Job Satisfaction and Motivation in a Large Agile Team

Bjørnar Tessem[1], and Frank Maurer[2]

[1]Department of Information Science and Media Studies,
University of Bergen, NO-5020 Bergen, Norway
bjornar.tessem@uib.no
[2]Department of Computer Science,
University of Calgary,
2500 University Drive NW
Calgary, Alberta,T2N 1N4 Canada
maurer@cpsc.ucalgary.ca

Abstract. Agile software development processes emphasize team work in small groups as one of the features that contribute to high software quality and knowledge dispersion among developers. Research supports claims that agile methods also lead to higher motivation and job satisfaction among developers. Research in workplace psychology indicates that factors like autonomy, variety, significance, feedback, and ability to complete a whole task are significant factors to ensure satisfaction and motivation among workers. In this case study, we show, through the analysis of semi structured interviews with software developers and business representatives, that large teams continuously adapting the SCRUM methodology are able to ensure these empowering factors, and thus ensure a staff of motivated and satisfied software developers. The study presented is based on data from an agile project involving 70 people (including 30 developers) building a software product for the oil & gas industry.

Keywords: agile software development, large teams, SCRUM, job satisfaction, motivation, qualitative case study.

1 Introduction

Agile methods have become increasingly popular in the industry, but have also been struggling with the perception that they are not applicable for larger projects. Some have tried to show how agility can be ensured also in larger projects [4], but research on what factors are essential to help with agility in larger teams are scarce.

In this paper, we use knowledge from workplace psychology combined with data from a detailed case study to understand whether essential factors for agility can be present in large development projects and how these factors can be ensured in such a project. Or chosen approach is to look for the five critical factors of Hackman and Oldham's Job Characteristics Model (JCM) [5] in our interview data and explain why and how these factors are maintained in our case study.

We continue by presenting the SCRUM methodology and attempts to scale agile methods in the next section, and proceed with a more thorough description of JCM in

Section 3. In Section 4 we describe the case, which is a team that develops software for the oil & gas industry. We describe our analysis and arguments in Section 5, and discuss the results in Section 6, before concluding.

2 Agile methods and SCRUM

Quite a few agile methods are used in industry. XP [2] is perhaps the most well-known and most widely used approach, but others also have their adherents. In particular, the SCRUM framework for managing projects [8] is commonly used. In a sense, SCRUM is more of a management methodology that encapsulates the daily practices of software engineers into a project structure. Many or all of the practices found in XP can thus be included in a SCRUM process, or the team may find other ways of doing the daily engineering work.

A SCRUM project is divided into iterations called sprints, lasting about four weeks. A backlog of things do be done is basis for planning. These "things-to-be-done" are usually called user stories, which are to be considered requirements for the software system, but may also be other tasks like bug fixes. The team estimates the work needed to do the jobs in the backlog, and a subset of them are prioritized and scheduled for the next sprint. The developers choose jobs to work on from the prioritized set of jobs, and report to the team on progress and impediments in short daily meetings. At the end of the sprint, the team goes through a retrospective meeting, where they demonstrate the software, assess the progress of the team and its work practices, and suggest and decide on improvements to be tried.

One issue in agile software development is how well methodologies scale to large projects with more than about 10-15 developers. Some authors, like Cockburn [3], indicate that scaling of agile methods is problematic, and difficult to realize due to the coordination issues met. Still, a concept like "SCRUM of SCRUMS" is found to be useful for making larger teams agile. Such teams are split into smaller sub-teams who run local a SCRUM process, and the team leads from each sub-team participate in the higher level coordinating SCRUM process - called "The SCRUM of SCRUMS" [8].

3 Group work research and agile methods

Within psychology, studies of the organization of work in groups and teamwork are many [1]. Using such perspectives, our concern is on how well-organized groups will display properties that result in employees who are more motivated and satisfied with their jobs. The assumption is that this in the next step will lead to productivity gains or other gains for the business. Within this research tradition, the factors

- Autonomy: the ability to define and solve your own work tasks
- Variety: the ability over time to work on different tasks.
- Significance: The ability to influence the result of the work process.
- Feedback: The ability to get meaningful responses to your efforts.

- Ability to complete a whole task: The ability to work on a task until it is complete without being removed or reassigned to other work.

are shown empirically to result in both higher motivation and job satisfaction for the employees (Job Characteristics Model (JCM), [5]), which again has been shown to lower turnover in the workplace [7]. Lower turnover has a significant cost effect on companies.

