Fog Project 2.SEM

**Patrick Fenger**

**Petru Catana**

**Aleksandr Osenov**

**Andrian Vangelov**

**08-05-2017**

[Skriv et resume af dokumentet her. Resumeet er normalt en kort beskrivelse af dokumentets indhold. Skriv et resume af dokumentet her. Resumeet er normalt en kort beskrivelse af dokumentets indhold.]

Indhold

[Project basis 2](#_Toc482980915)

[Introduction 2](#_Toc482980916)

[The company 2](#_Toc482980917)

[Our interest 2](#_Toc482980918)

[Problem definition 2](#_Toc482980919)

[Questions 2](#_Toc482980920)

[Procedure 3](#_Toc482980921)

[Resources 3](#_Toc482980922)

[Development method 5](#_Toc482980923)

[Development method requirements 5](#_Toc482980924)

[Consideration of choice 5](#_Toc482980925)

[Discussion about choice 6](#_Toc482980926)

[Conclusion 7](#_Toc482980927)

[Continuous Integration and SCRUM 7](#_Toc482980928)

[Practices 7](#_Toc482980929)

[Commit to the main branch at least one and more times a day 8](#_Toc482980930)

[Make Unit Tests 8](#_Toc482980931)

[Have an integration server that builds and runs Unit Tests at every commit to the main branch 8](#_Toc482980932)

[Implement Continuous Deployment 8](#_Toc482980933)

[Broken build has top priority throughout the team 8](#_Toc482980934)

[No one commits a broken build to the main branch 8](#_Toc482980935)

[Make sure the entire team is aware of the health of the build 8](#_Toc482980936)

[Tools 9](#_Toc482980937)

[Sprint 0: Establishing the project 9](#_Toc482980938)

[Identifying needs 9](#_Toc482980939)

[Issue 9](#_Toc482980940)

[Conclusion of the meeting 9](#_Toc482980941)

[Planning next sprint 10](#_Toc482980942)

# Project basis

## Introduction

As a final assignment to end the second semester on our computer science AP degree, we will be writing a report on a project we, ourselves, help to define. We have been partnered with the company Johannes Fog, who has given us a problem they would like to be solved with a piece of software we are going to develop.  
JF did a presentation at school where they explained the challenges of having an outdated carport website from the early 80’s.

The primary purpose of the project is to investigate the case provided by JF and to obtain academic knowledge through practically and theoretically practices. In our case, we have chosen to dig into continuous Integration and Scrum with the last mentioned being our main tool of work.

## The company

Johannes Fog was founded in 1920 by founder; Johannes Fog.  
The establishment of “Johannes Fogs carpentry merchant” in Lyngby was the first move and up until the founder’s death in 1970, it was a one-man-owned company. In 1973 Johannes’ daughter and her husband created a fund in his name. Today Johannes Fog consists of nine timber and construction stores distributed throughout North Zealand. JF specializes in all construction tasks ranging all the way from the paint on your walls to your garden furniture.

## Our interest

We have heard a lot about continuous integration through school and have gotten the impression that Scrum in general is a very exciting subject, with a lot of demand for people who understand this. Therefore, we have decided to go into depth and investigate what CI and Scrum specifically is and in collaboration with the knowledge from school, try to apply it in practice.

Since CI and Scrum are types of tools you implement in a development environment and not something that needs to be coded, we are also going to need a project we can develop in our newly created environment. For that (as above mentioned), we have been given a task by JF. They have asked us to create software able to keep up with the rising customer demands.

## Problem definition

If you want to develop a piece of software agile today, you'll quickly come across the terms Continuous Integration and Scrum. Through experience from our workplaces, we have noted that there is great interest in implementing CI and scrum solutions at companies. Therefore, we are going to investigate what is behind these practices.

## Questions

* What is the thoughts behind CI and Scrum?
* What are the pros and cons of implementing CI / Scrum?
* If you want to use CI / Scrum, which tools are necessary?
* What effect does the implementation of CI / Scrum have on the development process, both in terms of the quality of the software, but also in general?
* Which development methods are suitable to use with CI / Scrum?
* Can we develop an application using CI / Scrum?

