

University College of Northern Denmark

IT-programme

AP Degree in Computer Science

Dmaj0919

SOLVR.ONLINE

SYSTEM DEVELOPMENT

Repository:

Maros Cuninka, Sebastian Holesz, Samuel Horacek, Martin Hotka, Ioan-Sebastian Voinea

Aalborg, 20. 12. 2020



University College of Northern Denmark

IT-programme

AP Degree in Computer Science

**Class:** Dmaj0919

**Title:** SOLVR.ONLINE – SYSTEM DEVELOPMENT

**Abstract:**

**Project participants:**

Maros Cuninka

Sebastian Holesz

Samuel Horacek

Martin Hotka

Ioan-Sebastian Voinea

**Supervisor**:

Nadeem Iftikhar

**Submission date:** 20. 12. 2020

Table of Contents

[**INTRODUCTION** 4](#_Toc58968267)

[**1.** **System Vision** 4](#_Toc58968268)

[**2.** **Business model canvas** 4](#_Toc58968269)

[**3.** **Prototyping (Personas)** 4](#_Toc58968270)

[**4.** **Mock-ups** 4](#_Toc58968271)

[**5.** **Methodology (Compare plan driven vs agile vs kanban)** 4](#_Toc58968272)

[**6.** **Choice of method** 4](#_Toc58968273)

[**7.** **Planning** 4](#_Toc58968274)

[**8.** **Quality assurance & Quality management** 4](#_Toc58968275)

[**8.1.** **Quality criteria and architecture** 4](#_Toc58968276)

[**9.** **Risk analysis** 5](#_Toc58968277)

[**10.** **Requirements definition** 6](#_Toc58968278)

[**10.1.** **Functional** 6](#_Toc58968279)

[**10.2.** **Non-functional** 6](#_Toc58968280)

[**11.** **Configuration management** 6](#_Toc58968281)

[**11.1.** **Version control** 6](#_Toc58968282)

[**12.** **Architecture?** 6](#_Toc58968283)

[**CONCLUSION** 6](#_Toc58968284)

[**REFERENCE LIST** 6](#_Toc58968285)

[**APPENDIX** 6](#_Toc58968286)

# **INTRODUCTION**

Text…

# **System Vision**

When we, as a group, were faced with a decision to come up with a unique idea that could change an industry and could be used for our project, we had a range of ideas. The technical requirements laid out by the curriculum had to be considered, as well as an applicability of the solution to the real word. Thus, we allowed ourselves some time to innovate and we narrowed the ideas just a top 3 from which we voted the winner. Most of the ideas were related to e-shops or customer-to-customer marketplaces with which we decided to go. None of has ever worked on a C2C project ever before, so we took it as a challenge.

Product vision is something that shapes the project itself – it describes the overall mission of the product. For us it means that each member has something to fall back on, something forms the end-goal of the entire process. This is our statement:

“The ultimate marketplace where your homework finds its solver”.

It has to be noted that before coming up with our vision, we focused on the products name. We felt that it needs to be striking and that its domain should be available for us to buy, which we did straight-away. After some changes we ended up with Solvr.online.

# **Business model canvas**

After we came up with an idea what product we want to develop we decided to find out if it is even possible and would make sense for us to even start with it. For this reason and for better understanding of the entire idea we created a business model canvas which helped us to visualize the core business concept.

As you can see in Appendix A – Business Model Canvas, on the right side we have our internal factors we considered as relevant for us – Key Partners, Activities and Resources. In the middle we have Value Proposition, which describes what values we want to deliver and on the right side there are external factors, such as Customers and Market in general.

The main goal of the business model is to plan how a business intends to make money and business model canvas makes it easier for stakeholders to understand it. As our project was not meant to be developed in a such way, we have not considered this to greater extent than what was described before.

# **Prototyping (Personas)**

# **Mock-ups**

# **Methodology (Compare plan driven vs agile vs kanban)**

In order to develop a system or a product, one has to look at the way the entire development process will look like. How will he ensure that the system will not have to be remade in the middle of the development process and ultimately how will he meet customer’s deadlines & expectations. There are two main approaches: a more traditional, plan-driven development & more dynamic, agile development. The choice often depends on the internal circumstances (such as budget, team size, time to market, security, and reliability of the product, which can be crucial for a government system, etc.).

It has to be noted that both methodologies have one common attribute – they are both iterative – meaning that the development process is split into even parts, each building on the previous with an option to change more dynamically. This concept is called “incremental development”.

Plan-driven development tries to eliminate all the uncertainty and risks as early as possible. Overall, the process is very predictive, as well as thoroughly documented, meaning that it is quite formal and does not leave much space to change requirements. The process is what matters the most. This development method is based around development stages, with each having a certain output at its end. The most prominent plan-driven development framework is Unified process (UP).

On the other hand, agile development is, as the name suggests, much more about agility and dynamicity. It tries not to predict as much, but rather adapt to change, because it assumes, we cannot get things right up front. In contrast to plan-driven, it is all about the value, not so much the process. The formality is also much lower, meaning the number of ceremonies is lowered to a minimum. What matters is teamwork, strong collaboration with customer and flexibility.

Agile has many frameworks which are used all over the world. The ones that we will focus on are Scrum, Kanban & Extreme programming (XP). Scrum puts more focus on its artifacts and ceremonies. Development team uses artifacts such as product backlog, with user stories to be implemented, sprint backlog with user stories for a specific sprint (iteration). Scrum also has three for-mal roles (scrum master, product owner, development team), whereas Kanban has none. Kanban also omits most of the Scrum ceremonies and focuses purely on visualizing work on its Kanban board.

