Post Mortem Report Grupp 23

Which processes and practices did you use in your project?

We used the iterative and incremental software development methodology Scrum to structure our workflow within the project. We used Scrum practises such as short stand up meetings also known as daily Scrum. During these meetings we discussed a lot of different issues, like what problems we had and if we had any, how we would solve them. We also divided up the work within the current sprint. We had a sprint length of one week although sometimes we exceeded this limit due to difficulties in estimating the time needed for a certain user story. After each daily Scrum we did a version of Scrum of Scrums that everyone attended. At the Scrum of Scrums we discussed what the different teams (consisting of one or two people) had done since the last meeting, what the teams would work on that day, and how the different responsibilities were going for each team. We also used pair-programming which increased the effectivity of the coding.

Approximately, how much time was spent (in total and by each group member) on the steps/activities involved as well as for the project as a whole?

Our time spent on the project varied greatly, the last four weeks of the project were much more productive code-wise compared to the first three weeks. The first week was mainly spent discussing what kind of application we wanted to create, as well as discussing how we would work as a group. The two following weeks very little due to some extensive problems regarding the use of Android Emulator and devices paired with Google Maps API keys and setup of the basic map project. These problems took a lot longer to solve than they should have due to inactivity and lack of interest from some parts of the group. During the early weeks we also had problems scheduling the time for working

together due to the fact that we're not all from the same section of Chalmers (four of the members were from Datateknik, Lindholmen while the remaining two are from IT, Johanneberg).
The last four weeks worked out a lot better since we got used to our differing schedules and the fact that some laborations that previously had been colliding with scheduled work no longer were an issue.

Approximating the time spent for each individual is difficult due to some members working more from home while others spent their time in school working together. An estimate of the contributed work from GitHub shows a great difference in time spent even though pair-programming has to be taken into account. Some members averaged approximately 25 hours per week the last four weeks while others barely contributed until the very last sprints.

As mentioned above, the first week was spent deciding on what to do for the project and the following two weeks was spent trying to set up the project with Google Maps on a device, setting up GitHub and Android SDK on all members computers and rough sketching of the system architecture. During this period a few members really contributed to get the project on its feet. The last four weeks were as mentioned much more productive and we tried to divide our work into four sprints. The first sprint consisted of structuring the projects main MVC architecture, finishing the model and enabling new components to be added easily. The second sprint was focused on being able to place markers on the map that were saved locally. The third sprint contained showing the messages of a messagepoint when pressing the corresponding marker. The last sprint was to connect the local version of our application to a database.

For each of the techniques and practices used (such as standup meetings, pair programming, TDD, etc.) in your project you should answer all the questions:

i. What was the advantage of this technique based on your experience in this assignment?
ii. What was the disadvantage of this technique based on your experience in this assignment?
iii. How efficient was the technique given the time it took to use?
iv. In which situations would you use this technique in a future project?
v. In which situations would you not use this technique in a future project?
vi. If you had the practice/technique in a part of the project and not the entire project, how was using it compared to not using it?

Stand up meetings

i) Having short stand up meetings helped us get to work faster and avoid getting stuck on non productive topics.

ii) Missing details that had to be brought up later during work, interrupting flow.

iii) Since we had very short stand up meetings it was very time efficient and gave an excellent overview of what had to be done.

iv) As a quick way to get all team members updated and ready for work.

v) We would not use it in the middle of working on specifics when details are important.

vi) When we were not using it the meetings could be very long and focus could be on non productive topics like non-essential features that were irrelevant for the current time.

Scrum of Scrums

i) It gave a good overview of what everyone had been doing, or not been doing since the last meeting. It was also a good way to divide the work effectively among the group.

ii) Some individuals may not give accurate information on how the their responsibilities have progressed. Prestige or insufficient knowledge may cause the person assigned to a task not to ask for help or falsely indicating that the task is going well.

iii) It was very time efficient since it's to the point about essential issues that needs to be discussed.

iv) To get a clear overview of what has been done and as an effective way to divide work within a project.

v) Almost all the time except if it's obvious that every group member knows exactly what they are doing and what they are expected to do until the next meeting.

vi) This method was used almost without exception during the course of the project.

