DAT255 - Software engineering project
2016-10-27

Reflection Report

Team Tim and the process of creating "Stava"

Marcus Eliasson Sjöstedt
Emil Hukic
Simon Moos
Marcus Pettersson Johnsson
Gustav Svensson
Erik Sänne
Samuel Utbult

Table of Contents

1. Application of scrum

In the following section the Scrum implementation/application for the team is discussed and analyzed.

1.1 Roles, teamwork and social contract (relates to D1B)

Excluding the scrum master, who was also part of the development team, no specific roles were assigned to group members. The work tasks were divided into groups of two to four persons, including the scrum master. The choice to not assign people to specific roles was done mostly because people did not have earlier expertise, and also to endorse individual interests in specific areas/tasks.

In order to make sure all group members shared the same common baseline work ethics, a social contract (see *Deliverable 1B*) was constructed. This made everyone know each other's expectations and work towards the same goal. As no conflicts arose, the assumption is that there were no problems with the social contract.

1.2 Used practices

1.2.1 Pair programming

Every sprint the team was divided into smaller groups, each working on different tasks. Since many tasks did not involve a huge amount of code, but rather a lot of thinking and planning, pair programming was a useful practice. There are several other reasons for why pair programming is good, however one of the most obvious ones is the fact that people see things in different ways which helps when it comes to solving problems.

1.2.2 Meetings and teamwork

Due to somewhat inconsistent work hours, the whole team did not utilize daily standup meetings. However, discussions between the task group members happened on a daily basis face to face, through text messages and at times through Skype meetings. One of the major reasons some could not communicate in person was due to the fact that they lived quite far away from our main work location. It might have been better if the Skype meetings were documented but alas they were not, this was an oversight.

The team also strived to have at least 2 to 3 working sessions (beyond sprint review and sprint retrospective every Monday and Wednesday) together with the whole group. At those occasions, every task group could ask questions to the other groups.

1.2.3 Version handling and issue tracking

The *Feature Branch Workflow* was used on Git (and GitHub) to organize and manage seven people working on a somewhat small codebase. In essence this means that there are at least 2 active branches at any time: *master*, which contains stable and viable products, and *development*, which is the possibly unstable working branch (named “feature” for legacy reasons). Code can be pushed to the development branch at any time but features/user stories should be done in separate branches which are then merged into the development branch. At the end of sprints, in preparation for the next sprint review, the development branch was merged into master. This way master always contained a stable and working prototype that could be demonstrated/reviewed/released and iteratively got more and more feature complete. This is very aligned with the Scrum methodology and worked out good for the team and the development of this application.

When working on features some members of the team used the built-in GitHub issue tracking system, while other members did not. Any differences in methodology in a team of this size makes teamwork more difficult and the team agrees that the usage of the issue tracking system should have been more uniform. The extent to which available tools (GitHub tools, Trello tools, etc.) were used was not discussed very much but should have been discussed more rigorously for better teamwork.

1.2.4 Backlog and user stories

Trello was used to organize the backlog and scrum methodology as depicted in *fig. 1*.

