IT University of Copenhagen **Date:** September 28th  
**Course:** Software Engineering **Supervisor:** Renata  
**Group:** 16 **Version:** 1  
**Document nr:** 4 **Tollgate:** 1  
**Responsible:** Carl **Status:** Ready for TG1

Portfolio document 4

Software Process Model

A person sitting on books and using a computer

Description automatically generated

**Students:** **Student ID:** **Email:**  
Carl Villads Dissing Fristam 17950 [cavf@itu.dk](mailto:cavf@itu.dk)Cecilie Malmberg 20963 [cecm@itu.dk](mailto:cecm@itu.dk)Luca Giannini 22150 [lgia@itu.dk](mailto:lgia@itu.dk)Peter Klarskov 19517 [pekd@itu.dk](mailto:pekd@itu.dk)Victoria Hinley Deng 17952 [vide@itu.dk](mailto:vide@itu.dk)

|  |  |  |  |
| --- | --- | --- | --- |
| Version | Change | Author | Date |
| 0 | Added ChangeLog | Peter | 22.09 |
| 1 | Expanded text | Carl | 26.09 |
| 2 | Changed font type | Carl | 11.10 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

Choice of agile software process model

We have chosen **scrum** as the software process model for our project. We chose this model because it provides a clear framework for the delegation of roles and work in our group and allows every member to have visibility of all tasks currently in progress, thus ensuring that everyone is aligned and on the same page. The short sprint-style of iterative and incremental work(Sommerville 2016, 87) is a perfect fit for the toll-gate style of project deliverables that we are expected to provide and allows us to get concurrent feedback on our product and project throughout the process.

Scrum is an extremely popular agile process model and one that most of the team members are already familiar with on various levels. The model is also very well documented given its popularity, so an abundance of resources is available.

Scrum also promotes self-organising teams with team members who have a sense of ownership and responsibility, leading to a high level of motivation and general productivity.

Pros. and cons. of the candidate process models

**Kanban**

Kanban is a process that requires real-time communication of capacity and a highly transparent workflow. It aims to limit work-in-progress, meaning that it aims to set the maximum amount of work that can exist in a status of a workflow at a certain limit. It’s great for continuous delivery and is suitable for both large and small projects. It is however less structured than other agile processes and leaves most out in the open. Meaning that the only constraints it provides are visualising your workflow and limiting the work-in-progress.(Kniberg and Skarin 2010)

**Discussing Scrum – pros and cons**

As outlined by Kniberg & Skarin, agile methods are generally highly *adaptive*. 100% adaptive meaning you can do whatever you want, whereas 100% *prescriptive* models let you turn your brain off, as there is a rule for everything.(Kniberg and Skarin 2010) Relatively speaking, Scrum is more prescriptive than other models like Kanban, with Scrum giving a team more constraints and leaving fewer options open, by, for example, prescribing the use of timeboxed iterations. Something that Kanban does not. As mentioned however, these qualities of Scrum are welcomed for our case, given the nature of this project. Kanban, for instance, does not introduce well defined roles, which is something we need to have established for our project. Furthermore, we believe that we will operate with a certain degree of uncertainty and changing requirements – project qualities that go hand-in-hand with an iterative software development process.

There are also cons associated with Scrum, but some of these shouldn’t necessarily be seen as cons. As Kniberg and Skarin (Kniberg and Skarin 2010) mention, different software process models differ in qualities and levels of adaptiveness and prescriptiveness, meaning their effectiveness is closely related to a project-team's ability to select the appropriate model for a project. If you choose the wrong model for your project, you limit your options, and some qualities of Scrum may suddenly become cons. If Scrum is the correct project model for a given project however, it may still be challenging to navigate and deal with the lack of detailed planning and limited predictability that Scrum results in. But this is ultimately to make room for embracing changes and be able to adapt planning more easily.

**Extreme programming**

XP (eXtreme programming) is another agile software process model deemed close to Scrum, but heavily focused on engineering practices such as test-driven development and pair programming (Sommerville 2016, 77). The test-driven development is key to define and meet requirements for ease of use and trustworthiness. These tests are both in the form of automatic tests set up by the developers (based on user stories provided by the customer), and approval testing with the customer after iteration. Although it may seem like it, XP doesn’t involve the user in testing as much as other agile processes do. For instance, during the iteration of the system, the user is not testing and developing the product alongside the development team, but only does acceptance testing after internal testing, and provides the user stories for the developers’ test environment, as mentioned.

These prescribed engineering practices makes XP more prescriptive than Scrum, but at its foundation still lies an iterative process. For our project however, XP would not be the ideal choice, since the process model is best suited for projects which are heavily developer focused. Yet, XP is said to include most of Scrum, just as most successful Scrum teams most often include elements of the engineering practices mentioned in XP. It is therefore immensely popular for projects to adapt a mixture of tools (process models) to be most effective in delivering their product.

**The waterfall** **model**

The Waterfall model is a traditional, non-Agile approach to software development characterised by its sequential phases and limited adaptability to change. It is best suited for larger, sometimes government issued projects with well-defined, stable requirements and where there is little uncertainty (Sommerville 2016, 47). Due to the model’s nature, it is not a fitting model for a project like ours that require flexibility, customer collaboration and the ability to adapt to changing requirements.