Final Report DAT 257

Group Tatooine

Customer Value and Scope

In the beginning we misunderstood how to formulate our end goal in terms of specifications. We made them few, thinking that we could expand the product during the course duration when we had time left. We basically made what we all thought to be our MVP our first scope. So we did later redefine it a bit, but got aware of this a few weeks in when we already made significant progress. Overall our sprints were oriented to provide user value, however in the beginning most of them were made as developer stories with the purpose to get our project structure set up.

As the project progressed, the success criteria for the team also changed. By working with scrum and in a team, all group members have improved in how we hold meetings and how we make sure that things are done from sprint to sprint. Efficiency increased as our understanding of scrum and a project structure grew and the time needed for meetings each week shrank. We agree that our success criteria have developed alongside these other skills to make sure that every group member knows what needs to be done as well as how to get them done.

User stories and task breakdown

Our user stories in the beginning were very vague and didn't really specify too much what needed to be accomplished and lacked a definition of done. For example we had stories that read "understand the API" which could be open to interpretation. As the project progressed we had user stories that followed the INVEST criteria with valid effort estimations that could be completed with the help of checklists in teams of 2-3 people. This really improved productivity and helped us move toward a finished product more efficiently.

Acceptance tests

Our testing was essentially by trying different values and by playing around with the parts we implemented. We usually did this in pairs and, after the sprint, we showed it to the rest of our team. After having our sprint reviews and reflections, we had a meeting with the stakeholder in order to show our progress. Even though we did not have a refined testing-protocol, this strategy has worked for our project.

KPIs

The KPIs established were:

- How satisfied is the team with meeting productivity
- Completion of user stories

The two KPIs above were not measured as directly as the one below but rather discussed in the group every weekly meeting.

- Stakeholder satisfaction rating these parameters from 1-10
    - Navigation
    - Information
    - Design
    - Functionality
    - Overall impression

Below is the weekly feedback from the stakeholder.

week 1 stakeholder:

- Navigation: 4
- Information: 4
- Design: 5
- Functionality: 3
- Overall impression: 4

week 2 stakeholder:

- Navigation: 7
- Information: 4
- Design: 6
- Functionality: 5
- Overall impression: 7

week 3 stakeholder:

- Navigation: 8
- Information: 7,5
- Design: 7
- Functionality: 7
- Overall impression: 8

Social Contract and Effort

Social Contract

The social contract used was written early in the project and only required minor updates. The final social contract looks as follows:

How we're going to work efficiently:

- Always try to split tasks into something that can be completed individually or in pairs of 2.
- Partly work together in order to increase communication between relevant parts of the project.
- If you are falling behind in a sprint, ask for help.
- Everyone is allowed to raise their thoughts.

Expected behavior of members:

- Being on time.
- One meeting every week where everyone attends.
- Do your assigned part in time.
- Ask for help when needed.
- Be clear with directions.
- Updating reflections after every meeting (added later).

The contract was not used rigorously or enforced, likely due to the fact that all of the team members knew each other well from before the project and had previous experience of working with each other.

Time spent on the course

The time spent on the course varied week-by-week. During the first half of the project more time was spent holding meetings and sprint- review/planning than coding itself. This was because a lot of decisions had to be made in early stages, about what to do, what technical tools to use, what APIs to use and overall structure. To meet these challenges, two weekly meetings were held up until planning-week 4 where the group agreed it seemed logical to start holding one slightly longer weekly meeting. At this point in time tasks were easy to generate and the project started to take on a clear technical structure. However this was also where we found our stakeholder and started meeting weekly with her instead, although these meetings were not as extensive as the group meetings and sprint planning.

As a rough estimation the time spent from each team member varied from 10-30 hours weekly including meetings, tasks, supervision sessions and all the other surrounding work.

Design decisions and product structure

Initially we found an API provided by Göteborgs stad that provided air quality measurements. We made use of these with Openlayers to provide pointers on a map where the measurements have been made. Early on in the project we thought that the provided locations were too few and began looking for other APIs to complement the one from Göteborgs stad. That's when we found OpenAQ. OpenAQ collects the measurements from hundreds of stations around the world, including the ones we got from our first choice. So then we decided to only use OpenAQ and expand our scope from Gothenburg to include all of Sweden. This was a great increase in value for our product owner.

