Joël Maximilian Mai

11118561

Media informatics

## Exposé for a bachelor thesis

# How to solve the practice-theory gap in adopting an agile project management strategy for larger software businesses?

## Contents

## Tags

agility, scrum, practice and theory, frameworks, self-organization, trust, flexibility

## Motivation and problems to solve

Becoming agile is desired not only by developers anymore. Businesses all over the world try to become agile through many different strategies. Some are more successful than others. But becoming agile seems to be more difficult than thought.

To become agile, one has to first understand what agile is. The word agile stands for flexible business culture. It's about responding to change quickly instead of following a plan. Individuals and Collaboration are more important to agile businesses than processes, methods or documentation. Note the more here. It's not that, agile businesses don't have processes or documentation. Working quality software is just more important.

To make quality software and not waste any time, the high management has to trust its employees. Agile is a culture based on trust. If an employee has to ask for every change to the live system or to buy needed software tools, a business cannot become agile.

Agile is a culture based on self-organizing teams. If cross-functional teams self-organize, no time is wasted and the single increments can be developed much more quickly. Every piece of software that is developed earlier means, that the company can see the impact of the software more quickly and therefore react to the market and users. Outcome over Output. It's not about putting more out the door. You don't have to release every day. But releasing more often and in small batches, allows for more clear data to test and improve on.

Agile is no divine solution, but a process to become more successful. Often Companies look forward to Agile Coaches or even SCRUM Coaches in the hope of a divine solution that fixes their broken processes or products. But just using the Framework, Processes or Tools does not fix anything. Often these actions lead to "scrumfall", a mix of agile tools and processes that still resemble the waterfall project management structure. To fix this, the whole business has to embrace the agile manifesto and live it.

Since SCRUM is the most popular agile Framework, problems arising with using frameworks will be discussed by looking at problems using SCRUM as an example. SCRUM was developed as a wrapper for eXtreme Programming. It has worked in the companies Jeff Sutherland and Ken Schwaber had integrated them, therefore Sutherland made it a framework and since then many companies have adopted it or changed it up and integrated parts of it into their process. But why is it, that so many businesses fail with scrum, adopt it poorly, or even become less efficient with the new integrated meetings and methods?

# Status Quo

Although SCRUM is a wrapper for an extreme approach to agile development, it is often thought of as being the same as agile. It is used interchangeably. The reasons for this are plenty. For example, SCRUM is the most popular agile framework and therefore often used as an example for agile project management. In Addition, SCRUM predates the agile manifesto, which might also add to the confusion. At first, this does not seem like a big problem, but the reasons and ways to adopt agile and scrum are different. First, a company should understand agile and then decide on what framework to choose, not the other way around.

SCRUM seems to be the most popular framework simply by its simplicity. At its core SCRUM promises to only use three roles, produce three artifacts and add only three new meetings. This is just a barebone framework, used as a wrapper to manage the complexity of the value stream of any company. It promises to work as a wrapper around any existing practices. It promises to improve the quality and efficiency of the product and value stream. All these promises and the simplicity of the framework make SCRUM highly attractive to a company that wants to improve but not change too much, more or less add on top of the processes they hold.

The Agile Manifesto was written by 17 software development thought leaders. It states 4 values and 12 principles, which makes it a rather abstract and short document. At its core, it is about better ways of developing software by doing it and helping others do it. Working agile means embracing collaboration, individuals, interaction, working software and responding to change. Although they still found that they also value processes, tools, documentation, contract negotiations and following a plan. But less than the things stated before.

One of the core principles of the Agile Manifesto states that they welcome changing requirements, even late in development. In addition, it also states that the best architectures, requirements, and designs emerge from self-organizing teams.

An agile business leads to the impression of chaotic disorder. But it's not true. Everybody talks to each other, there is no wasting time waiting until a meeting is planned. Clients appear on site to test and answer questions of developers and designers. Post-its are clustered on the wall or on virtual whiteboards.

SCRUM, on the other hand, can look the same but keeps the organization structure. Meetings around the SCRUM Teams stay untouched and are kept in the waterfall manner.

