where other engineering disciplines often are more well-established and have much more influence on company cultures, including the business views of senior management and product management. Furthermore, we cooperate with many stakeholders who have no or little knowledge about and experience with software development.

IWSiB'22 , May 18, 2022, Pittsburgh, PA, USA

ACM ISBN 978-1-4503-9302-7/22/05…$15.00
https://doi.org/10.1145/3524614.3528625

In this paper, we consider a traditional industrial company which has manufactured measurements instruments for many decades. The company has been a market leader for a long time and will strive to keep that position by supplying excellent products to the market. Mjølner has cooperated with this customer for more than four years. In June 2020, the customer asked Mjølner to contribute to the software development of a new measurement instrument ("Meas"). In addition, it was agreed that Mjølner should also facilitate modernisation of the customer's software development process with state-of-the-art techniques. Meas is our customer's flagship product and most complex product by far. In the Meas project, software and hardware were developed in parallel. The company is kept anonymous due to confidentiality.

From June 2020 to July 2021, which is the period we consider in this paper, 25 different persons worked on the software development for Meas. Approximately 25,000 hours were used for software development, including hours for supporting roles, e.g., software architect and software project manager. The entire cooperation was remote due to the Covid-19 pandemic.

The authors of this paper had the roles of software developers, software architect and as combined software project manager and Scrum master in the project. Our foundation for writing this paper is our own recollections of personal experiences in the project. Moreover, during the project, we have continuously gathered written material that we have used to support our memory in our analyses for this paper. Every other week, we created slides for sprint transition meetings and every other week (alternating), we created slides for meetings with the customer's senior management. This resulted in around 50 slide decks, in total 700-800 slides, in average 20 slides for the meetings with senior management and 10 slides for the sprint transition meetings. These slide decks can be seen as a project diary. In addition, we have reviewed relevant emails of the more than 6,000 emails exchanged during the project.

2 Context

Before June 2020, Mjølner had been working on designing and developing the GUI for Meas. This was a fixed price project. In June 2020, the project was changed to time and material. The software development teams – GUI at Mjølner and firmware at our customer - were merged into one cooperating unit, managed by us.

In this reorganised project, from June 2020, our mandate was to help improve the Meas project which had been running for years at

that time. Our first activity was therefore to conduct an initial project review to get an overview of the current situation. We examined scope, detailed requirements, estimates, cost, schedule, quality, test, staffing, organisation, communication, stakeholders and risks. Our customer was very aware that the project had several severe problems, e.g., unclear scope and low visibility of progress. We confirmed that.

The first improvement initiative we started was to introduce the use of Scrum in a systematic and structured way, as described in the Scrum guide (scrumguides.org). This meant that a product owner (PO) and a Scrum master were appointed and a development team with members from our customer and from Mjølner was formed. The regular Scrum meetings were organised and held, and the usual Scrum artifacts were defined. These initiatives immediately improved the visibility and understanding of the state of the project.

One important observation was that too many defects were introduced; testing had been done ad hoc until this point. The time spent on finding, analysing and fixing defects was high, and it was crucial to reduce it. Therefore, at one of the first retrospectives, it was decided that systematic testing and continuous integration should be introduced as improvement initiatives. Build servers and test servers were put in place and the team began writing of unit tests. Furthermore, during the first months, we focused on the definition of done and definition of ready, in the sense of Scrum. These important definitions were discussed, agreed on and revised several times. The definition of done stated among other things that systematic and thorough code reviews were mandatory before a task could be considered done. The definition of ready required that key stakeholders, including a product manager representing the business perspective, should sign off on the detailed requirements of features that were included in a sprint as ready to work on for developers.

In late October and early November, it became a priority that a preliminary version of Meas should be ready for release immediately before Christmas. This deadline was set in order to allow for our customer to do trial testing with their customers in early January. Therefore, it was decided to relax agile practices for some weeks, with less strict adherence to the definition of done and fewer retrospectives, in exchange of more productive time to meet this very important deadline. We decided that the unit tests could be written later, and the retrospectives could be held later. Hence, the project entered a kind of state of emergency. This had a price. After the Christmas break, the development team struggled with the technical debt that had been built up before Christmas. We used more time later to reduce the debt than what could be justified by the extra speed before Christmas. The team even failed to meet the deadline.