Technical documentation

In our case, the need for technical documentation was low. One part of this is due to us having a work structure where we switched the teams around to allow all team members to learn all parts of the project. Furthermore, as we implemented the map library through a javascript-based framework called Vite, which is built in connection with Vue. Both of these give a predefined framework, based on the MVC model, that the user would have to adhere to. It works by having the .html as the views and .js as controller files. As we did the project in its entirety with APIs instead of a database, the controllers also handled the model in the form of importing API-data.

Another piece of documentation that was heavily used by the team is, of course, Git. However, our use of co-authoring was lacking even until the end. We discovered this feature quite late during the project, and was not used frequently even at this point. Thus, the amount of commits shown by certain team members are a lot less than what they should be.

General documentation

Since our program is quite small, the in-line-documentation follows that size as well. By rotating every week, our team understood each part without documentation. Therefore it was not a priority for the team in comparison to making our user tasks. On the other end, we did update our social contract and added further documentation such as KPIs throughout the project.

Ensuring code quality

As stated before, we tested the user stories when finishing them. After our meetings we had a merging-session where if we ran into issues, we solved them together. When making the user stories, we used the INVEST-criteria to the best of our abilities. This was a good way to both make independent and valuable user stories, but also to make them independent and expandable. When following this, we realized that when implementing new user stories and that it was way easier than in our previous projects.

Application of Scrum

The only concrete main role the group assigned was that of scrum master. This person had the responsibility of conducting meetings as well as making sure that the sprints were at a reasonable pace. Aside from this, the team had a rotation with the smaller groups working on each sprint. This was to make sure that every person had an understanding of all parts of the project. To assure that every user story could be finished, one person with at least some prior knowledge of the language was assigned to each user story. By doing this, groups of two or three could be formed with one essentially acting as a teacher for the other two. A few sprints in, every group member had at least some experience with every part of the project and groups could be smaller and more work could be done each week.

Agile practices used

At the very start of our project, we didn't utilize agile practices to their fullest as neither of us had any previous experience. As we progressed through the sprints, we added more and more features through good feedback at the supervisions. We started by having one meeting every week where we would cover team reflection, sprint review and setting of new user tasks. We set some epics for the team that would serve as guidelines for upcoming user stories. We added concise meeting structure to begin with, and then added larger user stories that we could split into smaller developer stories within the team. As we got a PO, we were able to further tweak our project from an outsiders' perspective, placing our focus in the hands of the customer. All in all, we worked towards increasing our communication and short-term productivity for every sprint.

Sprint review

In the beginning of our project until week 5, we did not have a PO. The PO is a friend of our group, studying mechanical engineering. During PO-meetings, the team showed the PO the recent changes and let her use the program as she liked. Afterwards, she was asked to rate design, navigation, information, functionality and overall impressions from 1-10. We used these reviews to make user stories and also in order to prioritize them. This gave us feedback that we might not have picked up

by ourselves. For example, our PO wanted a button to zoom out to see the entirety of Sweden because she thought it was a hassle to have to zoom out every time. After that PO-meeting, we added that as a user-story. Having a PO gave us another perspective and let us see how a non-programmer viewed our program.

Practices for learning

For our project we used several tools to make our final product. We chose to code in IntelliJ, something everyone is familiar with from previous experience during our education. Then we worked with a library for the map, an API for air quality in Gothenburg that was later swapped for an API with air quality that covered all of the world that we narrowed and used the air quality over Sweden. In order to keep track of our user stories we used Trello as our scrum board that was easy to navigate. To become familiar with all these tools we split up during the first week in teams so we could focus on understanding how we could use these tools together. After that we divided the user stories in a way that someone from the previous week would keep working with the same thing and another person could come in and learn from that person in question.

Using this method we ensured that everyone could work on every part of the project and learn from each other about how it worked. This became very useful since no one was left in the dark about certain parts of the projects and everyone understood everything in the end.

Thoughts for future projects and learning outcomes

As we all went into this project not knowing much of scrum or agile, all team members have gathered new and relevant experience with meeting structure and agile project structure. We also learned much about new programming languages as well as APIs. The team overall had no experience with HTML, CSS and TypeScript.

As of such, things to bring along for upcoming projects are many. Among these, prioritizing INVEST-criteria for more independent work-load and implementing a meeting structure early on for better project planning. Furthermore, having an external stakeholder was something the team valued as an important asset with regards to getting input that the team would not notice otherwise. In general, having a clear structure rather than just starting on a project is a definite learning outcome for all team members.