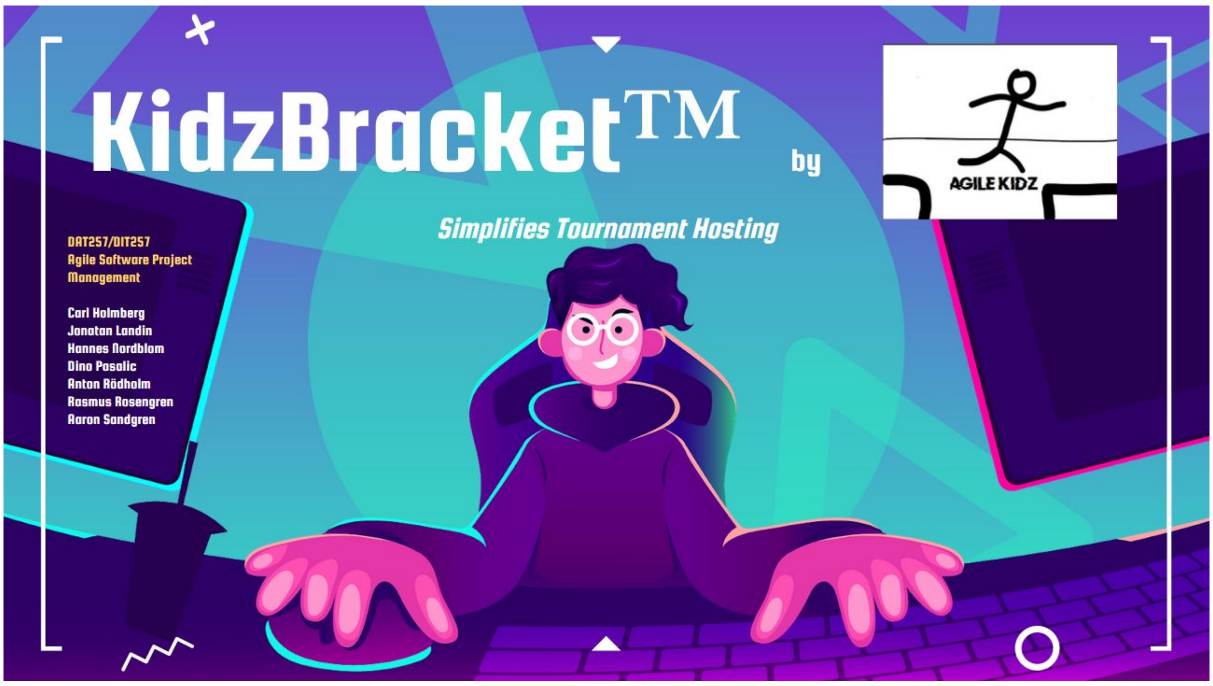
Final report



Background

We have created a website for the hosting of ESports tournaments. We found that a lot of websites offer services for helping hosting but no complete packages for admin contact, bracket generation, match score reporting and other features. Our ambition was to create this service.

Customer Value and Scope

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

What we did in this project

After we initially talked to the external part and got the idea and the MVP for the application, we wrote down our user stories in our backlog. We did not really have a priority order at this point. Because we felt that we had good knowledge about the use case of the app, we did not have a lot of contact with the external part at this stage.

The MVP did not change during the sprints; we added some bonus features that we felt would be reasonable to implement if there was time left. We also tried to keep in mind that we were developing the app for our external part, even though we sometimes forgot that and focused on a feature that would not really add value to the customer and that did not help us in reaching the MVP.

What we would do in the next project

The group agrees that we should focus more on feedback from the external part and have a deeper cooperation with them, especially during the backlog creation and the order of priority. That would help us to get a clearer view on what should be done first. We also reflected over the fact that the knowledge in the group made us less likely to contact the customer as we thought we knew what was needed in the app.

When it comes to the MVP and the scope of the project, we would not really change anything. We think that it worked out fine and that we did not undertake too big a task.

How we would do that

One way to get more feedback on the priority and what should be in the backlog would be to either involve the external part in the planning of the backlog or to make contact afterwards to get immediate feedback. Another part in getting a better cooperation with the external part is to use a product owner to a greater extent and to clarify what is expected and what responsibilities the product owner has, e.g. keep the backlog up to date based on the external parts feedback. This will depend on whether the external part has an interest in being more involved or not.

We would also try to acknowledge that just because we have a good domain knowledge does not mean that we have the same priorities and thoughts as the external part.

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)

What we did in this project

As a team, we were on the same level of ambition. We all wanted to see the working product MVP at the end of the course and be able to present it. Some of us play esport games and host tournaments while some of us just enjoy the idea. Our success criteria matched well with each other and it made the project enjoyable for us all. Our goal was to learn agile techniques to be able to reason about the agile process with experience and implement it in other situations such as work or another course. We were very keen on there being a relaxed work feel where group members do not feel forced to work, but work because it is fun. Therefore, we did not have the ambition to create the best product in the world. We had a comfortable effort level.