During January and the first part of February, the overall progress in the project was slow, because a lot of technical problems were encountered and had to be solved. In mid-February, the project situation had been improved and the status was re-assessed. It was decided to try to deliver the software in the spring. That was not possible, for reasons we will describe later.

In late February, the project entered a period of stability characterised by the fact that sprint after sprint, the work planned was also approximately the work done. In July 2021, Mjølner's engagement was reduced, because the customer was ready to take over the project management. The project was finalised by a Scrum team with developers from our customer and three developers from Mjølner. Launch is being prepared as we finish this paper.

About the project organisation: In June 2020, there was a software development team at our customer's offices with a software project manager and 8 developers. During late summer and fall, this team was increased with Mjølner developers, and around Christmas, there were 18 developers in total. The detailed organisation has varied, but most of the time, the developers have been split into three sub teams. The team members worked from four different geographical locations: Our customer's home country, Denmark, Israel and Egypt. The team members had seven different nationalities. Each of the sub teams had a coordinator. The sub team coordinators had daily meetings with the product owner, the software architect and the combined Scrum master and software project manager.

The software team worked in a context with many stakeholders including senior management, product management, hardware engineers, and an overall project manager who was responsible for the entire engineering project, including software, but also hardware, electronics and mechanics.

3 Lessons Learned

In the Meas project, we treated communication between software professionals and other stakeholders as equally important as developing the software itself and as improving our customer's software development process. We now identify and discuss seven lessons learned about communication which have contributed to make it effective. We enumerate them consecutively and set them in italics below.

We start with two key lessons about setting the stage for effective communication in general. *Lesson 1: be humble and explicitly acknowledge and communicate that software professionals often have problems with communication*. With our experience with communication problems in previous projects in industrial companies in mind, we set the stage differently for the Meas project. We explicitly explained to many stakeholders that there are significant differences between development of software and the work done within other engineering disciplines. Moreover, we told them that we, the software professionals, wanted to improve on this deficiency. If there were communication problems, it was our responsibility, because then we were not explaining ourselves sufficiently well.

Lesson 2: choose a time-and-material agreement form, if possible. This lesson is, in essence, equivalent with the agile manifesto's statement "customer cooperation over contract negotiation" (agilemanifesto.org). We include it because we want to stress its high importance, and because it was applied successfully in the Meas project, in June 2020, when the agreement form was changed, and major project improvements began and gradually had effect.

3.1 Communication With Senior Management

The software project manager had fortnightly meetings with the customer's senior management. This included the technical director and often the managing director. The overall project manager always participated. The technical director had a background in mechanical engineering and the managing director in chemical engineering; thus, none of them were software professionals. These meetings were crucial because they offered an opportunity to describe the status and the progress of the software development. Most of the key decisions were taken at these meetings.

During the first 3-4 months, we used the senior management meetings to report on the improvement initiatives we initiated. For every meeting we brought an updated risk list, which was presented and acted upon. We also focussed on the progress made in the sprints, based on the assessments we could make. Several of our meetings were difficult because we had bad news to report. *Lesson 3: always present bad news very thoroughly, including clearly understandable reasons (not too technical) and always present possible solution proposals with clear descriptions of advantages and disadvantages. Include a solution proposal not requiring an increase of budget, such that if the budget is final, then this option is the way forward.*

A key point for our customer was to know when it would be possible to finish the Meas software project. When we analysed the scope, requirements descriptions, backlog, and re-estimated, it was evident that it would not be possible to deliver by November, which was the goal. It was our impression that the deadline was the most important factor, so we initiated an investigation of whether the scope could be delivered in two stages – a first version in November and an upgraded and extended version later. This turned out to be impossible. The subject was discussed with sales and marketing, and we learned that the features in the backlog all were the features promised to the market. Therefore, making a scope reduction was not a viable option. Consequently, the answer was that delivery in November would not be possible. This was disappointing for senior management, but they understood and accepted it.