## Procedure

We will answer the above questions through both practical work and theoretical research. We will start with finding some literature on the subject and seek out professional people with knowledge about CI and Scrum we can drain for knowledge. Next we will describe the theory found behind the disciplines. We will also investigate which tools we can/need to use to run a development project using CI / Scrum.

Once we have investigated the possibilities and decided which tools and practices we will use, we will develop an application to test our theoretical knowledge in practice. It involves setting up a development environment with the before mentioned tools, state some development methods and running the project as an actual development project.

## Resources

The school has provided us with both technical and human resources. Since Continuous Integration and Scrum depend on tools, it is necessary to have a platform that they can run on. In addition, we need some feedback from the "customer" and therefore we have been allowed to leech on the knowledge our teachers possess. Both feedbacks to the requirements for the project, but also professional knowledge about the development process, programming, etc.

## Human resources

Kasper

Tue

Palle

Marjahan

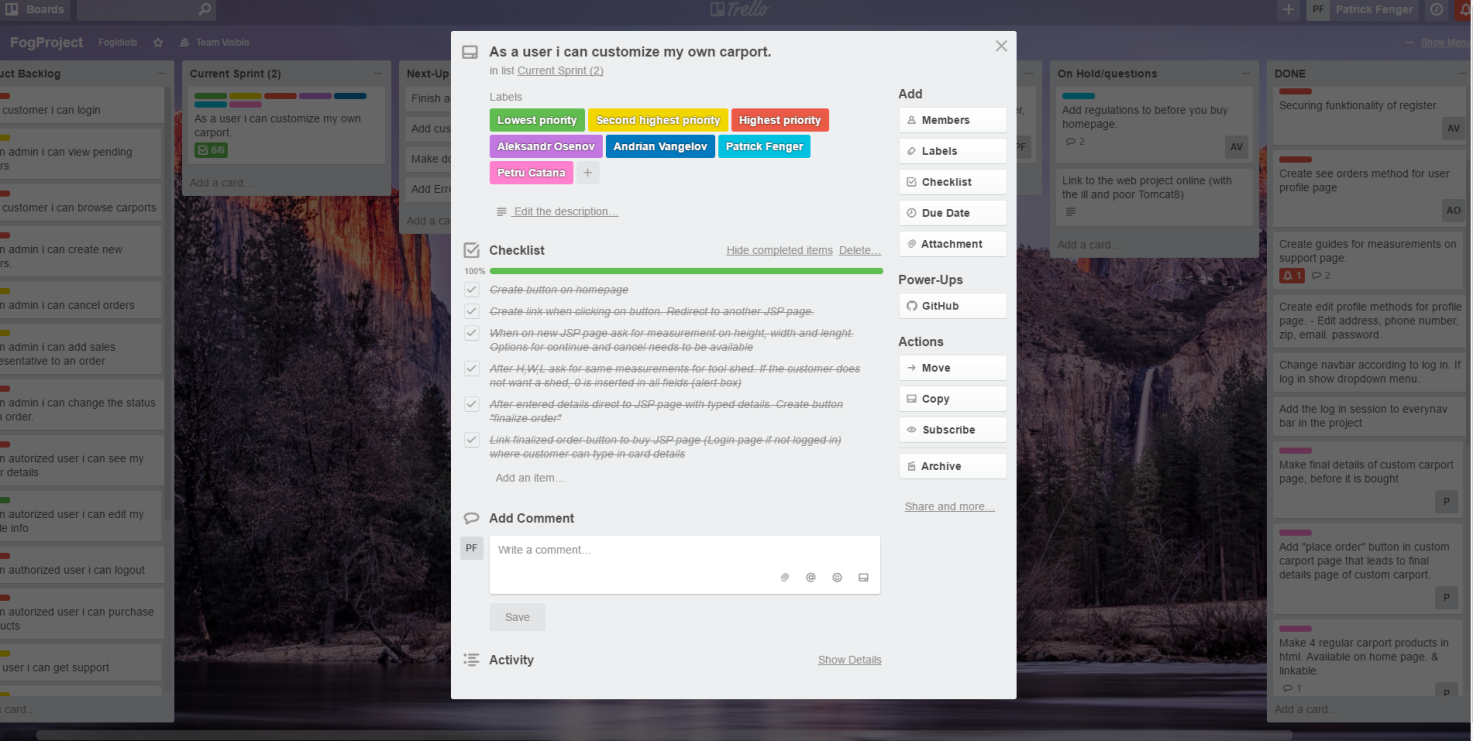
Tobias

## Technical resources

**NetBeans** We will be programming in NetBeans IDE 8.2 - Java EE editions, as it is the IDE we have the most experience with and because it’s closely integrated with the other tools we will use. NetBeans has a lot of functionalities that will be useful in our project. Generally NB has many tools that make it easy to create and develop applications in Java. To start with, we will use the free edition, as we have not found any need for the functionality of the paid versions. Should we discover the need for it we can always upgrade in the future.