What we would do in the next project

We had similar viewpoints and aspirations and therefore cannot really think of anything we could improve. There would be a different situation if everyone's expectations and ambitions did not match up as ours did.

How we would one achieve what we did

A good first step is luck. If your group just matches perfectly like ours. Another important thing we did is to talk about how you want your group to work and what ambitions and try to agree on an ambition and effort level everyone is comfortable with.

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

What we did in this project

A standard pattern we used was user stories. The user story format has not been changed much since the start of the project. We previously had a part where we put the user story into a situation by writing a scenario in the form: “Scenario: Given that I am a tournament host, when the teams are ready, then I want the bracket to be generated”. The scenario was specified in the acceptance criteria instead to not duplicate information. Acceptance criteria remained the same throughout the project. We were unsure of how to write these at first but learned to write them from an external part or customer perspective. The acceptance criteria should be a guide on how to present the feature to the customer.

We had trouble with getting the size of the tasks to the correct level. In the beginning we made the tasks too small so that the overhead was too large and the solution to the task was to add one line of code. We also had a sprint where the user stories were too large and were difficult to manage. We later arrived at a process through reflection where we managed to correctly size user stories and tasks. We also started writing a detailed to do list on every task card with technical details so that the task was easier to manage. This improved our ability to also set the size of the task correctly as it gave us insight into how big the task was.

We tried to estimate the time needed for every task. This was inaccurate at first but became more accurate as time moved on. This process was not optimal as we didn’t quite know what to think of when we were estimating so this process could be improved.

What we would do in the next project

Our user stories and tasks worked well in the end. Especially writing todo:s on tasks helped us very much. The todo:s were valuable as they were formulated with the intent and detail that anyone in the group could get a big hint towards what they needed to do on the task. This made us work more efficiently and created opportunities where people of different skill levels could learn and improve.

We still would structure our user stories, epics and tasks in the same way if we started the project anew. We also liked the acceptance criteria we chose and would do them the same way.

We could improve our process of estimating. We would like it to be more efficient and precise when estimating the time of the tasks. One of the problems now is that you vote on tasks you have no clue on how to estimate if you are inexperienced. A more experienced programmer might know that a task for example might take more time because there is database stuff that needs to be done.

How we would do that

We would try to ask the less experienced in our group to evaluate if they understand the todo. This could make it more obvious where we needed to add more precise descriptions and where we needed to do that, and make the instructions less complicated when they need to.

We would also like a voluntary blank vote when estimating when you feel unsure about what the task contains. Everybody would still have the right to vote, but you are not forced to vote. It should be your own choice to submit a blank vote in good spirit.

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

What we did in this project

After several discussions, we came to the conclusion that the following three steps below was our process for acceptance test, meaning that if these 3 steps succeed the user story is complete. The steps are as follows:

1. Make sure a code review (according to our method) is accepted.

2. Make sure the acceptance criteria that belong to the specific user story are completed. These criterias should be written according to the SMART-principle.
3. Make sure Travis has run the code and there are no apparent errors.

We also make sure that the external part has viewed the functionality of the user story.

What we would do in the next project

For the next project we would like to add documentation as a step into the definition of done. We would also like to make more automated tests, integration tests and unit tests.

How we would do that

We would add a step in the documentation that says that you must update relevant documentation if the underlying functionality is changed. This is important as it makes the documentation remain current and useful. The automated tests would make it possible to better guarantee that the application works to our customer and helps the developers catch some bugs.

The three KPIs you use for monitoring your progress and how you use them to improve your process

What we did in this project

In this project we discussed and agreed on three KPIs: survey, velocity and process debt. We've used surveys to measure the group members wellbeing, stress level and experienced team support. During the project we did change our survey form to some extent to get a more representative form. The second KPI we used was the velocity, this we defined as the amount of story points we finished each sprint. We estimated the hours (story points) a task would take at the start of the sprint and when we finished a task we added the story points to our velocity count. We used a burndown chart to visualize the progress, but we had some troubles to get it to work properly. Lastly, we used what we called technical debt at first, but later changed to process debt after input from Jan-Philippe. Here we measured what must be finished from previous sprints before we can start with the new tasks.

We could see progress in the velocity during the project, mostly as a consequence of the group getting more efficient while coding and that more group members got the knowledge to code on their own. We also had some process debt from every sprint, but not an overwhelming amount at any point and we made sure to take care of it in the next sprint. The surveys helped us in keeping track of the group's wellbeing and if there seemed to be any dissatisfaction or too much stress during the sprints.

Although having the KPIs and using them to some extent, we felt that we didn't use them as much as we would have wanted or maybe should have.

What we would do in the next project

In an upcoming project we would make sure to have some easy measured KPIs, like the ones we had, but we would follow up on them to a bigger extent. Maybe make them more well-defined as well. We would discuss the KPI results from each sprint more in the group to have a chance to lift any problems we had and use it as a tool to improve the next sprint. Although, our 3 areas of measurement (members wellbeing, what we deliver and how the process is going) functioned well and would be a recommendation for next sprint.