A couple of months into the project, we presented the observation that the backlog was growing with high speed due to defects being introduced and discovered later. The technical director said: "Now I can see why we have been doing wrong for so long". He understood one of the main reasons that the project had not been under sufficient control, and his appreciation of our initiatives with introducing continuous integration and the definition of done grew. We were even able to present numbers that made the case very clear. Before we joined the project, the software development team created around three defects per day. Some months later, when the improvement initiatives had effect, that had dropped to around one defect per day. This was a major improvement because, as a rule of thumb, it took on average two days to find, analyse and fix a defect.

Between December and February, senior management's confidence in us decreased. They explicitly told us that, and the reason was that the project was hit by several deeply rooted and very time-consuming problems. Therefore, we were not making good progress. On the contrary, there was a stagnation or things were moving backwards. Some of the problems had their root causes in hardware, which is an element that the software depends upon but that the software team cannot control or change. The problems were solved in February, and the project again began to move forward with good pace.

Management now requested delivery in late March. Again, we re-estimated the backlog and concluded that in the current situation, things were not in balance. Our estimates, the development capacity and the deadline were not in accordance with each other. Therefore, various options were considered, and it was decided to reduce the scope; not by excluding features or functionality, but by doing less work in improving internal software qualities. A modularisation and generalisation that had been planned when we joined the project was removed from scope. We presented and discussed 4-6 different solution options and our customer chose the one with the highest probability of making an early completion of the product, rather than the one with the best internal software qualities.

Lesson 4: keep insisting on project buffer. Each time we discussed estimates and schedule with senior management, we have insisted on a project buffer. In other projects, we have encountered busy senior managers, who are always hopeful that a project will be finished in time if only the estimates are lower than or equal to the capacity. This is of course only the case if we do not meet unforeseen problems. In the Meas project, senior management accepted this, because they had seen many projects in their company with low predictability. Obviously, the reason that buffer is a subject to be discussed with senior management is that it, in essence, is about time and money.

3.2 Communication With Product Management

In the Meas GUI project before June 2020, Mjølner (including one of the authors of this paper) worked with a product manager from the customer. His role was to clarify the requirements for the GUI. When we started to use Scrum as development process, it explicitly introduced the product owner (PO) role to the project. According to Scrum, there is always one and only one PO. Ideally, the product manager would have been the one and only PO since he had the mandate and the insight to clarify questions about requirements that the development team might have. He had the business insight needed for prioritisation.

However, he was a very busy man with many important things to do and supporting a software team was not high on his prioritised list of tasks; it was our impression that his interest in supporting the software team was low. So having the product manager as the one and only PO was not a viable option. Initially, a member of the software team was appointed as "the PO". However, in practice, the PO responsibility was shared between four people: the product manager, the overall project manager, the hardware lead and the appointed representative from the software team.

When the Scrum guide insists that "The Product Owner is one person, not a committee", as we see it, this is an ideal for a software team, but not always achievable. *Lesson 5: accept a PO committee*

and iteratively work to improve the way the PO committee works. We did this in Meas. We were aware of the risk associated with this accept, and we continuously improved the committee's ways of working. It was difficult, e.g., just to get members summoned for meetings. We realised that expecting busy PO committee members to participate in all sprint reviews and sprint planning meetings was not realistic. Sprint transitions had to be prepared and post processed in a different way. We had meetings with the PO committee members prior to the planning meetings to have them agree on the prioritisation of tasks for the next sprint. *Lesson 6: keep product management well-informed and as involved as possible, and work to align expectations.*

Early in the project, we observed that some of the PO committee member expected that everything that they had prioritised would be delivered in the sprint. We taught them the differences between sprint goal candidate, sprint goal and sprint result (increment). The sprint goal candidate is what the PO brings to the sprint planning meetings. The goal is the result of the planning. The result materialises in the sprint and is presented at the sprint review. E.g., the sprint goal candidate could consist of 10 tasks, the sprint goal of 8 of these (after the developments team's analysis and estimation), and the sprint result of 7 of the tasks, because more was planned than accomplished.

