Teamwork in a first Scrum project[1]

1. Background

The team organized the project according to generally recommended Scrum practices. Plans were made at the beginning of each sprint, after the team had reviewed what was produced in the previous sprint. Features were recorded in the sprint backlog. The team held three project retrospectives to identify and discuss problems and opportunities that arose during the development process. Daily meetings were organized throughout the project, though these were less frequent in the last two sprints. These meetings were usually about updating the others on progress, development issues, and the project in general. The daily meetings we observed lasted from 10 to 35 minutes, but were usually shorter than 15 minutes. The product owner, who was situated in another city, often participated in these meetings by telephone. He participated because both he and the Scrum master thought that it was important to share information constantly and participate in the decision-making process.

The project began in May 2006, with the first installation planned for October and the final installation for November 2006. However, the first installation was not approved until December 2006 and from January 2007, two developers continued working with change requests until the final installation was approved in October 2007. Five of the sprints lasted one month, the sprint during summer for two. The figure below shows major events in the project together with a project-participant satisfaction graph. This figure was created by the team in the final project retrospective and was based on a timeline exercise. To create the project-participant satisfaction graph, each team member first drew his own graph for the emotional ups and downs during the project, after which the graphs were merged.

[1] Excerpt from: Moe, Nils Brede, Torgeir Dingsøyr, and Tore Dybå. "A teamwork model for understanding an agile team: A case study of a Scrum project." *Information and Software Technology* 52.5 (2010): 480-491.

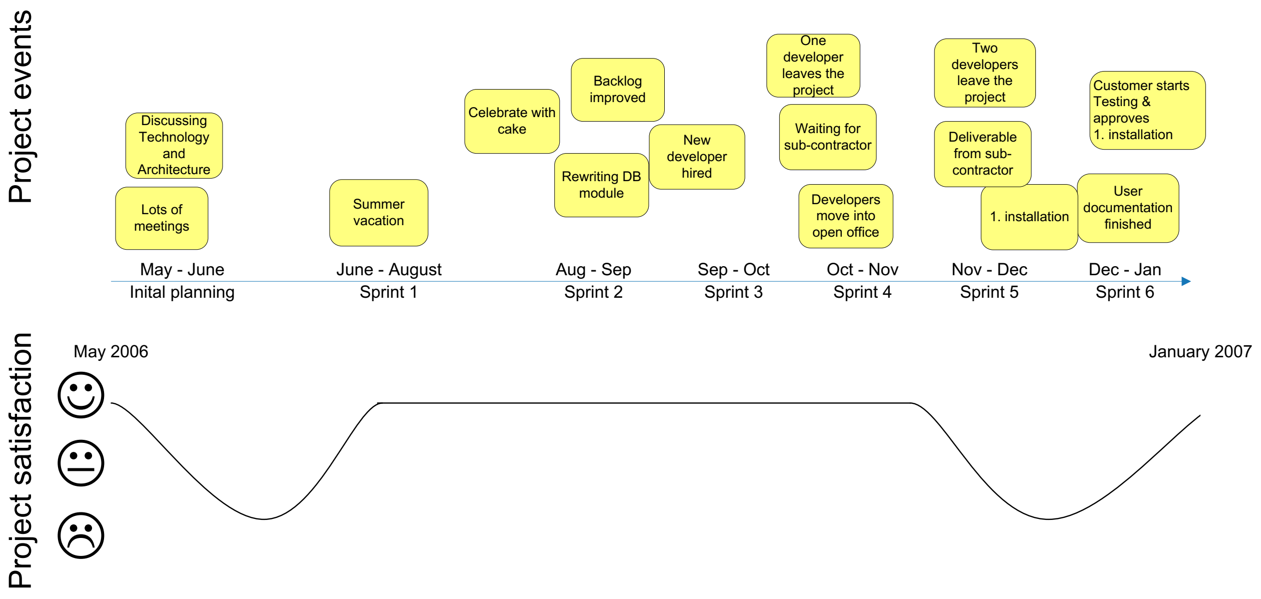

Figure 1: Main events in the project and project satisfaction

In the initial planning phase, before coding began, several meetings were used to discuss the overall architecture, and decide on the technology and development platform. As can be seen from the figure, the team was frustrated in this period, because of what the team described as "endless discussion without getting anywhere". After Scrum was introduced and code writing began, the team was more satisfied with the project. In the first retrospective, the team itself concluded that the team members were taking responsibility, that they were dedicated to the project, and that the team was protected against external issues. Meetings and work were perceived as well-coordinated. During the first retrospective, a developer said:

> Earlier we worked more alone, and when you got a project doomed to failure, you would get a lot of negative response. That was unpleasant. Now we share both the risk and opportunities.