How we would do that

Make the KPIs a standing item on the meeting to make sure it gets discussed and analyzed. Use a burndown chart to a greater extent to follow the progress during the sprints.

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.

What we did in this project

The first step into developing our social contract was to have a group discussion of what it was, its purpose as well as researching different aspects and areas that are relevant to have social rules about. That resulted in us finding a template which we revolved around while having the group discussion. One person was noting all the different aspects which were brought up and got the responsibility to write a draft, which was later agreed upon in another meeting.

As the project progressed, we seldom felt the need to update our social contract since we thought our process as a group operated well. Therefore we did not feel a need to discuss it on a weekly basis. However, some aspects were hard to pinpoint. For example, identifying consequences for coming late to a meeting was hard since we never met in person. Halfway through the project we found a good “punishment” for being late, which was donating money which we at the end of the project would use to eat and drink. This was not only a good punishment, but also improved our ability to be on time in general. Looking back, we formulated our social contract and its principles, but then relied mostly on common sense within the group rather than relying solely on the paragraphs of the contract. This resulted in the social contract being broken in aspects other than being late, but the common sense within the group did not find any punishment for this most of the time. Overall, we felt this process worked very well with very few conflicts emerging.

What we would do in the next project

Bringing our learnings to another project, a more developed contract as a starting point would be useful. We also believe we throughout the project could circle back to the contract more often than we did to not only remind people about the contract's principles but also to take time to reflect and make suggestions to improve it. However, we feel a weekly checkup is not necessary.

How we would to that

To further develop a social contract we need more research and discussions. Therefore we would set a meeting only discussing the social contract and write it as a group instead of pinpointing the responsibility to one individual. In this way all group members most likely feel more connected to the contract. For optimal results the research should be done individually so the team members come prepared with opinions and suggestions to the meeting, making the discussion more lively and constructive. Regarding updating the contract we would, other than encouraging to address problems as they occur, set a meeting halfway through the project to solely focus on the social contract and how it can be improved.

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)

What we did in this project:

During our sprints we logged the time spent on a task in Trello, by noting the time we collectively spent on the task on the card. Our goal was to get better and better at estimating the effort and conversely be able to find an optimal workload each week. This only includes working with specific programming tasks. The other assignments, for example meetings, have not been tracked. We have summarized the process in a velocity chart, seen in figure X, below.

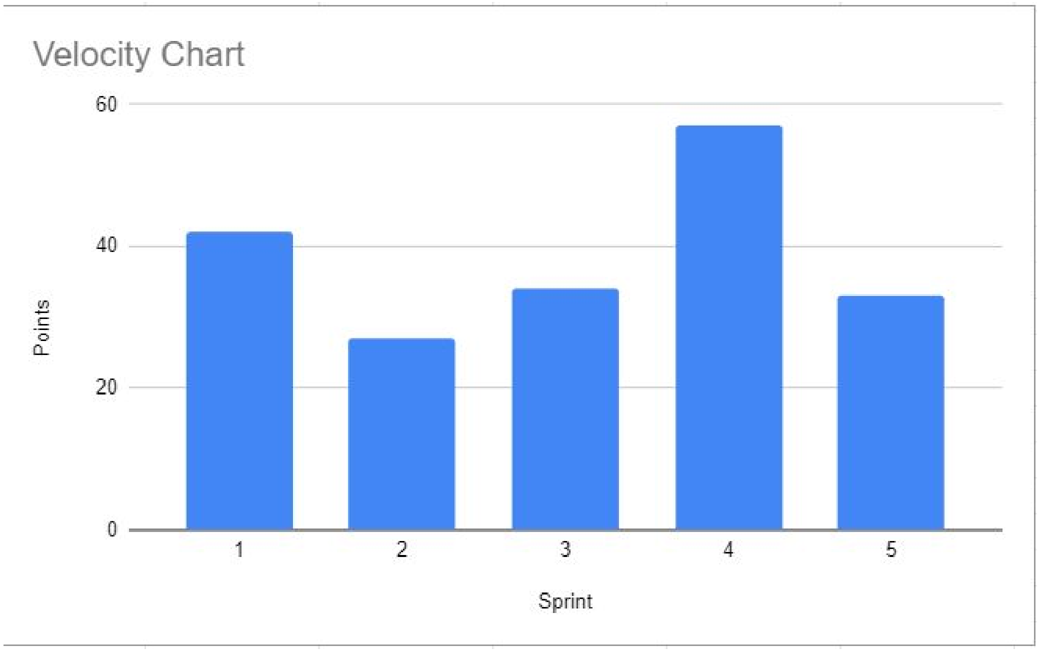


Figure X. Trello velocity chart

From the literature and the lectures, it was said that most projects implementing scrum overestimate what they can deliver in the first weeks, which was something we wanted to avoid. Therefore, we had this in mind during the first 3 sprints, which also can be seen in the charts. Then we reflected and realized we could do more in week 4, resulting in much more effort done, but from the survey of wellbeing we knew that it was also a more stressful week for all of us. However, what the figure does not show is our ability to manage through the technologies and understand how they are connected. As mentioned in Team Reflection, we were around halfway done with our MVP in sprint 3, including the setup time and learning curve of the technologies, which we thought was great progress. Sprint 4 and 5 were more focused on functionality and design since we had adequate knowledge about the connections in the application by then.