3.3 Communication With the Hardware Team

Meas is an embedded device. Access to and information about changes in the hardware was crucial for the software developers. When we joined the project, the software team was isolated in a communication silo, in the sense that the software team reported to the overall project manager, as did the hardware team. There was no direct communication between the engineering teams, so the overall project manager told the software team what he thought they needed to know about hardware and vice versa. This was ineffective and it caused delays and misunderstandings.

When the project had severe problems in January and February, we had the suspicion that many were caused by the interplay between hardware and software. In that period, we initiated two weekly meetings with representatives from the hardware and software teams. Here, we discussed planned modifications and issues of relevance for both teams. In general, this closer collaboration gave each team a much better understanding of the needs of the other team, and helped to solve the problems.

Lesson 7: establish direct communication with the hardware team – and more generally, be aware of communication silos and try to replace them with direct communication in organisations where hierarchical ways of thinking and organising are common. This is evident for software professionals with an agile mindset; it is not necessarily evident for managers and employees in a traditional industrial company.

With reference to lesson 2 – choose a time-and-material agreement form, if possible – if the agreement with our customer had been that they developed hardware and we developed software on fixed price, it would have been much more difficult to have constructive cooperation between the software team and the hardware team to solve the problems. Time should also be spent on analysing and documenting which team was causing which problems, because it would be needed in negotiations about who should pay for the effort required to solve the problems.

4 Discussion and Conclusions

In [1], Boehm and Turner discuss general challenges in introducing agile practices in traditional organisations. They list communication as one of them. The findings of [1] are described in overall terms. We have focused on a specific project and gone into details with the communication challenges we encountered. Communication as a general topic has been studied in many other professional fields, e.g., in rhetoric and as a primary field of interest at business schools. Also, literature on communication about software engineering in general exists, see, e.g., [4].

In [3], we presented an experience report about a project for an industrial customer, which was not successful, even though we delivered quality software. However, our customer got the impression that we were not able to provide reliable estimates and do proper project planning. We were not sufficiently good at communicating effectively in this project.

We have done better in Meas. The evidence for this claim is comparison with our own experiences in previous, similar projects, cf. [3]. Evidence is also our customer's testament. In October 2021, we have had a joint retrospective on the entire Meas project, with six participants from our customer and six participants from Mjølner. Among other things, our customer expressed: "Mjølner really understands the business side" (the technical director), "Mjølner has taught us to be truly agile" (the overall project manager) and "I have learned a lot from Mjølner about communication" (the PO member from the software team).

Our focus in this paper has been communication from software professionals to other stakeholders, and it is our assumption that it is difficult for such stakeholders to understand the intricacies of software development. However, we must also be aware that sometimes, software professionals may not have a proper understanding of business issues, and of other engineering disciplines. We must understand that their problems and challenges are different from what we face in software development project. In Meas, we have been very explicit in working to understand business issues. E.g., we have accepted senior management's messages about budgets and deadlines, and we have accepted the product manager's busy schedule and his priorities.

REFERENCES

[1] B. Boehm, R. Turner, "Management Challenges to Implementing Agile Processes in Traditional Development Organizations", IEEE Software, 2005, 22(5), 30-39

[2] F. Brooks, "No Silver Bullet — Essence and Accidents of Software Engineering", IEEE Computer. 20 (4): 10–19, 1987.

[3] M. Holmegaard, J. B. Jørgensen, M. S. Loft, M. S. Stissing, "Requirements Problems in the Development of a New User Interface for Healthcare Equipment," in *23rd IEEE International Requirements Engineering Conference (RE15)*, Ottawa, Ontario, Canada, 2015.

[4] I. d. F. Junior, S. Marczak, R. Santos, H. Moura, "Communication in Distributed Software Development: A Preliminary Maturity Model", *in 11th IEEE International Conference og Global Software Engineering (ICGSE),* Orange County, California, USA, 2016.