The team was satisfied with their performance in sprints 1-4. However, in sprint 5, problems with integrating a deliverable from the subcontractor emerged, which resulted in the two last sprints being chaotic and the project being delayed. During this period, we saw many empty pizza boxes in the office space, which indicated that the developers were working late. Developers told us they also worked at weekends. The team became less satisfied, as shown in the figure.

However, after the client had approved the first installation, the team became more satisfied. In the last retrospective, the team described the project as a good one, except for the problems related to the deliverable from the subcontractor. Then again, the team saw this as something beyond their control.

Despite the teams' overall satisfaction with the teamwork, throughout the project we observed problems with completing the backlog and following the sprint plan, unproductive meetings, developers often being silent in the planning meetings, and developers often reporting working on issues other than those that it had been initially planned to work on. In addition, the developers received little feedback when talking about what they were doing. In what follows, we will use the data we collected to explain some of our observations.

2. Introducing Scrum: Sprints 1-2

The project leader participated in a Scrum master certification course. The first sprint was initiated with a two-day Scrum course. The first day was spent on introducing Scrum to the whole development department, the second on planning the first sprint.

The first sprint completed most of the backlog, and the team was satisfied with the progress. However, in the first retrospective, the team reported problems with both defining a stable sprint backlog and finishing it. We observed these problems as well. The team also ended up working on tasks that were not discussed or identified during the sprint planning meeting.

Figure 2: From the review and retrospective meeting in sprint 2.

In this company, each team member is usually assigned to work on a specific software module from the beginning to the end. The advantages with this structure are that it is organizationally simple, it allows many tasks to be completed in parallel, and task responsibilities can be clearly defined and understood. The Scrum master subscribed to this view [interview]:

> Let the person who knows most about the task perform it! We cannot afford several people doing the same thing in this project. We need to continue working as we have done before.

The team mostly kept this structure after introducing Scrum. A developer said [interview]:

> Because we have to deliver every month, there is never time to swap tasks.

Because of the division of work, the developers typically created their own plan for their own module, often without discussing it with the team. A developer commented [interview]:

> Some are more motivated by the perfect technical solution, than thinking of when things need to be done.

In this phase, one developer even implemented features for future projects, without informing the others. This was discussed in a daily stand-up of the second sprint:

> Developer: The customer databases will be used by several applications, so I have implemented support for dealing with various technologies, including Oracle. It took a lot of time.
>
> Scrum master: Did we not agree on postponing this?
>
> Developer: We need this later and now it is done.

As a result of this incident, the Scrum master lost trust in this developer and started to supervise him. Consequently, the developer was not part of the team leadership any more, even when discussing modules where he was seen as the expert. We observed that he was sometimes absent from the daily meetings.

The Scrum master also observed that the team was not reporting problems. In interviews, we found that the developers thought that the Scrum master was overreacting to problems stated at the daily meetings, which resulted in the team not reporting problems when the Scrum master was present. After the Scrum master confronted the team with this issue, the situation improved. However, for the rest of the project, the Scrum master still felt that problems were reported too late. This was confirmed by our observations of daily stand-ups.

In the second retrospective, we found two more reasons for problems not being reported: problems were discovered late and they were seen as personal. One developer said:

> People working alone results in the team not discovering problems, because you do not get feedback on your work.

Because of the isomorphic team structure, the developers perceived new emerging tasks and new problems as personal; as a result, they did not seek assistance when needed. They focused on their own modules. In the second retrospective meeting, one developer said:

> When we discover new problems, we feel we own them ourselves, and that we will manage to solve them before the next meeting tomorrow. But this is not the case, it always takes longer.

When individuals are independent and have more control over their schedule and the implementation of their tasks, there is less interaction between the group members. One developer said [Retrospective sprint 2]:

> When it comes to the daily scrum, I do not pay attention when Ann is talking. For me, what she talks about is a bit far off the topic and I cannot stay focused. She talks about the things she is working on. I guess this situation is not good for the project.

In this phase, the team spent more than 100 hours rewriting a module. The developer responsible for the module said [interview]:

> I was supposed to create a database that every project could use. After I had created it, I explained how it was done during a stand-up, and then I went on vacation. Later, when they started using it, they did not understand how it was supposed to be used, and they decided to rewrite the whole module. The team had probably not understood what I was talking about when I explained the database in the daily stand-up. If I had not gone on vacation they would not have needed to do the rewriting … another problem is the daily meeting. It's only a short debrief, there is never time to discuss what you are working on.

The developer did not verify that the team had understood how he had implemented the module and no one gave feedback to the effect that they did not understand how the module was implemented during the stand-up. The consequence was reduced progress and team efficiency.

In the second retrospective, the team concluded [retrospective report]:

> The team must work more on the same tasks, and then no one will sit alone. Working alone results in knowledge not being disseminated, and there is no backup. Also, problems are being discovered late and developers not getting feedback on their work.