What we would do in the next project:

The key aspect to take into account is that you need to learn from your mistakes early on in the project. How else would you know your optimal effort for a week if you do not test your limits? With that said, we would spend more time trying to understand and reflect upon the week's effort and get a sense of how many points could be optimal each week in the long run. Since this was a relatively short period of SCRUM we believe that if we had 5 more weeks we would get more precise results of where the optimal points might be measured to. To be able to do this we would need to focus more on our ability to estimate assignments in the sense of time spent.

How we would to that:

Our concrete approach would be to broaden our workload the first week, to quicker find our pain threshold regarding effort. With that said, we would be very ambitious in our first sprint planning of what we could accomplish in the form of tasks and user stories. Then reflect around our results regarding 3 different questions:

- What was the most time consuming?
- What did we realize being harder than we thought?
- Are these matters a part of our learning curve, making them less time consuming in coming sprints?

From these continuous reflections I am positive we would find an optimal work effort each week.

Design decisions and product structure

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

What we did in this project

We have chosen to use PostgreSQL since it is widely used and has support for integration with other libraries. This is good because if the customer wants to continue development in the future, there is a good chance that those developers, if not us, are familiar with PostgreSQL. PostgreSQL is also a very reliable and tested database that allows for scalability if the customer increases the demands on the application in the future. Furthermore, every person in the team was aware and has used PostgreSQL, making the decision obvious since we have not received any requests from our external part.

Considering the back end, we have chosen to use Node together with GraphQL libraries. There was a discussion about using Rust, a more complicated language, but this idea was scrapped and we picked NodeJS since web development experience was limited within the team and using TypeScript for both the frontend and back end reduced the strain for newcomers at learning new languages. This increases the productivity for the customer and by using TypeScript we have utilized a common language that facilitates future development.

For the front end, we picked React since it is considered by many to be the best language for web frontend now and it was in TypeScript. Front end languages offer different pros and cons and the decision was purely based on what we thought would be the best supported in the future and what would be relatively easy to develop in.

We used the same language, TypeScript, in the front end and the back end. This makes it easy to learn both the back end and front end. It helps us create customer value by making it easier for those with less experience to get going faster. This also makes it easier to do vertical development since one does not need to change language when jumping from the front end to the backend.

We used GraphQL as the communication protocol between the back end and the front end since it gives a better developer experience and reduces the risk of bugs with code that undergoes rapid change.

We chose to use docker as a way to run our application. This helps the customer because it allows them to run it in the cloud or on a server. As it should work the same in development as in production, it also helps the development team.

We used git and GitHub for version control, which creates customer value by giving a history of the project.

Technical documentation was written, both for the internal team to understand the product, but also for the external part to understand how to use the product.

One thing that came very late in the project was deciding on styling on the project. We wrote our own components until the last week but decided that it would be easier to use a library instead. This should have been decided on earlier and been implemented from the start to avoid a big ending refactor.

What we would do in the next project :

It is important to understand what the project needs, therefore looking for options and picking the API’s and such so it suits the project at hand.

We should have discussed the technologies more in the group before we chose them. There might have been better alternatives, but we did not really discuss the decisions we made.

There should also have been a discussion about our choice of graphical components. Either we should maybe have added a greater emphasis on creating “good looking” components from the start or we should have used a library from the start. The final refactor was at least not desirable

How we would to that:

By doing some research before the project starts so we can see the different options we have and choose accordingly depending on what the project entails.

It would be better to discuss the technologies in the group because then everyone could have input on the choices.

There should also be an initial discussion about what technologies are needed. Maybe we are missing a library, like with the graphics.

Which technical documentation you use and why, and how is it used and updated? (e.g. use cases, interaction diagrams, class diagrams, domain models or component diagrams, text documents)

What we did in this project:

Leading into the first sprint, one of our more experienced programmers constructed a stack document in Github with the purpose to simplify the understanding of our projects system and the connection between different parts of it. This document contains an overview of the project's functionalities and how they relate to each other, our applications' different DevOps practices (e.g. Travis and GraphQL) and vital information about the structure of the backend, frontend and the database. The purpose of this was to shorten the time it takes to understand the project as a whole and the different connections between technologies, especially for more inexperienced programmers. Looking back at our sprint reviews, we can see that this proved helpful. In addition to this we created a mockup to give an intuitive experience on the flow of the applications as well as a use-case diagram to show different stakeholders’ relation to the functionalities in our MVP and to whom these features create value. These have not been changed over the course of the project since the need has not existed. Lastly, we constructed a design document which contains a summary of our features and technical documentation in the final version.

