Team reflection week 5 - RCA CT-100

Customer Value and Scope

- the chosen scope of the application under development including the priority of features and for whom you are creating value

- the success criteria for the team in terms of what you want to achieve within the project (this can include the application, but also your learning outcomes, your teamwork, or your effort)

- **your user stories in terms of using a standard pattern, acceptance criteria, task breakdown and effort estimation and how this influenced the way you worked and created value**

We have been discussing and reworking user stories a lot so right now we feel like we have a good praxis and internal standardization to define user stories, acceptance criterias and tasks. What we want to get better with is the effort estimation, but right now we have chosen a good KPI to help us determine our work speed and therefore also help us with our effort estimation in the future.

- your acceptance tests, such as how they were performed, with whom, and which value they provided for you and the other stakeholders

- **the three KPIs you use for monitoring your progress and how you use them to improve your process**

We now have chosen the following KPI's:

- Amount of hours of effort put into the project each sprint.
- A burn up chart to track our progress.
- The overall happiness of our external client Linda with the product.

Design decisions and product structure

- how your design decisions (e.g., choice of APIs, architecture patterns, behaviour) support customer value

- which technical documentation you use and why (e.g. use cases, interaction diagrams, class diagrams, domain models or component diagrams, text documents)

We want to become better at documenting our code through comments and code reviews. This has been lacking almost entirely during the first two weeks of programming. We'll see if it cracks something later on…

We also plan on refactoring our code more often and not make as large efforts in output to have time to do this better.

- how you use and update your documentation throughout the sprints

- how you ensure code quality and enforce coding standards

Application of Scrum

- **the roles you have used within the team and their impact on your work**

Product Owner:

- Everyone kind of feels like a product owner and have been on every meeting with our stakeholder, it's just that our role of product owner is holding the meetings with the stakeholder and has the main contact with the stake holder if we have any questions during the sprint. Everyone prioritizes user stories together in the planning phase.

Scrum Master:

- The role of scrum master will be on a rotation so everyone gets the chance to be scrum master and learn from the role. The scrum master so far have been helping out with prioritizing what task should be done in which order and also helped with resources if someone got stuck on a task, resources in the form of helping with searching for information or just contributing with analyzing brain power.

Secretary:

- Anton has the role of secretary. The secretary documents the key points of all the planned meetings, mondays, wednesdays and fridays, including the meeting with the customer and course support. The notes are used in order to structure the meetings and being able to look back and see what has been said. The notes are especially useful for the reflections and sprint reviews.

- **the agile practices you have used and their impact on your work**

Sprint planning:

- The sprint planning is done on the first day of the sprint after the sprint review (Wednesdays). During sprint planning we select and make time estimation of the user stories that are supposed to be implemented during the sprint and if not present we write acceptance criterias for the user stories as well. After the user stories have been selected, they are broken down into tasks.

Daily standups:

- Every meeting begins with a daily standup from each member of the team. During the daily standup each member presents what they have been working on since the last meeting. This is usually done with a short demo and by showing the code real quick.

  Everyone in the team will be informed and up to date with what the others are doing and how far they've come with their tasks. This also helps everyone and motivates them to keep working and make sure they have something to show during our daily standups.

Sprint retrospective:

- Analyzing our first sprint we quickly learned that the way we approached problems and dealt with them were sub standard. We realized that our git workflow was far from optimized and quite messy. During the retrospective we discussed this and came up with solutions to our problems. Which we implemented in the second sprint. We also changed structure in Trello to make more concrete tasks.

Sprint review:

- We did our sprint review together with the presentation for the external stakeholder. This helps our work similar to the daily standups. We all get a sense of what we have done in the last week, what tasks and user stories are done but also making sure that the external stakeholder is happy with the result.
    - the sprint review and how it relates to your scope and customer value (Did you have a PO, if yes, who?, if no, how did you carry out the review? Did the review result in a re-prioritisation of user stories? How did the reviews relate to your DoD? Did the feedback change your way of working?)

We have let our external stakeholder prioritize user stories when she has preferences and discussed those with her. We have weekly meetings with her and write questions during the sprint if her input seems needed.

- **best practices for learning and using new tools and technologies (IDEs, version control, scrum boards etc.; do not only describe which tools you used but focus on how you developed the expertise to use them)**

We have been pair programming to make coding more fun and to complement each other. If someone is more confident regarding something that person either introduces the topic for the other ones during a meeting or helps out in smaller groups. Sometimes we have had short introductions to new tools and kinds of coding as well (like React).

In git we have decided to split our tasks into smaller branches to make reviewing code easier and to store every sprints effort in a separate branch. This structure was decided upon during our first sprint retrospective. Our expertise comes from using the tools in practice and asking if something goes wrong as well as reflecting on better alternatives.

Like coding in general we google a lot when needed.

- relation to literature and guest lectures (how do your reflections relate to what others have to say?)

Social Contract and Effort

- your social contract (Links to an external site.), i.e., the rules that define how you work together as a team, how it influenced your work, and how it evolved during the project (this means, of course, you should create one in the first week and continuously update it when the need arrives)
  There is a survey (Links to an external site.) you can use for evaluating how the team is perceiving the process and if it is used by several teams it will also help you to assess if your team is following a general pattern or not.

- **the time you have spent on the course and how it relates to what you delivered (so keep track of your hours so you can describe the current situation)**

We have an excel doc to calculate how much time we have spent each week and have related the time to user stories and sprints as well, to see what time they took and to make better effort estimations in the future. These first two weeks we have spent a relatively large amount of time (more than 20hr/week per person on average) on meetings and coding. This has resulted in a lot of output in code with much functionality already developed for the site as well as some general structure for the design. Later on we plan on spending more time refactoring code and commenting because we think we have time for that.