**Trello** is a project management application that allows you to collaborate with team members on projects, and track the pieces of a project as it moves through stages of production. The visual layout Trello provide makes it easy to quickly see how far along all your tasks are, and what still needs to be done. Here we can create our User Stories and create our own workflows for these so that they are customized to our way of working. These User Stories can then be assigned a sprint and broken down into smaller tasks.

In Figure 1 we have used our Trello board for sprint 2 as an example. A user story has been assigned to sprint 2 and we have set up some tasks below in a checklist. When you start working on a task, you start with assigning it to yourself (attach label with the individual color) and move it to the *in progress* column. When you complete the task, you move the task to the column *done.* Should you experience challenges or find yourself in doubt, you move the task to the column *on hold / questions –* This will then be reviewed at the daily scrum meeting.

****

**Apache Tomcat** If you are developing a web application something like Quora. This app should be accessible to the out side world through the Internet or Intranet.  So you  develop application in your system. To access your application to the outside world you need a server. That server can be your Apache Tomcat.  
  
           To access the application from the outside world, it needs HTTP  protocol so all this HTTP protocol are implemented inside Tomcat Server. So it automatically take care of handling "Request" and "Response"  
  
**Digital Ocean**

**Git** Will be (except for scrum) our foundation for the project. Git is a Distributed Source Control Management (Distributed SCM) system. An SCM system is used to deploy source code for software solutions. Each time you make a change in your code, you upload (Commit) the entire change to the project repository in the SCM system, which ensures that all changes that have been made are kept on track. This way you can always go back through versions if you need to. It is also possible to make a copy of a version (Branching) and work with it parallel to the "main track" (Master-Branch). That way, multiple developers can work on each of their functionality, with full version support. Once the developer has completed his task, he can merge his branch to the Master-branch and the functionality is now part of the main track.

Git is a so-called distributed SCM system. This means that all developers have a local copy of the central repository and therefore, in addition to committing their code to their local repository, they must synchronize (Push and Pull) the local repository with the central on the Git server.

# Development method

Before we start the project, it is important to choose the right development method. Therefore, in the following section, we will look at the different development methods that we have been introduced to through practical experience and from there select elements that support the requirements we make for the method.

### Development method requirements

First, we will define the requirements we consider necessary for the method we will use during the project.

* We must be able to present a demonstrate able product relatively quickly, so we can make use of CI and SCRUM
* The method should support the use of iterations as we wish to comply with the wishes of the collaborators, but also to be able to publish regular releases.
* It's important for us to have a meeting with a partner (in this case the teachers), so we can get feedback on the work we have done.

## Consideration of choice

Since it is important for us to develop over iterations so that it is easier for us to show new implementations, we have chosen to disregard sequential development methods and only consider the iterative models. These are the following development methods we will consider:

* Unified Process
* Extreme Programming
* Spiral model

The reason we have chosen to consider these is that they support the requirements we have for the development method. They are all iterative, though they are still different as they have different approaches – An example being release frequency. We will now look at the pros and cons of the mentioned development methods and then make a choice. It is possible that we will use artifacts from other developmental methods, if they become relevant, or should they help to make problems clearer.

**Unified process**

+ Solution of risks in project associated with customers changing requirements

+ Good documentation and model tools (UML)

* Role distribution that requires you to be an expert in the role area, which makes it difficult for smaller teams to develop software using this method
* The development process is complicated and focus a lot on organisation

**Spiral model**

+ Focus on risk analysis

+ Extra functionalities can be implemented continuously

* Is best in larger projects.

**Extreme Programming**

+ Customer satisfaction through fast and continuous delivery of working software

+ Frequent dialogue between customer and developers

+ Adaptation to changing circumstances

* Can easily be thrown off track if the customer does not know the end result

We have also been considering SCRUM as a project management tool since we want to have an overview of tasks started but also the tasks that have not yet been started.