XP is the only purely software development framework. It is described in its four values – communication, simplicity, courage and feedback and it has twelve principles. In short, it focuses on frequent releases, programming features only needed at each moment, coding in pairs and constant refactoring.

Looking back, we started of by following purely XP practices. It took us few days to get accustomed to their “extremeness”. After sprint 1, we chose to follow Scrum, whilst borrowing some of the attributes from XP, we thought will help our development process. In the end, the formal structure of the project was shaped by Scrum, but the coding itself by the coding principles of XP.

From our experience, if we are to compare unified process used in previous semesters com-pared to frameworks used in the project, the biggest difference could be felt at the start of the project. We were not spending considerable amounts of time on planning. We have assessed potential risks and worked up some crucial artifacts, but most of our time was spent on the actual development. Difference was also felt on daily meetings, called “daily stand-up” in Scrum terms, which were much briefer than previously. We discussed purely issues at hand and tried to minimize attempts at predicting future.

ADD SOME MORE EXAMPLES?

# **Choice of method**

# **Planning**

When we had finished with defining almost all user stories, we had to prioritize them, in order to start with the most important ones. During this process we followed Kniberg, who suggested using value (from 0 to 99 in our case) instead of priority, where the priority number one is the most crucial. By that, we gave our two core user stories the highest value 99 and 90 and other user stories we evaluated from 0 by tens (10, 20, 30...). When we have added a new user story, we could then easily place it in the middle of two different values, so we did not have user stories with the same value.

Before each sprint, on a sprint planning meeting, we picked the user stories with the highest value and the process of estimation started. We thought about tasks it will include and then we used the Fibonacci numbers to estimate them. We used a breakdown and calculation method when we ended up with a user story with value higher than 21, we split it into two different stories. It was very difficult for us to estimate the first sprint, therefore we eyeballed it and ended up with a sprint velocity of 26. On the sprint retrospective we agreed that we finished everything but did not have time to implement a new user story. On the other hand, we wanted to challenge ourselves, thus for next sprints we used an analogy planning method and planned the entire sprint with very similar, but slightly higher velocity (sprint 2 - 29, sprint 3 - 32, sprint 4 - 36). As you can see, we increased the velocity by ten between sprint 1 and sprint 4, which was caused by changing the overall architecture in sprint 4. Unfortunately, we did not play planning poker, because as we mentioned above, we got rid of epics and during estimation we all had similar opinions.

At the end of the project, we had also in mind what would be the next features we would like to implement. For next releases we would add:

* payments and other third-party APIs,
* forum page,
* a mobile app or make it optimized for mobile users

Chart, line chart

Description automatically generated

Figure 1 BurnUp Chart for Sprint 4

# **Quality assurance & Quality management**

## **Quality criteria and architecture**

Reflections on methods and their uses in practice

# **Risk analysis**

Test Risk analysis can be best described as a process of assessing the likelihood of an unfavorable events that can happen during the project and can influence its success and lead to failure.

We started with risk analysis at the beginning of the project, even before sprint 0, by identifying the risks. You can see the result in the table XXX in first column. After we found out what risks can affect us, we analyze the likelihood and the consequence of each one of them and by multiplying these two values we got the overall score (risk priority).

The top three risks are very connected to each other, because we were using completely new methodology, which is different from plan-driven, and our planning was divided to sprints. From beginning it was very difficult to estimate, therefore the high probability and impact. Even though, there is a corona pandemic, we set the impact to very low, as we already worked in a situation like this in our previous project. We think that this is even a good experience to the future as well, because we could try how is it to work only online, instead to face it our jobs for the first time. Last two risks are brought from last project, where we also encountered burnout at the end of sprint and not all of us are using Windows.

Table 1 Risk analysis

|  |  |  |  |
| --- | --- | --- | --- |
| **Risk** | **Probability** | **Impact** | **Score** |
| Wrong total estimates | 8 | 7 | 56 |
| New system architecture | 6 | 9 | 54 |
| Wrong sprint estimates | 8 | 4 | 32 |
| Fluctuant sprint velocity | 8 | 3 | 24 |
| Illnesses | 7 | 2 | 14 |
| Corona pandemic | 10 | 1 | 10 |
| New programming language | 1 | 9 | 9 |
| Problems with new SD framework (methodology) | 2 | 4 | 8 |
| Burnout | 2 | 4 | 8 |
| Different OS | 1 | 5 | 5 |

Next step was to plan how to minimize the effect of these risks as a part of qualitative risk analysis. We did not plan everything ahead, but we were monitoring it continuously during the entire project. We have only written down the strategy for the most crucial risks. For the new system architecture, the strategy was to find out what suits us the best and ask for a feedback. Nevertheless, we planned this, we end up with changing it three times, which had great impact. With the estimations, we decided that in the first sprint we will not overestimate ourselves, based on that we chose the starting sprint velocity and after that we knew if we could deal with higher or not. Monitoring crucial risks is shown in table XXX.

Table 2 Risk monitoring

|  |  |
| --- | --- |
| **Risk** | **Indicators** |
| Estimation | Good estimation at first sprint without changing the velocity by much |
| Architecture | Changed architecture 3 times |
| Illnesses | very low impact other members substituted |
| Different OS | dealt by booting to Windows and by changing .net framework to .net core |

# **Requirements definition**

## **Functional**

## **Non-functional**

# **Configuration management**

## **Version control**

# **Architecture?**

# **CONCLUSION**

# **REFERENCE LIST**

https://www.productplan.com/glossary/product-vision/

https://www.bmc.com/blogs/scrum-vs-kanban/

https://www.visual-paradigm.com/guide/software-development-process/agile-vs-waterfall/

https://en.wikipedia.org/wiki/Extreme\_programming

# **APPENDIX**

Appendix A – Business Model Canvas