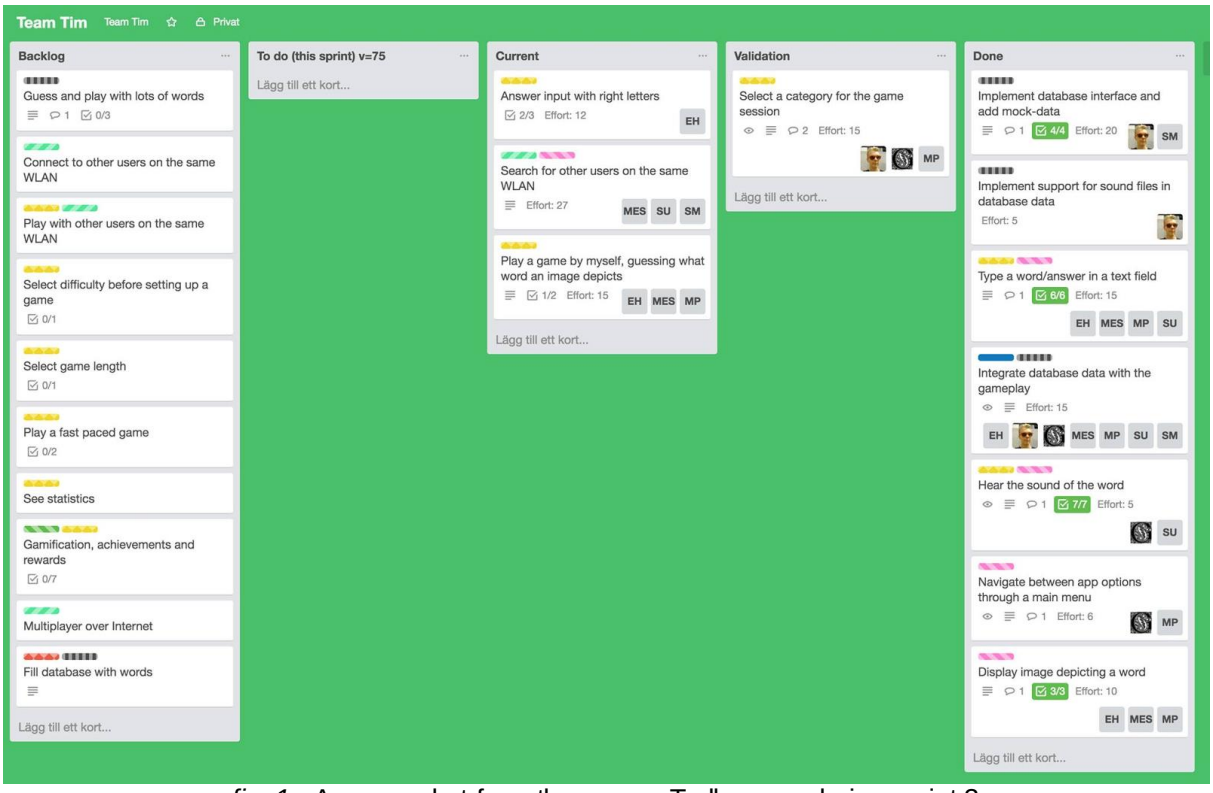


fig. 1 - A screenshot from the groups Trello page during sprint 3.

Working with vertical slices was a challenge. Some of the “tasks”, especially the early ones, were like user stories, more or less on the form “As a user I want to … ”. Later on however, as more features were added, they were more like implementation tasks for the developers. There was still always a strive that each new feature should have a connection to, or be represented somehow in, the GUI, but there was not always that much effort put into making it visually appealing or sometimes even complete. As a result of this, the sprint reviews did not always provide the feedback that might have been needed. More extensive reflection in the “Reflection on the sprint reviews” part.

1.3 Time distribution (person / role / tasks etc.)

Every monday after evaluating effort for each task, each group member got to “sign up” for a task they wanted to work with. Thereafter the tasks were evened out to make everyones load roughly equal. Members more experienced in an area related to one task were put together with inexperienced members on a task so that the task could be completed with efficiency and give all members experience. By doing this, the more experienced members could help the less experienced members in the group if they got stuck.

1.4 Effort and velocity and task breakdown

Initially the group decided that one velocity point would represent one man-hour on the project. Since the group consists of seven members and it was estimated that the group would work approximately 15 hours a week on the app, the velocity was set to 100 (7*15 = 105, but 100 is a nicer number). Later on the velocity had to be lowered to a more suitable value of 75. This was done after the first sprint retrospective where it became obvious that the team velocity was too high.

Effort points were assigned to the highest priority tasks *before each sprint*. It was done this way (in contrast to assigning all points before starting any work on the project) since it was not known at project start how many user stories could be completed in the limited time. The time taken to evaluate/assign effort points to user stories that didn't end up being started would be wasted time, so by doing it this way the time wasted could be kept quite low. The effort points were assigned through "Planning Poker" and was moderated by the scrum master.

Throughout the project there were no ways of getting any good feedback of the effort estimations. It was possible to look at the Trello board and try to analyze the flow of user stories, but it's not very easy to parse any usable data from it. Ideally a burndown chart should have been compiled after every sprint. If this was done it would be easy to see that group was slightly behind schedule after sprint 1 and 2 (see section *burndown chart & analysis*) and correct the velocity based on that. Unfortunately, the chart was not compiled until after the project was finished and therefore only worked as a "post mortem". One very noticeable consequence of this is the ineffective velocity change after sprint 1: it was changed but the lack of proper data made sure the change wasn't very significant, as can be seen comparing the slope of the burndown.

2. Reflection on the sprint retrospectives

Due to travel distance and lack of time, the team decided to have one big meeting each Monday that included both the sprint retrospective from the previous week and the sprint planning for the upcoming week, together with a general discussion on how to progress forward. This was a compromise that worked out quite well for the team in general, however it complicated the documentation since the meetings were too sprawling and there was "too" much information to process.

