Reflection Document

Software Engineering Techniques

Scrum did work very well for us. Managing the project was really easy compared to the waterfall method we had used before. The fact that it doesn't rely as heavily on documentation allows for more time to be laid on coding which seems far more efficient for smaller projects like this one. The slightly more abstract documentation structure also seemed more relevant compared the strict patterns we were made to follow when using waterfall.

We did not manage to run many proper daily scrum meetings, but since most of the project members were communicating for hours basically every day, this was not a problem.

We found a neat tool called Pivotal Tracker (www.pivotaltracker.com) which we used as our scrum backlog. The tool made it really easy for us to handle the sprints and to split up the work as everyone could see exactly what the other team members were working on at any point in time. Assigning yourself to a task from the backlog as well as adding task like bugs or feature proposals could not have been easier.

Although we worked mainly individually from home we did try out pair programming a couple of times with quite satisfactory results. We found that working side by side often was a faster approach when solving problems scoping areas more than one member had been working on. A fresh pair of eyes can often spot a problem someone else is overlooking but since the project ran over 6 weeks we felt we needed all the time we could get and pair programing obviously cuts our forces in half. Looking back, it might have been a good idea to utilize this technique a bit more.

With time being a constant issue throughout the project we did not manage to try as many Software Engineering Techniques as we had liked to. We had numerous discussions on trying out test driven development, since this is a widely used technique in the programming industry. However, as implementing our most essential application functionality took far more time than we had expected, this was never done. We all agree that in a future project this is something we would like to put a lot of time and effort in. Testing overall is an area we know very little about.

As with all projects we have done so far, this one included, the planning stage and a lot of the documentation seem slightly excessive. This is not to say that it is unnecessary in any way but because of our lacking experience, decisions made in the planning stage has a tendency not to last. A lack of knowledge makes it very hard to in the early stages imagine the structure of a finished application. However we realize this is a natural - maybe even childish - reaction to something we are yet to realize the real purpose behind, time again this makes the planning process a frustrating task.

Team

As stated earlier we did the majority of the work individually from home. Since we all had overlapping schedules, setting up meetings for us to code together seemed an impossible task, this was a natural decision. This had both positive and negative consequences. Working from home, where you have a better setup (bigger screen, proper keyboard / mouse and so on) seems like a good approach as long as you can focus on your work. Despite the obvious downside of communication issues, this worked quite well for us. With Skype, instant messaging, screen sharing and other neat tools and features available over the web communication was not a big problem.

Since Scrum dictates a project team where everybody is capable of pitching in at any time in any part of the code, we put a lot of effort in explaining to each other what we were doing to keep everyone up to date with the application structure.

Overall the team worked very well. During the project we never reached a point where we didn't know how to move on and continue working. We always had someone to ask if we got stuck and didn't know how to proceed, our advisor (Max Witt) did answer a lot of our questions concerning Android and gave us pointers on how the database should work.

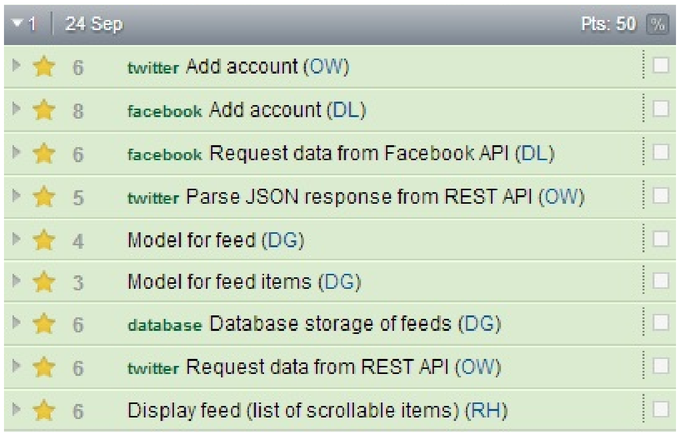
Doc

As touched earlier we were very content with the Scrum documentation workflow. The lessened workload compared to the waterfall project we did earlier this year was a blessing to say the least. Although we are yet to see the documentation as an essential and time saving process a lot of what was written this time actually seemed very relevant.

All documentation was written in Google Docs. This is really a wonderful way of working on documents, and worked very well. Everybody can view and edit all the documents at any time, and it has really been a big advantage working with this contrary to using local documents.

As for the javadoc we did not put as much time on is as we would have liked to but as a self explanatory code seemed more important, this was what we choose to lay our time on. Constant refactoring of the application made us put javadoc to the side as methods often were removed and thus the javadoc written in vain.

Sprint 1



1	24 Sep	Pts: 50 %
6	twitter	Add account (OW)
8	facebook	Add account (DL)
6	facebook	Request data from Facebook API (DL)
5	twitter	Parse JSON response from REST API (OW)
4		Model for feed (DG)
3		Model for feed items (DG)
6	database	Database storage of feeds (DG)
6	twitter	Request data from REST API (OW)
6		Display feed (list of scrollable items) (RH)

Coverage

Robin was appointed the assignment to started implement ANT and EMMA in our project. He managed to get the ANT working just fine, but didn't manage to set up EMMA. Olle however took charge over the EMMA tests and managed to resolve this issue. Currently both ANT and EMMA tests works fine.

As for now we're only testing a limited part of the application, this is because we have have mainly focused our time on getting the backend of the application ready and fully working. The obvious is of course to test every part of the application, but due the limited amount of time we have not reached as much coverage as we aimed for. We are testing the model, database, both the parsers and partially the client side classes.

While we wrote the tests we also managed to find some problem in the classes we were testing. An example of this is sending in objects with null value into the database could cause runtime exceptions. Since we found these problems while creating tests we were able to make the database methods more secure.

Since we can build our project through ANT and EMMA we're able to find out the coverage of our application. We didn't find out anything with EMMA that we didn't know already but it's very handy to find out how much of the application we're testing. Of course we didn't write a lot of tests so that number isn't very high. It doesn't hurt to find out the coverage but it didn't really give us anything, more than a few interesting numbers.