What we would do in the next project:

In a project with the same size, we would have the same way of constructing our use-case diagram. Given the size of the project, we felt we did not need to use the use-case diagram since we all had a good understanding of how the features correlated to the stakeholders. However, in bigger projects this might not be the case and the use-case diagram could then be of great importance, but for a project with the same size we would probably not use put time to form a well done use-case diagram. We would continue to construct a stack document since this worked perfectly well and functioned as a great overview. We believe our mockup should have more work put into it, but that refers to its purpose. The purpose this time was to show what flow on the website we had in mind. However, many mockups are used to impress the external part and communicate how we can create value to the external part. For this we need to involve the external part more than we did to understand the value we are creating and if they have specific design preferences. Another example of technical documentation that has been discussed is a UML diagram, but we did not really see its value if it covered the entire project. The diagram would just become convoluted and overly complex, so it would not be of any use. However, an UML diagram with the purpose of creating an overview by modeling different modules, for example describing the frontend by different screens, pinpointing its functionalities and structure would be more helpful. It could also help a new programmer understand the flow of the application and which modules depend on which.

How we would to that

To decide which technical documentations are suitable, information about the project to be done is needed. Then we can decide for example whether a UML diagram is the most suitable. For this Documentation which is not updated is not very useful, making continuous updating of it of high importance for projects. To make sure that the documentation stays updated, we would suggest adding a requirement in the definition of done that requires you to update the documentation where applicable. There are also cases where the code base does not prompt you to update some documentation, for example adding another library that needs to be added to the technical documentation but might not show up in an UML. These need to be addressed, whereafter we suggest a standing paragraph in the meeting agenda where similar issues can be discussed. Regarding the mockup, if the case is to impress and pitch the idea to the client, you need to plan the design and have opportunities to receive information on what the external stakeholder is looking for in the application.

How you ensure code quality and enforce coding standards

What we did in this project:

Regarding coding standards we started out with a style guide, that we didn’t change, that was enforced by different programs. We used ESlint and Prettier which are programs that automatically check your code and change it to follow a defined style guide, e.g. changing all “ to ‘. If the code did not follow this, Travis CI would not let us merge to master since it would use ESLint to check for these errors, forcing us to keep having good code standards. One big problem we ran into was that we did not clearly state what these standards were from the beginning. They were defined by a group member which had considerably more knowledge than others and therefore was not always clear to the rest of the group.

To ensure the code quality we used branches that were created for every feature that would need reviews to get merged. We defined the branch strategy already in the first sprint planning which really helped keeping it consistent throughout the project. We did not really create a definition of done from the start but instead it was only defined by a code review being done. We later defined a real definition of done which also included that Travis CI would build all the way and we added acceptance criteria that would be written according to the SMART criteria. This really helped evolve our branch strategy. Still, code reviewing became challenging for people since not everyone knew what to look for while reviewing. This was solved after a couple of weeks by defining a code review guide which included file structure, variable names, readability, code duplicates and documentation. We also decided that code reviewing was to be done in pairs. This meant that more people had a chance to ensure the quality and also that people could learn from It was important to give group members an objective way of reviewing code, both to make it a bit easier but also to make sure that we did not miss anything.

What we would do in the next project :

We would start by writing the documents for the style guide, code review guide and definition of done. It felt really fun just jumping into the work but we feel that the documents really help members to understand what to do. This leaves a whole lot more room to find improvements in our processes instead of just not knowing what you should do. More structure from the start feels like it would help launch the project more steadily with more members knowing what to do and how to do it. If you are uncertain you can at least find some confidence in well-defined processes.

How we would to that

We would dedicate a good amount of time at the start of the project to write these documents and make sure that all members take part in the discussion and have a real opportunity to read through the documents. It's also important to set up some automatic services like travis in the beginning of the project.

The code review guide would make code reviews more valuable and effective since it gives clear instructions on what you should look for. This would make sure that all the code has had a thorough look through of at least two people which would make sure that the code follows our standards and style guides.

We had automatic style testing that prohibited code to be merged if it doesn't pass the linting. This makes sure that the code at least follows our ESLint and prettier config. We thought this worked well and would use, and recommend others to do the same, in the next project.

Figur Z: Engagerade gruppmedlemmar under inlämningsdag

Application of Scrum

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

What we did in this project:

The only role we used since the beginning was product owner who was assigned the responsibility of communicating with the external part. While it worked well from our part, the external part was not very engaged as it was voluntary work for him. We did try to use a scrum master during one of the sprints, but it was loosely defined and therefore we came to the conclusion that it was unnecessary. Lastly, we decided to implement a chairman somewhere halfway through the project. The chairman's tasks were to make sure meetings went a bit more smoothly by making an agenda for each meeting so that they would not derail. This worked out great, as the efficiency of our meetings increased greatly.

What we would do in the next project :

Having a chairman to have more structured meetings from the beginning would make the entire workflow much smoother. It would also be ideal if the agenda is published at least a day before the actual meeting if one of the other members has any input on it. When it comes to the product owner, getting more feedback facilitates work on future tasks and shapes the project as you go. Since the scrum master did not have too much use for us in our project, we could have given other tasks to the scrum master to make it more relevant for our case. This would entail that the scrum master would have a bit more oversight and delegate tasks.

