Final Assignment Retrospective Timmethy Tran

The project started with the requirements document. I had a rough time wrapping my head around all the details needed in the document. I went by the template and didn’t understand why there needed to be so much details in the requirements portion. I also did not realize how long the document was going to turn out to be. I thought the document was going to be a few pages at most detailing major functionality. Little did I know that there needed to be much detail in every portion of the functionality and that this is the master sheet I was going to use to design my software. So because of this, I made a very poor requirements document and didn’t use it effectively. Instead, I kept referring to the assignment sheet to find my requirements, which took more time than if I had a proper requirements document. As class went along, I started to understand more the reasoning of a requirements document and how to better structure one so that I can actually use it. In the future, I will pay more attention to the assignment and create a document I can read and use rather than look back at the assignment details. To do so, I will start planning and asking question about the project earlier so I can get a clear grasp on what is needed.

The next document was the design document. I found this document extremely helpful not in reading but in the process of making. This document entails the structure and use of your software. It represents the preliminary stages of coding that everyone needs to go through to efficiently create software. It was here that I went through multiple structures of organization I can use for my software and wrote them all down. Once I had a structure I liked, I had it almost completely memorized. In doing so, I wrote a document that had a decent grade on it and with confident knowledge on what I’m doing. As previously noted, I didn’t really come back to this document. I had most of the structure memorized and the process of creating one greatly helped me get started on coding. I valued the process much more than the document, and really enjoyed this problem solving portion of software engineering.

Lastly there is the testing document. This document was less fun but still helpful. The process of creating this document teaches you to think of extremities to test your program on. Without putting a solid few hours of thought into this document, I would have been able to test as accurately. It helps to look back on the document after your software is finished to double check any faults that might occur. Now that I know of more faults that can happen after developing this software, I can use that knowledge to create better and more efficient testing documents.

During the coding process, I found a lot of things that did not work well. I haven’t been coding in about a year because of the class structure for the CS major. I have been taking mostly logic and structural classes rather than coding ones. So I was a bit rusty, especially on java, when I started coding. It took me a few hours to remember all the functionality, libraries, and rules of java after using c++ for so long. But using java was really entertaining compared to c++. So that was one of the things that went well. Once I got used to coding in java again, I could fly through the code once I had an idea of how it was supposed to be written. Using the language is very easily and the things that took me a long time to write were algorithms rather than syntax and rules.

Something that didn’t work well for me was the use of the external data file for my providers directory. I am not too familiar on how to use them in java, so my code I wrote for that portion was most likely very inefficient. For example, if the user entered a wrong service code, the report had to be reloaded each line at a time as the user entered the next code. It was a lot of unnecessary work on the system. I wanted to stick to my design document and leave the provider directory as an external file and not create a data structure for it. In the future I will have to alter the way I deal with external data files, even if it means changing my documents. We have learned that these documents are in fact fluid, so I should use that to my advantage.

One of the things I had challenges with also was the efficiency of looping and consistency in style. In some functions, I have loops run around helper functions, some helper functions have the loops, and some loops needed to be put into helper functions period. Using helper functions did not work so well for me in this respect because I am not fully back into coding yet, and haven’t developed my style. Having return arguments and looping for helper function was really challenging for me and created ugly code. To remedy this, I’m going to spend some time before my next project to look at my style and pick one that I stick to throughout and the project with efficiency in mind. I want main functions to look neat and clean with function calls and the helpers do all the work. To do this I’ll spend some time in the design phase to take a close look at the functions and their helpers to structure them properly.

Another aspect that didn’t work well for me was the structure I had created to begin with. At first, I thought the structure would work perfectly, and it did once I finished it. The model I had made was fit almost perfectly by my software. I realized too late however that it was not the most efficient model. I have separate classes for running activities and objects for storing information. This I still think works effectively and efficiently. However, they are all separate. I realized that a hierarchy would have worked perfect for this project. Having members at the very top with limited functionality, which is the followed by the providers, provider’s terminal, and the manager’s terminal. Because essentially everything the provider can do a manager should be able to, since they have access to that information anyway. A lot of the functionality can be shared between classes. In the future, instead of looking for a structure that simply works perfectly, I need to take into account efficiency.

One of the things that worked really well for me was an IDE. I was introduced to the IntelliJ IDE when we first learned java. Admittedly, I don’t know how to code java without it, but using and IDE made coding a breeze. Phrase detection, spelling corrections, error messages before compiling, auto indents, warning messages, they all help me fix problems before I code too far ahead. Null pointers for example are shown to me when I’m using them and variables that aren’t used or changed are also labeled. It just helps me keep track of what I’m doing so I don’t run with an error for too long, causing long fixing times. The compiler was also very helpful, as I can be directed to the point of error at a click, without even running a debugger.

One last thing that worked well for me was to keep the majority of the functionality within the terminals instead of the object for storing data. This kept me clear on where things are happening, especially when a problem occurred. Doing this also helps me narrow down where problems are. Since my objects didn’t have much functionality, they almost never had an error. It was almost always in my terminals. Its also important to note that keeping functionality out of my objects helped limit the unnecessary movement of memory, which reduced errors too.