Job satisfaction and motivation are claimed to be one of the main effects of using agile software development methods, and this is confirmed by Melnik and Maurer [6] in a comparison of agile and non-agile software developers. In an ethnographic study of an XP team by Sharp and Robinson [9], we see how and why agile methods in fact contribute to create a work environment where the developers are highly motivated and satisfied with their work situation.

We can assume that the five factors of Hackman and Oldham's JCM model are present in smaller agile teams. But as small agile teams use the physical proximity and direct communication as means to ensure these factors, we are interested in how they are ensured in larger agile projects, where the number of participants requires more structured coordination and communication across sub-teams. This is the goal of the rest of this paper, where we see in a case study how motivation and job satisfaction is assured in a large scale SCRUM project.

4 The case

The development project we are referring to in this study, was run within a large ICT company. The project's goal was to develop a production accounting system for the petroleum industry to distribute produced value among shareholders in oil and gas wells, using the terms gas allocation/oil allocation. While old legacy software for this area exists, several oil companies formed a consortium to develop common oil and gas allocation software using Java™ technology. The project started in 2005 and was supposed to deliver the complete software in early 2007 with a total effort of about 150 person years. The project started with only a few developers, and grew until it was the work place of a total of 70 persons including 30 developers plus business representatives, quality assurance people, managers, and technical support.

The development team followed a SCRUM methodology using XP practices like user stories, pair programming, and test-first design. In the beginning, the project was run as one single team with traditional SCRUM practices. At one time in the process, the team became too big because of an increasing amount of requested features from the business side. A need for specialization was recognized, and the development team was divided into three sub-teams. In addition, the project managers appointed an architecture and refactoring team consisting of earlier team leads, a user interface team, and several persons with specialized assignments. The project thus is run as a SCRUM of SCRUMS project. Working with the development teams were eight spocs (SPOC = Single Point Of Contact), as they were called in the project. The spocs were representatives from the industry and customer representatives in the team. They had the responsibility to come up with requirements or user stories. Also working with the developers were quality assurance persons (QAs), who also had substantial business

experience and knowledge. Their responsibility was to test the software to see if it fit the intentions of the requirements and report bugs to the developers. Each development team had a couple of spocs and a couple of QAs assigned to them.

The physical setting of the team was a large open space where teams of developers were placed with computers around a large table, together with the associated spocs and QAs. Around this open space there were meeting rooms, management offices, and rooms for equipment, as well as other facilities.

5 Observations and findings

The data we use for our analysis is mainly five semi-structured interviews gathered in this company. The interviews are part of a larger study where we focus on studying team work, decision making, and empowerment in software engineering. The interviewees either volunteered, or were appointed by agreement in the team after we asked for a QA and a woman developer. We interviewed three developers, of which one has a special role as a database migration specialist, one QA person, and one spoc. The interviews were done in fall 2006, a few months before the end of the project. Each interview took 30-60 minutes. In addition, we use general knowledge about the company gathered from observations at the locations, and various conversations with people involved in the project. The interviews are rich in context and opinions about the project as seen from interviewees' perspective.

The interviews were transcribed into about 40 pages of text, and analyzed through several rereads. In the presentation of our analysis, we particularly indicate support of the factors of the JCM model, as well as argue for how these factors are realized in the project. In addition we will indicate how we have used the data for looking for evidence of motivation and job satisfaction among project participants.

5.1 Autonomy

The developers' daily work was mainly pairing up with a colleague, picking a story, do some initial work to get an understanding of the story, divide into tasks, and then program tests and production code. This way of working gives the developer significant autonomy in the daily work. In between stories, developers fixed bugs. Ideally, they selected bugs from a bug registration system. However, it seems as if the QAs had a lot of influence on who was going to fix which bugs, and when. The developers respected this, but they also seemed to feel a small dislike for this.

As in other jobs, there are of course some limitations to autonomy for these developers. There are, for instance, architectural guidelines for the implementations, or the pair programming practice which is strongly encouraged by management. On the other hand, we see that confidence in people's abilities made leaders trust them with advanced tasks. One example was the database migration developer who together with another developer was responsible for developing the database migration process mainly on their own, choosing tools and automating solutions.