### Discussion about choice

It is important for our method to focus on iterations and quickly be able to deliver demonstrable and functional software to our customer. As the time for the project is short, it will also mean that our iterations will be short.

**Unified Process** focuses on risk analysis, analysis and design continuously, which may be useful for some projects, but we do not see it necessary as our project does not seem abstract or prohibitive. In addition, it does not support changed customer requirements as well as other methods which makes it harder to comply with changes in the project and handle them. Unified Process does not support the release of previous prototypes, and iterations can be long, simply because of the documentation that has to be done before. In order to avoid long iterations, one has to cut down on the artifacts in this methodology. Therefore, other development methods will be a better option. However, it may be necessary to use UML charts along the way to explain and communicate how the system works.

**The spiral model** supports the use of iterations. The spiral model focuses heavily on the fact that you initially carry out risk analysis and are therefore more suitable for the development of systems where the customer does not really know what is desired and where it is continuously adapted accordingly. Of course, we need to analyze and evaluate the system, but we don’t think it should take up too much in our development process, as we do not find it relevant for the defined task. We will of course use risk analysis if we later realize that it may be beneficial, but we see other methods as better alternatives than this.

**Extreme programming** supports most of the requirements we have set for our use of development method. In XP, focus is on delivering working software quickly. However, XP is new to us, but we have been introduced to other agile disciplines in the 2nd semester (Scrum). Since XP is an agile working method, this is not really surprising. Especially the fact that Continuous Integration and Scrum are disciplines specifically mentioned as rules for Extreme Programming[[1]](#footnote-1) strongly suggest that this method is the one to go with.

**Scrum** is a project management method that is designed for use with agile development methods. Scrum focuses on getting the best out of a team. The way it is done is that every day you have a daily meeting with your team. Here the plan for the day gets discussed, as well as any problems that members may have encountered. This way, the members of the team can help solve this. Again, it is about being able to adapt to changes. In addition to PRODUCT BACKLOG which is the overall amount of tasks for the project, you also have a SPRINT BACKLOG containing the user stories you have decided to be done for that sprint/iteration. Every sprint has a sprint planning meeting where you break these user stories into smaller tasks. In addition, every time you finish a sprint there should be a meeting with the product owner where you hand over the product and get feedback on the functioning software. After that, the team internally holds a meeting and discusses how the work process has been and whether the work has been adequate or needs improvements.

## Conclusion

Based on the requirements we had for the development method we would use, or at least use some ideologies, we can conclude that the one that best suits the requirements is, Extreme Programming. This is based on the practices of Extreme Programming. Including pair programming and XP's use of precisely what we want to investigate in the project here, CI and scrum. In addition, we have also taken a look at the pros and cons of the selected methods. Here we looked at whether they matched the size of the project, but also to get started with the development of the piece of software that we should have produced. It also argued that what we want to investigate in connection with the assignment here is used in the development method. However, one can argue that we could adapt a method to our project of fragments from other methods, so if we find out that we may benefit from applying practices from other methods, we will of course do that. Furthermore, we have chosen to use SCRUM as a project management method as it helps to provide an overview of the project and its tasks. At the same time, within SCRUM, great focus is on evaluating oneself all the time and taking care of problems through retrospectives. We will use this because we also believe that it should force us to reflect more about the way we work on the way. Together with our product owner, we will also host Sprint Planning Meetings and Sprint Reviews, where we plan on what to do in our sprint and on the one hand, continuously deliver the product we have developed.

# Continuous Integration and SCRUM

## Practices

Continuous Integration (CI) is a practice derived from Extreme Programming (XP). Much of our research is based on Martin Fowler's thoughts on the subject. Continuous Integration is, briefly told, a working method developed as a natural extension of the agile approach to system development by developers constantly working in small iterations and committing its code once or several times a day. The entire Continuous Integration practice is based on the development team's Source Code Management System (SCM), such as Git. In short, Git is a repository where you keep your code and all other code artifacts needed to build the application you are developing. Each time you add a code to the project, the committer adds code to its repository and can at any given time go back through commits and restore previous versions of the application.