Pair programming

i) Improved code quality due to having someone constantly watching and reflection over the code written. It was also easier to find mistakes and helping eachother solving problems as you write.
It was also great being able to discuss idéas as you wrote code, tossing idéas back and forth.

ii) It could cause problems if the people involved are not comfortable working with each other, for example not being able to take criticism and differing idéas. Hence cause a person being afraid to write code or comment on it.

iii) It saves a lot of time since errors are noticed faster and you come up with better solutions, not having to rewrite as much. It also saves time by eliminating distractions such as facebook or anything that is not productive. If you are coding alone it's easy to get distracted but if you have someone to discuss with it's easier to keep each other on topic.

iv) On occasions where the code written is non trivial and extra input is needed. Cases like this can include design decisions, writing code that depends on complex algorithms or any time you want to write better code.

v) When the task is very simple and very little thinking is involved. For example implementing a very simple class like a Point class with x,y coordinates. Even in this case it would not be bad to use pair-programming, just less needed.

vi) Different members used pair programming throughout the project while others prefered to code on

their own. Those who used it experienced it to be more fun, effective and an easier way to write better code.

What worked well in how you worked in this project?

Working together as a group at school was a lot more effective than working separately. Code written when not in the company of the group was much more prone to being rewritten. The reason being that it's much easier to communicate while sitting in the same room. Questions can be asked and answered quickly. Pair programming was also very successful for the members utilizing this practice.

What did not work well in how you worked in this project?

As mentioned above, working separately was less effective since misunderstandings caused a lot of code to be rewritten. Naturally people will have different opinions on how to solve problems. These could be resolved much more quickly when working together.

Reflections over non process specific decisions?

As first time Android developers, it wasn't very clear how to implement the MVC pattern. Because you start and manage intents, and you can't get a reference to the the activity object, sending messages between components are done differently from frameworks we had worked with previously. This made the observer pattern hard to implement when our view was an activity. We thought that it was logical to have the view be an activity since activities shows a window. After discovering problems and reading about Android and MVC we learned that the controller most oftenly was the activity. We refactored our code and made this change, even though it can seem weird that the controller shows a window, it's just an empty window, and the view is still the component that shows the google map and all of its graphical content.

How did you work together as a group in the project? What worked and not

in your interaction(s)?

Working together at school and pair-programming was definitely what worked out best during the project. What became an issue is that we did not do these things as much as we could have. This caused problems such as people working on the same issues, insufficient communication within the group and productiveness suffered due to this. The reasons behind the issue was that some members of the group prefered to work at home, different schedules and difficulties deciding when to meet. Another issue was lack of interest from some members throughout the project and the contribution was not even among the individuals. A major problem was that some members failed to take initiative solving problems and expected other members to solve them instead. When we actually was working together it was fairly productive and we managed to reach our vision of the application, albeit without some extra (non essential) features.

What would you do differently in a future but similar project?

We would set up responsibilities and expectations for all members earlier in the project so that each individual is aware of what he or she has to do for the project to succeed. We would also structure up the work better in the early stages so that everyone has a clear role within the project. Another thing we would do differently is to get started right away and not have one or two persons trying to set up the project while the rest do nothing.

Another important thing we would do is to schedule the project better. One example would be to have a few days per week working together at school where everyone has to be present during the entire scheduled time.

We should have utilized test driven development. Creating tests first makes sure that classes and

components have the intended interface towards other classes and components. Communicating the expectations of a component would have been easier since the tests are a common reference point. You wouldn't have to explain that class x should manipulate data and return y, since the tests (if written correctly) makes this obvious. This would have made the project easier and saved time due to less communication errors, and thus less code rewriting.