How we would to that

We should ask the right question when we talk to the external part and show our progress from our sprintlog. We should also show the external part our process so that the feedback process becomes simplified. Something else that should be done is to standardize the meeting process. It could also be the scrum master's task to check on deadlines and the social contract. Not all deadlines are strictly scrum related but you do not want the social contract to be broken consistently since then there may be something wrong with it.

The agile practices you have used and their impact on your work

What we did in this project

We followed a standard agile workflow which includes sprint planning, sprint retrospective and daily standups (although they were not daily). During the sprint planning, we split up user stories into tasks that members of the team could take on and we tried different strategies each sprint which got reviewed at the sprint retrospective

What we would do in the next project

Start with the same basis as in this project. While the entire point of working agile is to change it to suit the new groups needs and workflow, starting from a sound foundation and working yourself up. It’s important that every member in the group talks so that the agile process is tailored for the group as a whole. If possible its optimal if the standups are daily and include how you are feeling and status on what each member is working on. For the next project it’s also ideal to bring up past experience of working with agile as they may resonate with other members and make the group go “online” quicker.

How we would to that

Write a summary of the retrospective so that it's clear to know what the group needs to work on. It's not helpful if the team does not follow the conclusion of the retrospective for a sprint. it's helpful to put these focus points as a memo in the beginning of each sprint. While these are some good points, the thing with agile is that it's always different from group to group and its important for the group to find how they work best together

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?)

What we did in this project:

Aaron in our group was the Product Owner and were together with Hannes responsible for the sprint review with our external part Jonas. Jonaz, being in a situation where he thought the application was interesting and gave value, was also about to resign as Chairman of laggIT. Therefore, we did not meet the enthusiasm we wanted from his perspective, making the review quite tough and would have liked to receive more feedback. However, the feedback we got was analyzed and each week led to some reprioritization of user stories. Regarding the scope we had a MVP that was followed and the scope did not adjust much from it. Since we had this restrained feedbackflow, it was hard to relate them to our definition of done and therefore change our work process. This could be because of a lack of structure as every sprint review happened very spontaneously.

What we would do in the next project:

As discussed, a problem occurred with the limited feedback we were given, but that does not mean we do not play a role in this. We would try to find ways where we could make our external part more involved and make him/her understand that the more feedback we receive the more adjusted the application will be to his/her needs. At the same time we want to make it as easy as possible for that person to communicate and remove all communication barriers.

How we would to that:

There are a lot of ways to communicate better, and we have pinpointed what we could do from our side the next time. Firstly, we recommend a meeting discussing how the reviews should be done, and where both parts share their opinions of how and when the meeting should take place. The purpose is to set a ground and relationship for optimal reviews. Furthermore, the feedback received is strictly dependent on our process but most importantly our questions, which definitely is an area where we could improve. An open separate discussion on priorities would also be recommended so the work is collectively focused on the external parts most important features.

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)

What we did in this project

The project introduced a lot of new tools and technologies that most of the group was not very familiar with. We used tools such as git for version control, GitHub for version control sync, Visual Studio Code for coding, Trello for the scrum board. The technologies used to develop the project include TypeScript, React, GraphQL, Prettier, ESLint, Travis. To become better at using these tools and technologies we did pair programming and pair code reviews. We also created a stack repository where Rasmus created a large document where he wrote information about all the technologies we use, how to use them and sources where you can find more information. This was helpful for the less experienced. We planned times where we could sit two and two, one more experienced and one less experienced to try to distribute knowledge.

What we would do in the next project

We would start planning times to sit together from the start. It would help us match up more experienced with less experienced to help the less experienced get started.

We would like to get the less experienced more independent.

It would have been good to have more verbal instructions and more discussions about the technologies we use. Everyone should be more aware why we made the choices we did.

How we would to that

We should let the inexperienced write the code with instructions from the experienced. This would help the inexperienced be more independent in the future which would aid our team in creating value. It would have been good to take the long start up time into account and started learning the technologies before the first sprint. A meeting where we make sure that everyone has their computers set up for development.

Creating an excel sheet where people add the times where they can work would be helpful in making the process more efficient.

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

What we did in this project

Most of our base principles were taken from the course material. The length of the sprints. The KPI was inspired from the lecture. One decision we made was to use user stories instead of using job stories. We discussed the different alternatives, but concluded that it would be best to use user stories because we could get more help and support. We tried to use a scrum master, as the teachers and literature suggested, but we found it kind of useless in our case. We also deviated from the lectures by working on weekends. We found that this suited our schedule and was no trouble, so we worked on weekends if we felt like it. The weeks and weekends floated together with working at home. One thing we took from the Husqvarna lectures was their concept of a “fikapaus”. We implemented this in our group as a meme-break with unclear start time and an unclear end time, usually 30 minutes, where we sat and memed. This greatly improved team morale and helped us through long meetings.

What we would do in the next project

The efficiency of meme breaks could be improved. Sometimes they take longer than necessary and lose their functionality.

We did not try job stories, so it could be interesting to try using them.

