**Challenges and Lessons Learned**

CS 414 Object Oriented Design

Fall 2017

Project 5.0: That’s (Almost) All Folks

Team: Function;Chaos

**Zach**:

* Diagrams, documents, and design patterns are important: In previous classes, I was programming assignments alone, not in a group. But in a group, you need to understand other members code. This usually comes from someone explaining the code verbally. Well, when you have a sequence diagram, package diagram, or any other descriptive diagram, it makes the task of discussing code much easier. **I think this is the main take away for me from this class.**
  + Lesson learned: Diagrams, documents, and design pattern might seem like overkill when the program is small or you are working alone, but as a project grows, these artifacts can be extremely useful for group work, debugging, and even refactoring! USE DIAGRAMS!!!
* Time management: One challenge I faced with this project was deciding where to focus my work. I found that the project was much larger than I first thought. On the second and third iteration, I overextended my scope of deliverables to work on. This caused me to produce some work that was lower quality than it could have been.
  + Lesson learned: A simple list of tasks done and yet to do is very helpful. This list could also include other group members tasks done and yet to do so that the group can visualize how much each member is contributing. This also helps with basic organization. Even though it is overkill sometimes, it can help the group get tasks done and spread work evenly. GET THE BASICS DONE FIRST AND GO FROM THERE! **KISS!!!!**
* Separate group work: For the first iteration, this wasn’t a problem because we sectioned out the work evenly. But as the project grew, I noticed our group start to struggle when we would merge our group work. This caused our group to work less efficiently; multiple people working on one task while some other tasks weren't being worked on. Eventually, the source code got so complex that it was difficult for individuals to switch from one task/use case to another. Eventually, our “master branch” for the code was housed locally on Martin’s computer because he was the only one that had it working. Eventually, Martin was the main contributor to the code. This was obviously problematic when others would try and hop in to help finish use cases and eventually, Martin took on more than his fair share of responsibilities.
  + Lesson learned: Use github (and maybe waffle.io)! It is a powerful tool for programing in teams. We should have made sure everyone was able to clone the master repository and run the program on their computer. This would have saved a lot of time and would have made it easier for everyone to continue to contribute to our program. Additionally, our group should have noticed this earlier and corrected the situation. However, it is tough when some members have very little experience with github and version control. STAY ON TOP OF YOUR WORK LOAD!

**Martin**:

**Paul**:

Challenges

* Communication
  + Keeping everyone up to date and on the same page
  + Arranging group meetings
  + Using github
  + Distribution of workload
* Everyone coming in with very different backgrounds and skillsets
  + Some had never used github
  + Some had never used databases
  + Some had never made GUIs

Lessons Learned

* Start working earlier
* More frequent, more regular group meetings
* Make more use of github for version control
* Need more group communication and coordination
* Agree on design before starting implementation, and then stick to that agreement

**Shun**:

* Pattern Connection: One challenge I faced is the different pattern and different idea to the hole design. Although the concept is similar, but for team working we should made a realizable product. Especially some of the skills and idea would take more time to realize but the time is not enough.
* Connection to the code: The pattern should be fixed after the code is made because the different designer may code in different ideas. Although the code would be different to the original design, some of them would be better when we realize the function than without actual coding.
* Communication: Team should have a same concept for some function and method to realize in the code. The diversity may begets to some argument. Thus, more meeting will increase the consensus.
* Combination of the code: Different coding style may begets to the project error on different interface. The refactoring may solve this problem.
* Separate working: Most members in our team has the idea to the code, but Martin were doing main coding. Here, we face to combination of the idea and the code. Using github make it easier but still difficult on material discussion. However, there are some problem with the meeting and communication problem, and the distribution of the work. Here I have learnt that every meeting should have a clear list and work orientation, thus the work will work on the process.

**Alber**: