First experiences with Large Scaled Scrum

https://howtosurviveasaprogrammer.blogspot.com/2019/04/first-experiences-with-large-scaled.html
(also see Marcell's excellent followup article at
https://howtosurviveasaprogrammer.blogspot.com/2019/10/experiences-with-large-scaled-scrum.html)

by Marcell Lipp

Two months ago I started at a new company. At this company we are following a working model which is much different than anything else I experienced before. This working model is called Large Scaled Scrum (LeSS). At the beginning it was very strange for me, it was totally against the mindset I had before. Right now I think I could already change my mindset partially, but I still have a lot of challenges.

What is LeSS?

LeSS is an Agile framework which is scaling Scrum for really huge projects. It is trying to keep the organization as simple as possible without a big hierarchical system. It is mostly following the classical Scrum events (sprint planning 1-2, daily standup, retrospective, review, backlog refinement etc.). Main values the are multifunctional, self-leading teams, learning and flexibility. You can find more info on page https://less.works/.

What does it mean in practice?

What I am describing now is the adaptation used in our organization. It can be that it is sometimes differing from the official theory of LeSS.

As I started at the company I became an official LeSS practitioner by taking part on a 3 day long training. Here I became familiar with the whole theory. Afterwise I joined an already existing team.

Our team has 8 developers and a scrum master who is also taking part in the development. Our task is basically end-to-end development: clarifying the tasks, architectural decisions, development, testing etc. The main concept is that none of the tasks are assigned to individual developers, they are all assigned to the team. Usually we have less tasks than team members and we are doing pair or mob programming (programming in a team). That means we are not usually sitting alone in front of the computer. The goal is that all team members should have an overview on all topics which are done by the team. The responsibility is also shared within the team.

Why is it so strange for me?

I have been already working in working models where I had my own task, which I was responsible for, I needed to implement it, I needed to hold the due date and if there was a bug in my code I was responsible to fix it. Additionally in the last time I was a technical leader in my project, so I was also responsible for the work of others. One hand I enjoyed this kind of responsibility, it made me feel good to tell that: “this is a functionality implemented by myself”, on the other hand it was of course stressful as well.
In this current working structure it is not the case anymore. On most of the tasks we are working with 2-3 team mates. Really often the code is note written on my computer, it is not pushed with my name. I am not feeling myself so much responsible anymore.
It also involves an other big difference in the way of working: earlier I was often sitting alone in front of my computer, listening to music and not talking to other hours long. It is not the case anymore. 90% of my time is about communication, sitting with someone and talking about the problems. I totally did not use to do it in this way.
In fact I’m learning a lot about different topics, working on many tasks, but I will never become the expert of the topic, I will never become the owner of a functionality.

What are the advantages of LeSS?

There are several advantages of this way of working what I can see. For example, due to the shared knowledge and responsibility, if someone becomes sick or has a holiday that does not have a big effect on the results of the work: the others have also all the knowledge, so they can just continue from where it was finished on the day before. And anyway multiple people are working on the same topic, so minus one does not make a big difference.
It also saves a lot of efforts at knowledge transfer. I had always bad experiences with situations when someone left the team: all his collected knowledge needed to be transferred to the others, it took a long time and it was not always effective.
In this model it is also easier to give tasks for junior colleagues: they can join the more experienced colleagues and learn very fast.
Due to the lot of pair and mob programming the code quality is also better and a lot of good ideas are coming up during talking about the tasks.
Also by lacking hierarchy the company is saving all costs of having multi-level management.
The stress caused by due dates is also pretty clear less and the time spent by randomly surfing on the internet in working time is also much less due to the pair programming.
So I can see pretty clear advantages of this way of working.

What are the biggest disadvantages?

On the other hand it reminds me a bit on the theory of socialism: shared responsibility, everybody is just doing his best. But in the end no one is really feeling himself responsible for holding the due dates, solving problems or making technical decisions. Really often we are just talking hours long about alternative solutions and no one is there to finalize the decisions, so we are just talking and talking and not making a clear decision.

I'm also lacking the career opportunities: in classical working structures there are hundreds of roles which can be overtaken and which are making some changes, some step further into the career, here I can not see much of them.

I'm also lacking the feeling that I can tell: "it is my code". Maybe it's childish, but that's what I feel.

I also have issues with the communications: there are so many communication channels, that it is really difficult for me to collect all the relevant information.

I also have the feeling that the project is really lacking someone who has a good technical overview on the whole project, like a software architect. When we need to use an interface really often no one can tell us which interface is it and a lot of interface are duplicated.

Last but not least I would like to mention that the permanent pair and mob programming is really exhausting. It is really difficult. The team members needs very good social skills and a lot of patient, which is not typical for most of the programmers.

Summary

I didn't expect that this change will be so big for me, it was also surprising for me that a working model can be so different from everything I experienced before. With time I'm changing my mindset and my feelings regarding this working model is also changing permanently. I am often asked if I think it is more effective, than classical way of working? To be really honest, I have no idea. We are loosing time by assigning multiple people to one task, but we are winning a lot of time by reducing review and knowledge transfer time. I would like to also clarify that these are my personal experiences, other developers may have different opinion about this way of working. I plan to write again about this working model after several months, maybe my opinion will change again.

Disadvantages Enumerated For Exercise

1. But in the end no one is really feeling himself responsible for holding the due dates.
2. [But in the end no one is really feeling responsible for] solving problems.
3. [But in the end no one is really feeling responsible for] making technical decisions.
4. Really often we are just talking hours long about alternative solutions and no one is there to finalize the decisions, so we are just talking and talking and not making a clear decision.
5. I'm also lacking the career opportunities: in classical working structures there are hundreds of roles which can be overtaken and which are making some changes, some step further into the career, here I can not see much of them.
6. I'm also lacking the feeling that I can tell: "it is my code". Maybe it's childish, but that's what I feel.
7. I also have issues with the communications: there are so many communication channels, that it is really difficult for me to collect all the relevant information.
8. I also have the feeling that the project is really lacking someone who has a good technical overview on the whole project, like a software architect. When we need to use an interface really often no one can tell us which interface is it and a lot of interface are duplicated.
9. Last but not least I would like to mention that the permanent pair and mob programming is really exhausting. It is really difficult. The team members needs very good social skills and a lot of patient, which is not typical for most of the programmers.