An essential feature of SCM systems is that you can work in tracks or branches where, as a developer, you can take a copy of the application and work parallel with your own copy. This allows multiple developers to work on the same application or for separate teams to work on each feature for the same application. Once you have encoded its feature or the user story you should merge back to its main branch and everyone has the same starting point again.

To summarize the main features of Continuous Integration, here is a list of the points we have identified to be the most essential practices:

### Commit to the main branch at least one and more times a day

The idea behind Continuous Integration is that by working in small branches in its SCM system (in our case Git) and regularly integrating with its main branch, you do not have to spend unnecessarily time solving integration conflicts or merge conflicts. This makes integration faster and easier to review.

### Make Unit Tests

One of the foundations of CI is that you code unit tests. The code that will be committed must be self-testing so that you can keep an overview of the quality of the code.

### Have an integration server that builds and runs Unit Tests at every commit to the main branch

### Implement Continuous Deployment

### Broken build has top priority throughout the team

If a build of the main branch fails, the idea is that the top priority of the entire team will be to resolve the error. This is not necessarily a Unit Test that fails, but may be it. However, if you use Git, where the developer has a local copy of the common repository, the risk will be significantly reduced. We will review how Git works, later in this assignment. As a failed build in the SCM system blocks for further development, it is therefore natural that the bug fix becomes top priority for everyone in the team. If everything else goes south, you can always go back to a previous version that works - In some cases that is the best solution. However, that decision lies on the team together with PO. One way to handle a failed build is to set a certain amount of time for the error to be resolved and if the error is not corrected then change to a previous version and start over. As you often commit your code, the lost work will be manageable.

### No one commits a broken build to the main branch

An important rule is that you do not commit code that cannot be built. There should never be committable code to the main branch, which cannot compile as this will destroy the build.

### Make sure the entire team is aware of the health of the build

How to get feedback from the system

### Tools

# Sprint 0: Establishing the project

We held a meeting with the stakeholders for the application. The purpose of the meeting was to map the vision and purpose of the application, as well as designate the product owner of the company, which in this case would be our teachers, who will serve as a communication link between us and the company.

The meeting began by welcoming Tue and introducing our task, so everyone was on the same page. Since we have very limited time to work in, it was important for us to make it clear that our task was to focus on the development process and not so much on the product itself. Therefore, we will probably not be able to implement all the features that would be desired. **Table 1 lists participants at the meeting.**

Name: Title:

Tue Teacher Patrick Fenger Student  
Petru Catana Student  
Andrian Vesonov Student  
Aleksandr Osenov Student

Table 1: List of participants

## Identifying needs

Hvad har fog I dag

## Issue

Aaa

## Conclusion of the meeting

We talked about many features that could be very nice to get involved in the application, but we had to manage the meeting and stopped it after an hour. Among other things, it was suggested to make a 3D view of the selected product where customers visually could view their product based on the selected measurements. Once again, we had to remind ourselves that the project has a sharp deadline of approximately 8 weeks and that it would be a waste of time to discuss too many features since it is not realistic that they are going to be implemented. We left the meeting with a very good idea of what stakeholders needed. We also received a Product Owner. It became Tue

Week 12 27/3 marjahan sprint meeting requirements + priority

Week 14 6/4 tobias technical review

Week 14 7/4 pab/tuhe sprint review

Week 16 21/4 pab/tuhe sprint review

Week 19 11/5 koe Teknisk review  
  
Planning next sprint

# Sprint 1

## Architecture

## User story breakdown

## Unit Testing

## Testable architecture

## Creating unit tests

## Branching strategy

## Sprint review

## Retrospective

# Sprint 2

## Sprint meeting

## User story breakdowns

## Code

## Sprint review

## Retrospective

# Perspective

## Business case (Vision and goal for the IT-system, traceability model, activity diagram and domain mode)

## Architecture model

## Design class Diagram

## Sequence Diagrams covering min. one core or complex scenario.

## Description of the qualities of the design (Coupling, cohesion, the use of patterns)

## E/R diagram and schemas

## Descriptions of complex or core SQL queries and updates

## SQL scripts

## Documentation of an automated test (Junit) of minimum one class in the data source layer.

## Conclusion

## Link to server running the program

1. The Rules of Extreme Programming. URL: http://www.extremeprogramming.org/rules.html [↑](#footnote-ref-1)