Team Reflection

Week 41

Team members: *Nur Hussein Abdulkader, Tom Bjurenlind, Daniel Cebe, Rickard Gyllensten, Roman Melnik & Christer Sonesson*

Customer Value and Scope

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

  The application is supposed to give points for the number of passengers in a car. The points can be accumulated in order to earn rewards such as coffee or snacks at gas stations. The idea is to encourage carpooling in order to reduce air pollution and traffic congestion. The priority of features: identify the number of people in the car, award points for the current ride, store/load points and show total, create a shop where rewards can be redeemed. Value created for: end consumers (drivers), Volvo Car Corporation, the society (less pollution, less congestion).

- the success criteria for the team in terms of what you want to achieve with your application

  A user-friendly and intuitive application that gets many active users who carpool often, thus earning a lot of points and redeeming a lot of rewards. A cleaner environment and less congested roads.

- your user stories in terms of using a standard pattern, acceptance criteria, task breakdown and effort estimation

  - As an eco-friendly driver, I want to be rewarded for driving with several people in the car and possibly for e.g. driving with low RPM where possible (high gears).
  - As a driver I want to see my awarded points on the board that shows the number of points awarded for the current trip and in total.
  - As a Volvo driver I want to be able to visit the store page quickly, since I want to spend my points for rewards, and view my inventory (what I have spent my points on).
  - As a team we want to learn about Android API since we need this information to to complete this project
  - As a team we need a version control system git repository so we can communicate and work with the tasks
  - As a developer I want to use the simulator signals in order to determine the number of people in the car.
  - As a driver I want to see the items I have bought in an inventory since it's easier to visualise the items.

Our general strategy for handling the above user stories is to attempt to estimate how much effort they require in terms of one Sprint (one week's task work) and break them down accordingly to smaller features, and these into even smaller tasks; which may then be individually completed during one or two week's work (approximately). Our acceptance criteria is simply that the user story is met by our implementation. If we run into some complication with this criteria in the future we'll have to try to compromise somehow.

- your acceptance tests, such as how they were performed and with whom
  Acceptance tests for:
  - Working on the coupon store: testing that the core functionality works as it's supposed to.
  - Getting simulator signals from the seat belt sensors: testing that the signals are read as they should, that the number of people in the car is calculated, used and displayed appropriately and that we receive signals indicating when the simulation is over (i.e. engine off).
  - Scoreboard for the current trip: visually verifying that the scoreboard shows the correct number of points for the current trip.
  - Scoreboard for the total points earned: visually verifying that the scoreboard shows the correct number of points for the current trip and that saving and loading the total points work.

The tests were performed firstly and foremost by whoever was responsible for the respective task, and secondly by the entire team when everyone merged the changes.

- the three KPIs you use for monitoring your progress and how you use them
  We have now developed most of the core features and our main KPIs are team velocity (taskwork done each week/Sprint), quality (e.g. bug-free app), usefulness (i.e. is the app solving anything) and lastly effort estimation. We want to keep a reasonable pace and complete the tasks assigned for the current Sprint. The most fundamental way we measure our work is by how many tasks and/or how much on each task we complete each week (in comparison to all core tasks we had in mind from the start). Next week we probably want to finish-off/polish the core tasks we've already started/"finished". To do this everyone in the team has to actively contribute to the deliveries in some way.

Social Contract and Effort

- your social contract, i.e., the rules that define how you work together as a team (this means, of course, you should create one in the first week)
  Our social contract still hasn't changed since formulated the previous weeks. We still strive to help each other out when there's a need as well as divide the work equally between all team members. We will try to follow this contract as far as possible. If we'd discover further ahead that something isn't working with respect to this contract, then we'll have to tackle that problem appropriately when (if) it does happen, and perhaps modify the contract.

- the time you have spent on the course (so keep track of your hours so you can describe the current situation)

  Like previous weeks we define the hours spent on the course as the hours spent strictly on the project (in group work plus mean individual work at home). With this definition we have probably spent around 6 hours this week, so in total 8 + 13 + 14 + 12 + 6 = 53 hours (as previous weeks' work was estimated to about 47 hours). Among this week's hours, most of it was spent as individual work (mean work of all team members). (The time for writing these reflections is excluded in this time estimate as per the usual.) We estimate that the next week's hours spent should be about the same; as we continue finishing off and/or integrating the remaining tasks. If more hours is demanded from us in order to finish our planned work for the next Sprint; then we will have to try to put in more hours collectively as a team.