The spocs and QAs seemed to have a higher level of autonomy than developers, as they very much were able to work with their primary tasks their own way. In early

stages, spocs did work in a single team separate from developers, but split up and specialized into different parts of the system like the developers. The QAs also seemed to be working individually, the one we interviewed had specialized in having the large picture of the development organization, and the tasks people were working on. That way she was, for instance, able to influence the assignment of bug fixes to the right people across teams, and also ensure that related bugs were handled together. Thus, she did through her own initiatives take an informal, but important role in the organization which gave her much influence on the project's progress.

5.2 Variety

Developers in this team worked with a variety of tasks, mainly implementing user stories and bug fixes, but their job was not only pure coding: they also had to create tests, get a precise understanding of user stories in collaboration with spocs and QAs, estimate user stories, and assess & improve work processes in the project. This brought them in contact with other issues than the purely technical, and thus contributed to a varied job. In the early stages of the project, all developers would also work on all parts of the system, as they all were part of a single team. However, for efficiency reasons, the project managers – during the project - split the developers into the several specialized teams. This has of course led to less variation in the kind of business domain issues or technical problems that developers meet.

Of course the spocs and QAs also participated in a variety of tasks, but in fact it seems as if they specialized somewhat more than the developers, as they really had a narrower focus in their daily work.

5.3 Significance

All the interviewees showed a clear understanding of the significance of their own and the other project members' role, although they also were aware of their own personal replaceability due to the spreading of knowledge in the team. Everybody's knowledge was considered important at meetings, which were highly informal.

Developers were occasionally able to influence the content of user stories, because by combining their domain and technological knowledge they were able to see simplifications or improvements to the requirements. There is an asymmetry here, because spocs and QAs were not able to influence the developers' way of solving problems. The spoc reported some frustration about this, because he felt that with his knowledge of the domain he could see software designs that obviously would have to be refactored later when new stories came up.

The team members were also able to significantly affect the work of the other teams through the project wiki. Another example is the QA interviewed, who had an understanding of special knowledge within the different teams, and were then able to distribute work to the right person not only in her own team, but in the whole project.

The interviews indicate that many of the changes in the organization seemed to originate in dissatisfaction among the project members, and when the pressure built up, the issue would be attacked by someone. This happened around the demo sessions

that were held at the end of sprint. The number of participants and the amount that should be demoed made this event grow to a size so that many felt it was a waste of time. The demo meetings were removed from the process and today the developers only have ad hoc demo with the appropriate spoc(s) when user stories are completed.

5.4 Feedback

The most important form of feedback found in the project was the daily feedback given in the direct communications between developers, QAs, and spocs in order to get a common understanding about what the content of the user stories was. The QAs and spocs also had direct contact with their colleagues in the other teams to get feedback on their work. The role of feedback in the sub-teams seemed to be invaluable. Developers tried to get feedback on their work results often and early.

From our data, there seemed to be little feedback about solutions and technological issues between developers in different teams. Such feedback would presumably take the way around the team leads. The project members, however, did get regular feedback on their own progress from the project management.

All the interviewees mentioned the very friendly tone among team members, indicating that the project was almost conflict free. Although there were misunderstandings and disagreements, these were solved in an open way, with respect for each others views and with good solutions as common goals.

5.5 Ability to complete whole tasks

The allocation of work in many agile teams and also in this team makes it easy for developers to identify with tasks that have been fulfilled. The user story represents a task that produces a visible part of the software. This is somehow opposed to some other practices for work assignment within software engineering where developers are given a specification for parts of the solutions, and do not have full responsibility in completing the whole requirement. In this project, the completion of user stories was involving some ceremony as the developers would give a demo to the spocs. Thus, the surrounding organization acknowledged their contribution. Spocs and QAs did not have this type of task completion, but worked more continuously on providing user stories and testing the software. Still, in particular for the spocs, the completion of a sprint marked a completion of work, as they worked both alone and collectively to finalize new user stories for development in the next sprint.

An interesting story suggesting the importance of identifying with a task, and developer's ability to complete it, is when the project due to time pressure, started a practice that was called double pair programming. Two developers would work on a user story each, whereas a third developer would alternate in supporting the two programmers. Soon, people were all doing single programming. Most likely this was due to the fact that the third programmer did not feel responsible for any of the stories the two others worked on. The practice was abandoned because of this, they returned to pair programming, and the interviewee telling about this reported a significant improvement in both productivity and quality as a consequence.

