Final progress report

Antenatal Care Team
Team Leader: Dan Lindberg

Iteration 1:

Iteration 1 started with two use cases, entering patient information and generating a midwife's report. Because the project guidelines were not quite clear what the midwives report should entail the group focused on the first case. This use case was split between the two teams, one working on the object models and services, the other working on the view and controller. Overall this iteration went smoothly, our estimates were not far off, we definitely needed more work on how models were passed between the view and the controller, instead of one object we used multiple primitives, which means changing the model slightly requires a large refactoring of the controller . One thing that helped this iteration is that many of the functionalities that were needed were also needed for the first assignment. This helped complete our biggest triumph of this iteration, working persistent storage.

Iteration 2:

Iteration 2 saw a drastic change to the project. One things that was completed before any of the teams started working on individual tasks, was DTO's. This simplified passing objects around as they were no longer just primitives but an actual object. After the DTO's were made though the two things that really needed work were the view and controller. The first team took to work on the GUI. Using the DTO's they were able to complete a large portion of GUI functionality such as displaying receiving stored information, saving input information, and return an error DTO that would display an error message for some of the fields. Overall though the GUI code was very large and very clunky though. The other team worked on the controller. By switching over to the DTO most of our code from the previous iteration was no longer useful. This led to a major refactoring to include object passing using the DTO's. This was very important because it allowed for low coupling between the GUI and the controller that the MVC model calls for. One thing that was a bit of an issue following that model is that some of the listeners were in the GUI but needed to be in the controller, but other than that the controller and GUI were separated very well.

Iteration 3:

Iteration 3 was mostly about validation and unit tests. In the previous Iteration we added DTO's to pass objects between classes. When information is put into the DTO it is

validated based on a regex and then an error message with all the fields that are incorrect is created and handed back to the view. This was shown in the previous iteration, but not all the fields functioned and mostly it was the simpler data that validated, like a number for parity. This couldn't be the case for a real application though and everything needed to be validated. Most of this was just creating regular expressions for each field. There was also work done on the GUI and controller to fix the fact that there was a listener in the GUI and not the controller, separating the two even further. The other team worked only on unit testing for this iteration. This was our biggest mistake this iteration and probably the whole project. The scales of work were definitely not even and at the end we all realized this. We also realized that the team doing the unit testing, couldn't really do their tests until everything the other team did was complete or else they'd just be actively changing the unit tests every time the controller or GUI changed, thankfully most of the unit test pertaining to the services and data storage at this point and did not have to rewritten.

Iteration 4:

This iteration saw a major refactoring of most the data models, GUI and the controller. The model names needed to be refactored because most of them were from the start of the project. For example there was a class named AntenatalModel, which made sense in the first iteration, but in the fourth iteration didn't make sense. The class was now being used to represent individual pregnancies for each patient and those pregnancies now held visits so we renamed the class to pregnancy. Changing all the class and variable names made the code much easier to read and understand. We also split the controller into two service classes and the controller became the handler for the two services, this followed the facade model and made the controller much more easier to read as well as write for. In the previous iteration we were told that GUI code was too large and much of the naming and visual layout did not make sense. So we made methods for repeating blocks of code where we could, along with naming each method with a more fitting name and restructuring the GUI layout. We also wanted to integrate with the consulting registers code, but for reasons beyond our control their code was not in a usable state (see iteration 4 documentation), but we still had to create a way to switch between patients. This led to the creation of a very primitive terminal interface where the user would select a patient at start up. We also needed to create a patient wrapper class so patients could have multiple pregnancies. And if the consulting register had given us usable code this would have made integrating easier, where they would just have to give us a patient ID and then our service could retrieve the information on that patient. Another accomplishment of this iteration was the creation of a monthly midwife report. It was decided that it should be an extremely portable document so we opted for a simple text file. I can't speak for other members of the team but I really felt that we were in a groove at this iteration and this was by far the best one.

Iteration 5:

For the final Iteration we may have bitten off more than we could do, I guess you could call it hubris on some level. Our team had people who had done database work and web development so we decided that we would try and strive to complete integrating both of the new technologies. On the web interface front Team 2 was able to create an amazing looking static interface with HTML relatively quickly. Because of this we assumed that everything else would go as smoothly, it did not. Getting the java servlet to work took quite a bit of time and collaboration between the two teams, and while it was successful, we ran out of time to complete the corresponding javascript's for the HTML to make the whole thing dynamic. If we had not waited to start the servlet I believe we would have finished this task, not having the servlet made or knowing completely how it functioned made the javascript hard to write. The other team worked on converting our json persistent storage to an SQL database. At first the team wanted to try and use Hibernate to interact with postgres server but after hours of going down this route they backtracked and interacted with the database like in the class example. This ment manually typing out SQL statements that would be needed to create tables along with storing and retrieving objects from the database. The Monday before the project was due we got together and decided that because we had one of the new technologies integrated into the project we would abandon the web interface as none of us had time to work on it further. Because of this there are a few small issues that made it into the final product, like missing labels for unit measurements in the GUI, and one major issue is if you want to select a different patient than you must completely exit the application to select a new one.

Strengths and weaknesses:

I think one of the things our group has gotten much better at, and is one of our prime strengths is paired programming. In the first iteration we barely did any paired programming because we thought it would be a waste of time. As things progressed and became more challenging it made the work go by fairly quicker compared to doing it alone. But opposite to that our biggest weakness was actually having and finding a time to do paired programming. This became very apparent in the last iteration, everyone had their plates full so there was little time where we could get together until the very end of the iteration which made finishing the project much harder. Another strength that our team has is we were able to keep the coupling low, the controller ended up being very abstract through the facade model, which means if we ever had to do another major revision working on it would be relatively easy.

Remaining Key challenges:

The biggest remaining challenge is to get the web interface going. Like said in the iteration 5 summary, there are left over issues in the GUI, if the web interface were to be actualized none of those issues would exist because we would have a new GUI. We also were not able to integrate with the consulting register. By the time iteration 4 was complete we still did not have usable code and seeing as we talked them through why we couldn’t use it and what was wrong it was kind of disappointing that that didn't happen.