COMP 313/413 - Project 3 - Reflection - K. Lieber

12/6/21

Group Development Journey

I found that the instruction to perform Project 3 as a group project illustrated a factor presented by Frederick P. Brooks, Jr. in *The Mythical Man-Month: Essays on Software Engineering* (1995) – mainly, that adding people to a programming project increases complexity and time. As Brooks notes, “In tasks that can be partitioned but which require communication among the subtasks, the effort of communication must be added to the amount of work to be done.” (17) This is because each participant “must be trained in the technology, the goals of the effort, the overall strategy, and the plan of work.” (18) In the end, I feel that as a group, we mainly set aside the “group” aspect of the programming as it added a difficult layer of complexity atop an already complex project which we each tried to solve independently to make sufficient progress in the available time.

A further key area working against the effectiveness of group activity was the differing aspect of each of our schedules and knowledge. My own level of knowledge required hours of private, intense study and mapping of the Android Studio project files before I could contribute. This meant that I was not immediately a good resource in contributing to the group and indeed, not until much later in the game when I’d unpacked all the code and finally begun to work with it.

The strongest group activity came out toward the end and was asynchronous. I appreciated that Sawyer’s inline notes helped understand where he had interacted with the code and found his work on the countdown decrementation led me to insights which were able to create the decrement. We used the convention of adding our initials to the inline code which helped identify each contributor’s notes. This made it easier to work asynchronously. I was considerably concerned about use of the shared repository as I tend to insert a lot of experimentation in the code, but found that as long as I documented it, I could create a reasonable set of breadcrumbs so that anything I tried to could be understood by the group and rolled back or changed if necessary.

Overall I found that group interaction on this project was primarily, very minimal, mainly from necessity in completing it in the time allotted. I felt that we were each focused on merely understanding the problem enough to work with it at our own individual level, but that in the end we each added value to the project. I think if we had all been graduates or all undergraduates (impacting availability and schedules, apps, coordination, leadership), and if we had all had a similar level of coding knowledge (impacting level of contribution or pre-study required), we might have had more immediate synergy in a group setting.

Relationship Between Extended State Machine Model And Actual Code

I felt that the State Machine Diagram by itself was only somewhat useful as a schema to understand and generate the states. I would add to it a diagram structure which I only sketched out for this project, but would develop further. This is a three-column advanced diagram as follows: the State Machine diagram on the left, the Android Studio file tree in the center, and snippets of relevant code on the right. This would assist in connecting the abstraction of the State Machine Diagram with the concreteness of where these actions take place within the Android Studio files, and the actual code that drives them.

To answer the question of whether it was more effective to model or code first, the answer is “it depends”. In this case, it felt like the State Machine model was not able to make *conceptual* sense until the code could also be unpacked enough to make *concrete* sense. The State Machine model represented an ideal construction, while the code reflected all the individual bits and pieces required to represent it in real life, and both parts had to be played off against one another to build a holistic understanding.