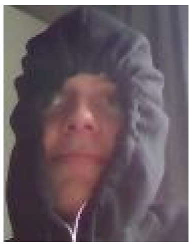
How we would to that

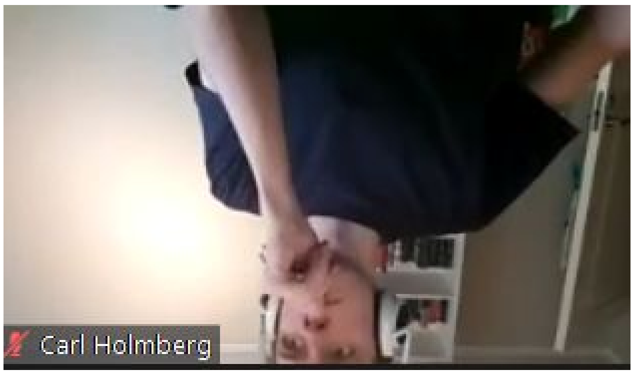
It would be better to set official end times for the meme breaks. Sometimes the meme break took too much time and the efficiency of the meetings could be improved if we set end times to the meme break.

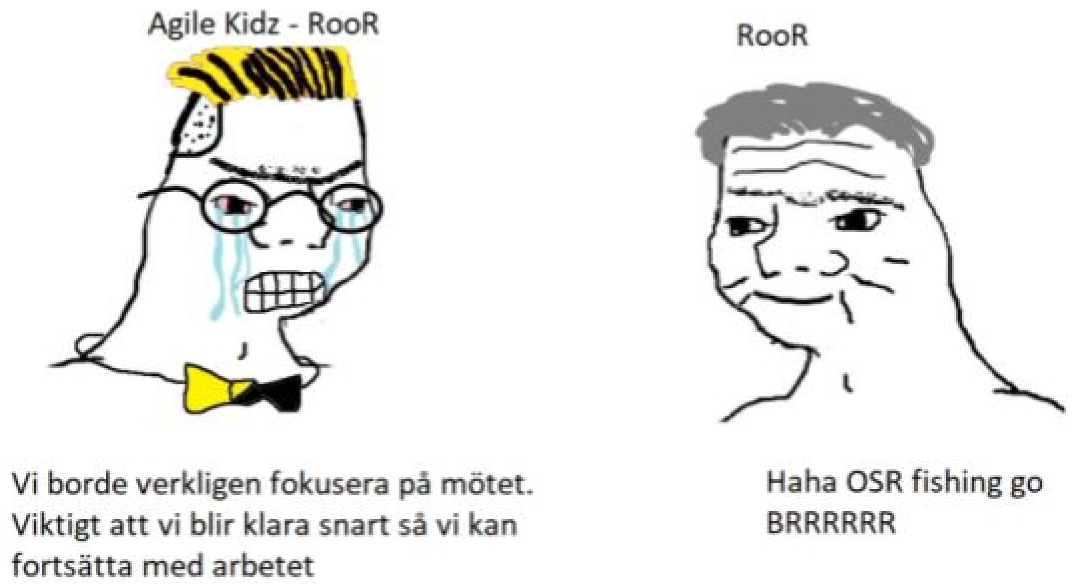
We could also have tried using job stories to increase the clarity in our stories. Job stories specify the reason why a feature should exist. They argue why they should exist which could help us determine if they actually produce value.

Other things would be improved by continuation of the agile workflow.

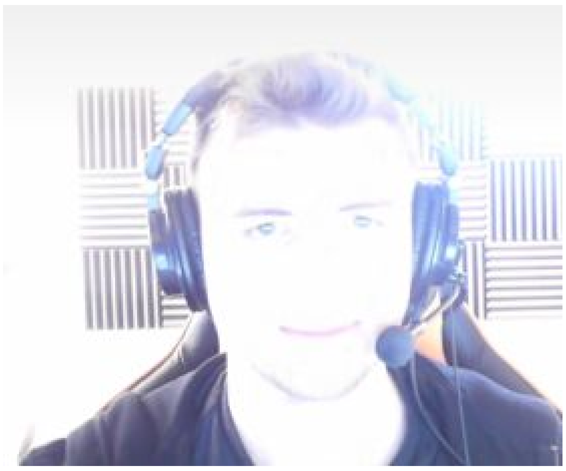
Agila Ungar ut!