As the focus was mostly on what did not get finished, any problems that arose and caused complications with what did get finished were overlooked even if they might have been important. For example, if a task that was supposed to take 3 effort points took 15 effort points for some reason, it would have been beneficial if this was discussed with other members so they were made aware of eventual pitfalls during development.

A way of doing so would be to write down everything that kept the group from moving forward to have all problems ready in a document when it's time for the sprint retrospective to discuss the points.

Another thing that the group could have talked more about during the retrospectives is how the teams solved the different tasks so that the other members got a little insight on how things work. This way it's easier for another team if they need to change things.

3. Documentation of sprint retrospectives

At the beginning of each sprint a meeting was held where the previous sprint was discussed. A document containing the main points of discussion were created by the secretary.

Both a retrospective and discussion were written; both of these can be found in the directory 'documents' in the GitHub repository. During some meetings, a blackboard was used which was helpful when solving more complex problems, UI related issues, or work on tasks where mutual decisions were necessary. These drawings, together with comments have been documented by taking photos of the blackboard and then uploading them to Trello or saving them locally.

These documents are not a full summary of what was said since all members were active during these meetings. Therefore, the responsible member for writing these documents had trouble keeping up with all the things that were being said and at the same time being an active contributor to the discussion.

4. Reflection on the sprint reviews

There was one primary issue with the reviews throughout the project. It was not clear if the sprint reviews were supposed to be held with just the development team, or with the interaction designers as well. In the beginning, the development team saw itself as its own customer, since there would not be any recurring meetings with the end users (the newly arrived immigrants). As the project proceeded however, the meetings with the interaction designers were more and more seen as the actual reviews due to the fact that they were closer to both the end users and others involved, and therefore these are the meetings mentioned below.

The sprint reviews often landed in the middle of the sprint and not in the end, so the group usually ended up showing the last weeks sprint. This could have worked better if the review was at the end of the sprint so the members could get feedback on what they have done and if they need to change something. This would also had helped them plan the next sprint by getting feedback on how certain features should look and function. Since the design group mostly had time on Wednesday and the workshops were on Wednesdays the team could have changed so the sprint started Thursday instead of Monday, so the sprint finished with a review on Wednesdays. This also makes it so the members can use what they learned in the workshops in the new sprint.

Since the group usually didn't focus much on GUI design, the design group had a hard time giving feedback. With the "we will fix this later" attitude, the members usually got feedback on things they already planned on fixing. This is of course not a very effective way of using the reviews. If the group focused more on finishing everything completely, even GUI, in each task, they could get more feedback on the final design and not only on placeholder design.

5. Best practices for using new tools and technologies

The application is compatible with all new Android API versions (as of 2016) and down to API 15, which equates to Android 4.0, released in October 2011. This is purposefully chosen to include as many potential users as possible (~97% of all Android users according to Android Studio).

The group planned to write extensive unit tests so that evaluations of whether the application was working or not could be performed automatically by running a single command. This

kind of evaluation is mainly meant for debugging on a developing machine. However, there is type of service called *Continuous Integration* where these tests can be run by a web service as soon as anyone push a commit to a branch.

This kind of development was not used in a large scale, therefore only a few tests were written. It was planned to use the service CircleCI for continuous integration testing, but this service was set up in the end of the project as there was big problems with getting it to work, because this is an android application.

To summarize, continuous integration should have been set up earlier and more extensive unit tests should have been written.

6. Reflection on the relationship between prototype, process and stakeholder value

There are already applications with similar features available to language learners. However, Stava has potential for being a great tool in the target audience education. By developing this application and merge its content with the material the target audiences use every day in their school it would be a tool for them that they could use after class to rehearse.

The plan was all along to get a copy of the book “Hitta rätt”, which is the material newly arrived immigrants get and use when they start learning Swedish. In which there is content in form of categories and useful words. Unfortunately, the design team never got a copy of this and therefore this was never implemented.

7. Relation of your process to literature and guest lectures

It was very inspiring to hear about the practices used in large companies that work with software development. Reflections of how they relate to the project are listed below.

7.1 Daniel Sjölie, Interaction Design and Technologies Masters Programme (AR & HCI)

Sjölie talked about Human Computer Interaction and how important this is in modern software development. In this project, feedback from users in the target audience played a heavy part in deciding the applications features.

Sjölie pointed out what to think about when developing software for mobile devices. This knowledge was already known to the team, since all members had experience from an extensive design course. This knowledge was applied, although not because of Sjölie’s lecture.

