Jack Davey

**Software processes**

This chapter of the dissertation outlines the major software development tools and processes I used during the development of this project and also gives explanations on why I chose those particular tools over other alternatives.

When I began the practical component of my project I needed to make decisions about how I would actually go about doing the practical work. I believe that a set of good software processes is essential for doing this so I set this out early on.

I decided against using a waterfall based development methodology. This is because I have never completed a project of this size on my own before, and I am using a large number of technologies, which are unfamiliar to me. Because of this, I did not feel that the waterfall approach would give me enough space to move things around if difficulties hit my project.

I therefore chose the to use an agile methodology with elements of Scrum built in. I chose an agile approach so that I could split the work up into manageable increments. This had two major benefits. Firstly, it allowed me to work on each feature in isolation, meaning that I only needed to focus on integrating into the main system at the end of each iteration. Another benefit of using an agile approach is that it meant that I could easily keep an eye on the progress I was making, and make adjustments to deal with this. As an example, during my first iteration, I completed all the work too quickly, so I was able to plan to complete more work in the second week.

Once I made this decision, I created an iteration plan that detailed exactly how I would manage my time in each iteration. Originally I had planned to use one week iterations and have each one self contained. After my first week working on the project however, it became clear that actually spreading this over two weeks would be beneficial. This helped me for two main reasons; firstly, it allowed me to vary the tasks that I completed each day. Secondly, it meant that I could give each task more than one or two days to do.

It is also worth noting however, that this plan did not always run smoothly every week. As an example, my work for the second iteration (which involved getting a basic version of eventual consistency working in my project) took much longer than I had expected it to, and therefore ended up spilling into the time that should have been used to develop iteration three. In contrast to this however, some iterations, such as iteration one, took much less time than I expected, so everything evened itself out in the end.

Another important consideration I had to take into account was the programming language that I would be using. The first choice that sprang to mind for this project was Java. Java was the first programming language I was ever taught, so I felt that I would be able to get something going pretty quickly in Java. The language can be used in most situations, because it has so many libraries and plugins available for it. On the other hand however, I was concerned that the project might involve dealing with concurrency, and the model based on threads and locks that Java provides is notoriously hard to get right. I also felt that I would be playing it safe a bit with Java, and I wanted to learn a language that I had never used before.

Another choice I considered was PHP. PHP is ideal for web-based projects like mine. This is because you insert it directly into HTML code, and most web servers support it. On the one hand, I thought the fact that the language was easy to deploy to be a major selling point. On the other hand however, the fact that PHP is dynamically typed makes PHP programs harder to debug. To add to this, PHP has no built in support for concurrency. I therefore decided that the negatives of PHP outweigh the positives for this project.

The third language I looked at was Scala. Scala was built on top of the Java ecosystem, so all the benefits of programming in Java were also true of Scala. Another major plus point for the language was that it supported the actor model of concurrency. This involves not sharing any state at all, and communicating between different parts of the system through the use of immutable messages. I could see this approach working well for this sort of system. The reason being that it would be easy to translate a real system full of servers communicating with each other to an actor based system. Another good point about Scala is that it is both functional and object oriented. This means that if I wanted to write safe code for concurrency, Scala would let me do that, but if I wanted to take advantage of object oriented design methodologies, then I was equally able to do that as well.

These were not the only tools that I needed. One tool that I found I needed in particular was version control. Although I wasn’t working as part of a team, I did find that it would be useful to keep track of old versions of documents. Another reasons as to why I chose version control was the fact that it allowed me to ensure my work was regularly backed up to a server to ensure that nothing was lost.

I therefore created an account on Github for this project. This allowed me to perform the functions mentioned above, and also allowed me to easily share progress with my supervisor about my work.

Another tool that I found particularly helpful during my project was Jira. Jira is an issue tracking tool that allowed me to log and monitor the work I was doing as I was doing it. It also allowed me to provide weekly progress updates to my project supervisor, as I could indicate in a graphical way exactly how much work I had done.

As well as the Scala programming language, I used two other libraries to complete the project. The first of these was Akka. Akka provides an implementation of the actor mode of concurrency that can be used within Scala or Java programs. Actors mainly communicate via passing messages to each other, and do not share any mutable state. Because of this, it is not possible for an Akka program to suffer from bugs that involve race hazards. Another plus point of the Akka system is that it is very well suited to the project, in that it is easy to map a cloud server in a database onto an AKKA Actor.

The play framework was another invaluable tool during this project. This was the main supporting library that I used in my project. I provided tools for developing the actual web service itself, as well as parsing the JSON requests needed as input to my application. This allowed me to put the low level details to the back of my mind and focus on the actual task of implementing eventual consistency.

Design was another key element to my project. Thee were two main kinds of UML diagrams I needed here. Firstly, I used UML class diagrams to model the different types of SQL queries that my system could process, so that I could visualize the inheritance relationships between these before I started the coding work. When designing the overall system architecture however, I used the process network diagrams introduced in module CO890. These show the overall design of my system and show how the various processes communicate. I chose these over a standard sequence or state diagram due to the high amounts of concurrency involved in my system.

Now I will look at testing the application. Firstly, I used unit tests to test each of the core pieces of functionality. This form of testing served two purposes. Firstly, it allowed me to verify that each component was working as I intended it to as I developed the system. Secondly, it also meant that I could assess rapidly whether that piece of software broke the rest of the system by running all the unit tests that I previously created.

I used the spec2 framework provided with Play to do this. The main reason for this was that it allowed me to test the actual http requests in my code. This turned out to be incredibly useful, as otherwise I would have had to do this manually, something that would have taken a great deal more time and effort. I was able to integrate my unit tests whenever I rebuilt the system, so that I always had a good idea of what I’d broken whenever I changed a version of the software.

Another valuable tool that I made heavy use of during my project was static analysis. This was useful because it caught lots of stupid typos before I even ran the code. A prime example of this kind of situation was a time when I wrote a Boolean function that always returned false. I ran my project under several different static analysis tools each time I compiled my project, so as to catch the most errors possible.

Because I was using the Akka concurrency framework in order to make my application scalable. I could not use the standard Unit tests provided by Play framework to test all aspects of my application. This is because the actors in play are protected by the special ActorRef class, meaning that you can only send and receive messages to them, . The akka testKit enables you to get access to the underlying actors methods, meaning I could make full use of Spec2 to enable correct testing of my application.

I should also probably briefly mention Intellij IDE here. While not directly related to the success of my project, it did make things much easier because it allowed me to use all the tools from one place and meant I saved much more time than if I had had to switch between each tool individually.