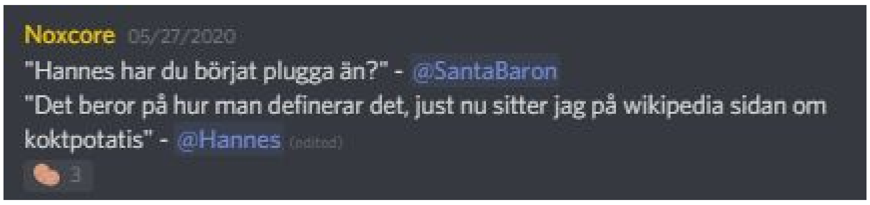

Carl Holmberg


Agile Kidz - RooR
RooR
Vi borde verkligen fokusera på mötet. Viktigt att vi blir klara snart så vi kan fortsätta med arbetet
Haha OSR fishing go BRRRRRR

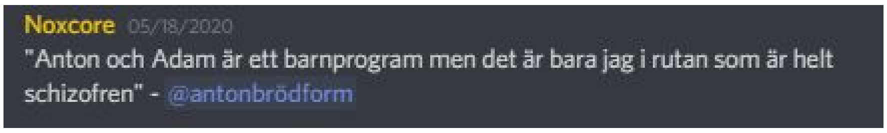
Jan-Philipp Steghöfer



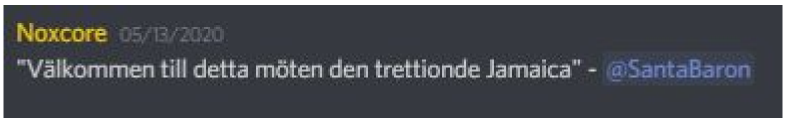
THOMAS & FRIENDS
No 1
THOMAS
THE TANK ENGINE



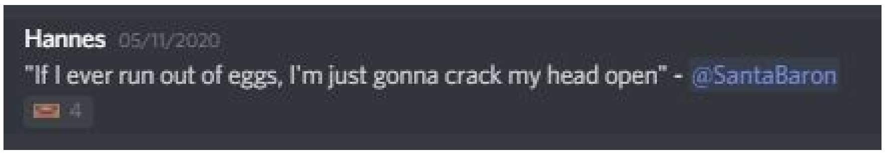
Noxcore 05/27/2020
"Hannes har du börjat plugga än?" - @SantaBaron
"Det beror på hur man definerar det, just nu sitter jag på wikipedia sidan om koktpotatis" - @Hannes (edited)
🥔 3



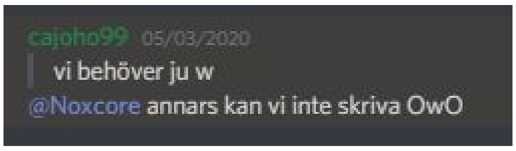
Noxcore 05/18/2020
"Anton och Adam är ett barnprogram men det är bara jag i rutan som är helt schizofren" - @antonbrödform



Noxcore 05/13/2020
"Välkommen till detta möten den trettionde Jamaica" - @SantaBaron

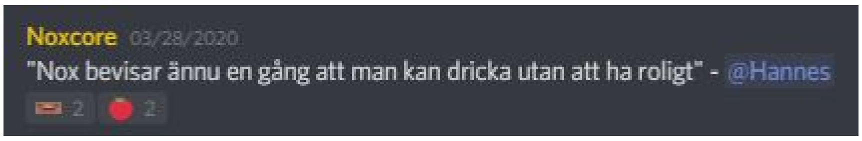


Hannes 05/11/2020
"If I ever run out of eggs, I'm just gonna crack my head open" - @SantaBaron
✉ 4

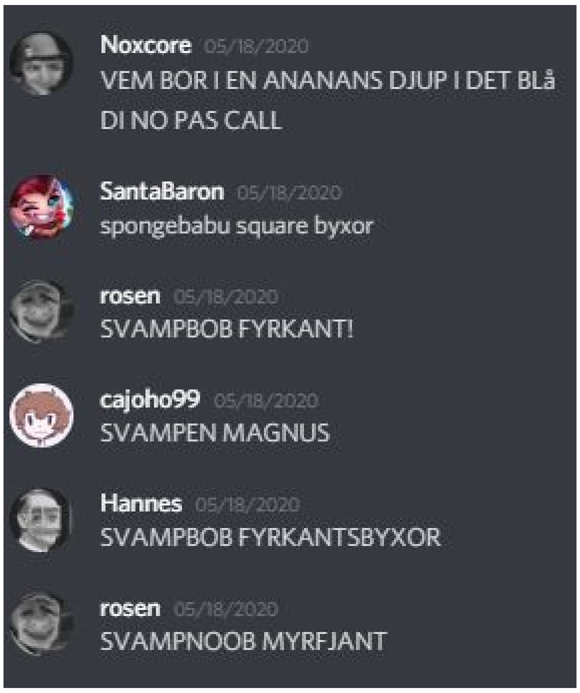


cajoho99 05/03/2020
> vi behöver ju w

@Noxcore annars kan vi inte skriva OwO



Noxcore 03/28/2020
"Nox bevisar ännu en gång att man kan dricka utan att ha roligt" - @Hannes
2 2



Noxcore 05/18/2020
VEM BOR I EN ANANANS DJUP I DET BLå
DI NO PAS CALL

SantaBaron 05/18/2020
spongebabu square byxor

rosen 05/18/2020
SVAMPBOB FYRKANT!

cajoho99 05/18/2020
SVAMPEN MAGNUS

Hannes 05/18/2020
SVAMPBOB FYRKANTSBYXOR

rosen 05/18/2020
SVAMPNOOB MYRFJANT

□□□□□□□□□□□□□□□