7.2 Maria Carlsson, VOLVO Cars Corporation

During the Volvo lecture, Maria Carlsson talked about the difficulties for large companies to adapt to a modern, agile development practice. Since Team Tim is a very small development team it was fairly easy to pick up on the agile method.

She also talked about continuous integration, which is a type of automatic testing executed upon deploys. This was an inspiration to why continuous integration was implemented in this project.

7.3 Michael Öhman, Spotify AB

Michael Öhman talked about how Spotify divide their company in different teams called tribes and squads (where squads are the smallest units). Since these teams has the largest insight in the development process, Spotify believes in letting them make as much of the decisions as possible. This makes the development process much more efficient.

Team Tim is quite a small team, and much like the squads at Spotify, the team was free to make any decision by them self. This was probably a crucial factor for getting the product to the state in which it is today, considering the short time-span.

Spotify uses a practice called A/B testing, where different versions of the app is delivered to the customers. These versions have minor differences in functionality. Spotify then collects metric on how their users reacts to these minor changes. These metrics are then compiled to show results of which version was the most well received. Spotify can then decide which version should be shipped out to all customers.

Doing this kind of testing requires the developers to implement a big metric collection system and creating algorithms that can give them useful results. This type of testing also requires a large user base (probably at least a thousand active users) that is using the application over a long time period for the results to be statistically accurate.

Team Tim neither has a large user base or a long development time, which would make A/B testing unuseful. Implementing a metric collection system would also extend the development time and leave less time for other functionality that would be more useful.

7.4 Joel Rozada, The Techno Creatives

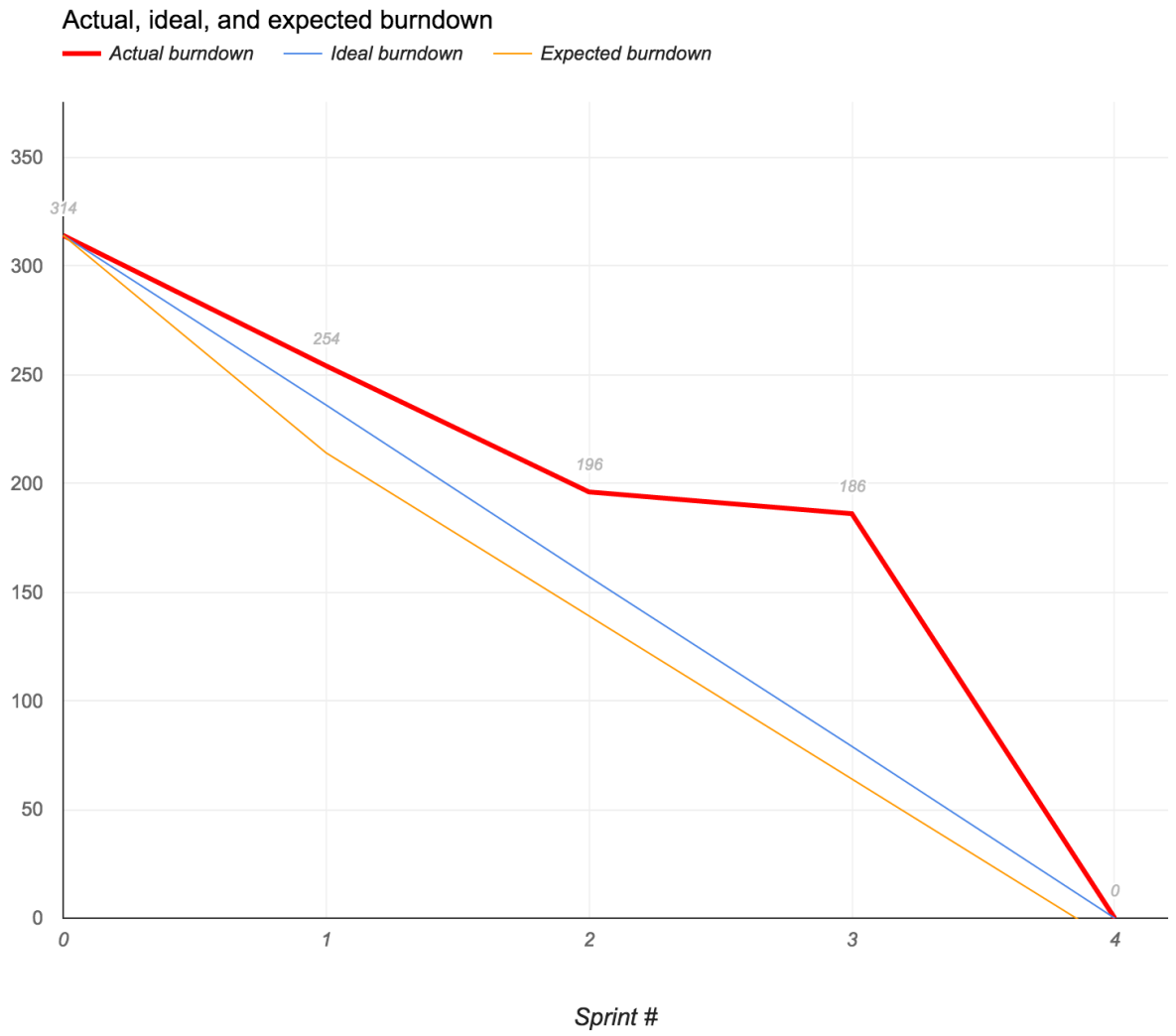
Joel Rozada mention how the digitalization of the world makes everything more accessible. This doesn't relate to the development of the app but more to the app itself. You don't need to have books with you to be able to study, you just need your smartphone, which makes everything easier.