Therefore SCRUM disregards two of the twelve key principles. SCRUM does not have changing requirements during a running sprint and it does not allow for self-organizing teams if put onto a team by management. If SCRUM is emerging through motivated self-organizing teams, it might work.

## Derived Thesis

How to solve the practice-theory gap in adopting an agile project management strategy for larger software businesses?

How and when should a company become agile?

How and when should a company integrate an agile framework?

How can a company solve common problems that show up from becoming agile?

## Procedure

## Project Plan

| | |
|---|---|
| September 17, 2022 | Submit Exposé |
| September 24, 2022 | Register thesis? |
| December 23, 2022 | Thesis submission |
| December 26, 2022 | Colloquium |

# Acronyms

**HCD**
Human-Centered Design

**HCI**
Human-Computer Interaction

**UI**
User Interface

**UX**
User Experience

# Glossary

## Roles of SCRUM

**Client**
Clients are the Contractors of a product. They are also known as business owners/ product owners.

**Customer**
Customers are using the product. They are also known as shoppers or users.

**Project Manager**
The project manager has the goal of ensuring that there are clearly defined project objectives and parameters and that the project team meets these objectives.

**Product Manager**
The role of a product manager is to ensure product success throughout the entire product life cycle, during and beyond the initial project.

**Agile Integrator**
The agile integrator has the task to implement agile methods into the organization and processes. He is also known as SCRUM Master. He is responsible for creating the best possible working conditions for the team and supports them in organizing themselves.

**Desinger**
Can be a Frontend Developer, UX Designer, UI Designer or else. Has the focus on the layout of the product.

### Architect

Can be a Requirement Engineer, Software Engineer, Software Architect, UX Designer or else. Has the focus on the structure and dependencies of the product

### Developer

Can be a Frontend, Backend Developer, Software Architect or Software Engineer or else. Has the focus on the technical constraints of the product.

### Tester

Can be partially automated. But every piece of software that is critical to the system has to be tested, by someone or something. Usually, Quality Assurance Department writes tests and implements this testing infrastructure.

## Definitions

### Requirement Engineer

The role requirement engineer is almost completely integrated into the product owner. But due to the product owner being the client, the requirement engineer is an intelligent tool to use to refine client-side requirements. They help understand the client's needs and translate them for the team members

### UI Designer

A User Interface Designer usually is more focused on the visual appeal than the effectiveness of the said interface.

### UX Designer

A User Experience Desinger is a skilled researcher. The main focus of his work lies on the efficiency and effectiveness with the user interacts with his interface.

### Software Architects

Software Architects are also known as Software Engineers. It is their task to develop the concepts and constructs for the developers to build. They have the bigger picture in mind and hold parts of the vision of the final product.

### Software Developer

A Software Developer develops the software based on requirements given to him. He is responsible for the code he writes. Therefore Developers are usually heavily invested in writing code that is efficient, clean and robust.

### Frontend Developer

A Frontend Developer lays his focus on the front end of a software product. He usually works closely with UI and UX designers.

**Backend Developer**
A Backend Developer is responsible for the frontend components to work together. He usually has the bigger picture in mind and therefore works closely with the architect together.

**Continuous Learning**
Continous Learning describes the culture of an agency or business to always try to learn new things and refactor their processes to become better at what they are doing. Continuous Learning is an important step to becoming agile.

## Phases of a Product

**Continuous Evolution**
Continous Evolution is the name for a software product life cycle. It describes its life cycle more biologically. Software products nowadays evolve. Maybe even becoming something that was not part of its vision in the beginning. See Facebook for example.

**Continuous Exploration**
Continous Exploration is the first part of a product. The development team makes hypotheses about the product and tests them on the live system with real users.

**Continuous Integration**
While the product is out there. The new feature becomes integrated. That's called Continuous integration.

**Continuous Deployment**
While the team focuses on developing, testing and hypothesizing about the product. The deployment of the product should be mostly automated and therefore continous deploy parts of the product.

**Continous Relasing**
Also known as Release on demand. In case of release on demand, the business decides at what time the product and its new features see the light of the day. But in case of continous releasing, the system always releases if something new is deployable. That way, new data streams into the next cycle and no time is wasted.

# References