5.6 Motivation and job satisfaction

Statements from the interviewees about a willingness to work hard to complete their tasks within the defined sprints indicated high motivation among the team members. The interviewees used positive phrases like challenging, 'I like the software', friendly, and gently about the project and the development environment. A very positive statement from one of the developers verifies high job motivation and satisfaction:

"*I think the work, the environment that we have here is quite interesting. I think I'm very happy to be here. I'm honored to be on this project. I think it is a big challenge to me. I don't think there are a lot of companies out there doing something like this*." (Developer, woman)

While discussions with some team members indicate that there was low turnover in the project, we do not have any hard data whether this perception was actually real.

6 The project as a growing organism

Through the analysis, we have seen that this particular project has been able to maintain the critical factors of JCM, as well as job satisfaction and motivation, and thus the JCM theory has been shown to be applicable. A perhaps more challenging problem is to explain what has happened in this project to keep the agile values, at the same time as coordination and efficiency issues had to be considered for the growing team. Our suggestion is to use of an analogy to growing organisms. They are initially small. As they grow, they adapt their shape to the surroundings or go through a restructuring. But as the organism approaches its end of life, it stabilizes into a fixed form which is not changed much in the last phases of life.

To see how this fits, let us go through the project history. The team first started with only a few programmers, with everyone working in the same group. However, as the need for more developers led to a larger team, it was evident for the management that a division into specialized sub-teams was needed. This way, daily work in small teams was maintained. When that split occurred, spocs worked in a separate team to create user stories. This lead to a distance between developers and spocs, and important feedback disappeared. The solution was that spocs specialized too and were placed together with the separate teams. Thus, direct communication between developers and business representatives was strengthened again. A significant change for all parties seems to have been the removal of the large demo sessions. This was a change that was initiated from within the teams, but was accepted as management saw that the demos did have little value. A late change was the introduction of the database migration task, which involved one and a half person. A more efficient method for repopulating the test databases was needed, and the management asked for volunteers for this task. Towards the end of the project, we also see that people were expected to stay within their special fields. The mature process did not change much anymore and team members just didn't see any value in the retrospectives. A more rigid organization developed as everybody worked hard towards delivery.

Case studies are often problematic to transfer to other circumstances, and whether this growing organism model may transfer to other projects remains to be seen, as the same practical solutions may not be possible in other projects. However, in our analysis, we see that essential values of agile teams like small team size, open communication, participation in decisions, and voluntarily choosing tasks has been maintained, and ensured the motivation and job satisfaction we would like to see.

7 Conclusion

In this qualitative case study, we have shown how a large SCRUM team can show properties usually found in smaller agile teams, namely high degrees of motivation and job satisfaction. We have looked for the essential factors supporting these and verified their presence in the case project. We demonstrated that autonomy, variety, significance, feedback, and the ability to complete a whole task were factors prevalent in the project. The study suggests that one way of maintaining agility in software development as the software and the developer team grows, is to let it grow slowly like an organism, where management continuously takes into consideration both business value and motivation and job satisfaction issues when deciding upon changes.

References

1. Batt, R. and Doellgast, V. Groups, Teams, and the Division of Labour: Interdisciplinary Perspectives on the Organization of Work. In The Oxford Handbook of Work Organization. S. Ackroyd, R. Batt, P. Thompson and P. Tolbert, eds. Oxford: Oxford University Press (2004)
2. Beck, K. and Andres, C. Extreme Programming Explained: Embrace Change (2nd Edition). Addison-Wesley Professional (2004)
3. Cockburn, A. Agile Software Development. Addison-Wesley Longman Publishing Co., Inc (2002)
4. Eckstein, J. Agile Software Development in the Large: Diving Into the Deep. Dorset House Publishing Co., Inc. (2004)
5. Hackman, J. R., and Oldham, G. R. Work Redesign. Reading, MA: Addison-Wesley. (1980)
6. Grigori Melnik, Frank Maurer: Comparative Analysis of Job Satisfaction in Agile and Non-agile Software Development Teams. In Pekka Abrahamsson, Michele Marchesi, Giancarlo Succi (Eds.): Extreme Programming and Agile Processes in Software Engineering, 7th International Conference, XP 2006, Oulu, Finland, June 17-22, 2006, Proceedings. Lecture Notes in Computer Science 4044 Springer-Verlag (2006) 32-42
7. Mobley W. H. Employee Turnover: Causes, Consequences and Control, Addison-Wesley, Reading (1982)
8. Schwaber, K. 2004 Agile Project Management with Scrum. Microsoft Press.
9. Sharp, H. and Robinson, H. 2004. An Ethnographic Study of XP Practice. Empirical Softw. Eng. 9(4) (2004), 353-375.