8. Evaluation of D1A and D2

After completing the LEGO exercise the team drew a number of conclusions regarding projects in a professional environment related to communication with customers and their expectations. The conclusions were also about time estimations and the difference of a hobby project and a professional project. The group had these conclusions in mind when working on this project and it gave an advantage to the group that they otherwise would not have.

Since the half-time review little has changed related to the work process. In D2 it was discussed the the groups/pairs working on user stories should be mixed up between sprints. This turned out to be difficult to continue through the last two sprints since there were a few very large tasks that all had to do with the same things (namely multiplayer). What ended up being done was essentially that one half of the team worked on the multiplayer-related issues for most of sprint 3 and 4, while the rest worked on varying tasks.

9. Burndown chart & analysis



The diagram above displays the burndown for this project. This only includes user stories that were moved past the backlog (i.e. to "to-do" or further). The accumulated effort points for these user stories add up to 314. User stories still in the backlog after sprint 4 are considered cut.

The actual burndown (red) displays the burndown achieved. For the first sprint the group had the velocity 100, which were adjusted to 75 for the following sprints. The expected burndown (yellow) is based on these estimates and shows what could have expected from the estimates if they were accomplished. The ideal burndown (blue) is a straight line from 314 to 0 and shows the optimal route to the finished prototype. The ideal burndown is roughly equivalent to a constant velocity of 80.

The diagram shows that the expected burndown is very similar to the ideal burndown. The actual team velocity through sprint 1 and 2 is somewhat too slow, but not by much. The main outlier is sprint 3 in which the group only burned off 10 effort points. The main problem here was that some team members worked on some big user stories that took long time, much effort, and multiple sprints to complete. Ideally they should have been split up into even smaller parts, however, that would not have been trivial to do. Partly because of the character of the issues, and partly because the user stories were not expected to be as difficult and problematic as they were. In hindsight more time could have been put into

research and planning of the implementation details. This way the team would have had more knowledge going into it and could have done better effort estimations.

Since sprint 4 was the last sprint all remaining user stories had to be completed: crunch time, so to speak. Note that sprint 4 was slightly longer than the previous ones since the deadline was on a Wednesday (and a sprint usually lasted from Monday to Friday). Therefore, the slope should realistically be slightly less steep between sprint 3 and 4 than what is shown.

10. Summary of desired improvements and changes

The following text is just a summary of the reflections (in terms of the process) already stated in this report.

- The Interaction designers should have been treated as a more valuable resource, and the meetings with them as sprint reviews.
- The sprints should have been based around these meetings, that is: sprint planning on Thursdays, and retrospectives and reviews on Wednesdays.
- All team members should have used a more consistent model for version handling and issue tracking. Since it was decided to use Trello with user stories, there was no need for the GitHub issue system that was sometimes used.
- More focus should have been placed on making vertical user stories, focusing more on finalizing the GUI for each story. This would have resulted in better feedback from the meetings with the interaction designers and better result from the final workshop.
- Burndown-charts and better analysis of the finished work each sprint would have resulted in better estimations.
- Overall more detailed retrospectives, covering more than just the delayed tasks each sprint.
- For the purpose of more detailed documentation, the Monday meetings should have been separated into two different meetings.
- Test driven development should have started earlier.