Design decisions and product structure

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

  This week we once again continued on implementing the app; so we got further experience with the different APIs. We have only used the standard Android APIs and Android CAR APIs thus far still, and we have built on top of the foundation work we executed previous weeks. This week there were even more focus on good design decisions; some of the foundation work from previous weeks were updated to reflect this. Next week we continue to work along these lines, with good design decisions in mind. To do this we will have to continue having good design decisions in mind when pushing new features, and refactor existing work where appropriate.
- what you document and why, by using e.g. use cases, interaction diagrams, class diagrams, domain models or component diagrams, text documents etc.

  Similar to last week we have only documented our project by our Scrum planning-board (Trello), drawn images of the different app screens on paper sheets, written reflections in text documents, and commented our code in an intuitive and explaining way. We'd say this works well for us still; so we do not really have any plans of incorporating further documentation (such as UML diagrams) next week neither. If we find out later that we might need further documentation for whatever reasons, then we will have to come up with an appropriate solution when/if that happens to be the case.
- how you use and update your documentation throughout the sprints

  Same as the week before. We have been using Trello to set up tasks and user stories, and we continuously update this as we progress. Google drive is used to write all reports, which are then uploaded to github. Github is naturally also used to store the project and all code. This is a process that works well and one that we plan to continue using.

- how you ensure code quality, enforce coding standards, and application of scrum

  No major changes from last week. Each team-member is responsible for his own code, but cross-checking by other team-members is also used, so any deviations

from the team-standard can be caught. Github is used to store the code, and with that all changes can be easily reviewed, and we can pull back changes to the code if necessary. A lot of code is produced in group-sessions, where most of the team can get together and solve coding problems (and review code), leading to a higher coding standard. Scrum is enforced partly by the scrum-leader, and partly by the will of the team. Everyone makes an effort to learn and apply scrum to the development process. This works well at the moment and it is a process that we plan to continue using.

Application of Scrum

- the roles you have used within the team

  The roles for the team are the same as the previous week. We have a scrum master who negotiate with other teams and ask/communicate with the product owner if needed. And as before the remaining five team members are ordinary developers. Their job is to contribute something to the team in the form of develop the product by encoding and complete the tasks. The scrum master also helps to develop the product as mentioned last time, but he also make sure that everything is fine and if there is some questions to the product owner or general questions to other teams.

  Further we should continue to use our roles as they are and continue to develop and contribute with everything that is required to deliver an application. To get there every team member need to follow the roles. As a developer for example, namely to develop code and be involved in communications and to perform the work that is relevant, specific, for every week. And finally in the end deliver a product.

- the agile practices you have used for the current sprint

  This week we continued in much the same way as last week. Those that were available came to the monday meeting to plan things out, then tasks were distributed among the group-members. Some team-members got together on another occasion during the week and developed in a group, while others preferred to develop from home. This is the basic system we have set up, and we plan to use it for the next week as well, with a little more planning planned due to the project nearing its end, so we need to wrap things up.

- the sprint review (either in terms of outcome of the current week's exercise or meeting the product owner)

  Things are on track and this week was very much like other weeks since everything is pretty much planned out and the only thing we are doing now is trying to finish the project. Thus we don’t have much to add here, things are proceeding as planned and steady progress is made, no new features have been implemented and there’s

generally not much to add.

- best practices for using new tools and technologies (IDEs, version control, scrum boards etc.)

  This will echo what has been written many times this week, since there's nothing actually new to do (all we are doing now is coding the project) there's also not much to add here (that hasn't been written before). Things are just proceeding in the same way they were last week.

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

  Our reflections relate to what others have to say structurally in the sense that we use the A-B-A->B way of reflecting where we find it relevant and/or applicable (as prescribed); where A is where we are currently, B is where we want to be next week, and A->B explains how we might get from A to B. When it comes to what we actually say when we reflect in this way (the actual content), we have not really read any reflections of others (nor have there been any guest lectures this week), so we couldn't really say how it relates content-wise to what others have to say.