Circuit Solver Reflection Statement

Collectively, we learned…

- We need to work on our estimations of how long tasks will take
- In consideration of the difficulty in implementing the image processing, we should have planned for and placed higher priority on building up the draw circuit use case in order to have a minimum viable product by the beta release
- We probably should have put more effort into the UI at the beginning of the project, since we underestimated how long the UI would take and how many issues and iterations would be required
- Scrum may have been more effective if we knew how to implement it from the start.

Moreover, we learned that...

Due to a lack of experience and knowledge with design patterns during the designing phase of this project, we never included design patterns in the original design. We did implement design patterns later, when we realized, after the fact, that they would improve the maintainability of our code. However, had we planned to implement the design patterns from the start, we would have saved a decent amount of refactoring time. Our implementation of the Factory Method for our CircuitElm class is a good example of this. Since we didn't clue in to implement Factory Method for instantiating CircuitElm classes until much code had already been written, we lost some time having to refactor everything properly. In short, more thoughtful design in the beginning would have saved us time in the development phase.

With respect to usability testing, the main learning experience gained was how vitally important regular feedback from an expert not working directly on the project was - namely, the weekly usability test sessions with Satish. These weekly meetings kept our progress in perspective and ensured we were on the right track. Additionally it motivated us to continually push forward and improve iteratively each week based on last week's feedback. Not being directly involved also meant that Satish could provide a user-perspective rather than a developer one. We are satisfied with our progress, despite solely having Satish's feedback as usability testing. The application has now reached a point where we could start performing more rigorous usability testing with actual clients, however we are glad we did not go this route (potentially very time-consuming with high risk of useless feedback) and instead pushed to get a minimally viable product first. Instead, many of us wish we had gotten such weekly usability testing feedback from an expert on various other projects we have worked on. CPEN 321 has solidified how important this is by simply comparing the vast difference in progress with such regular feedback vs without it.
For most of our team members, this was the first large software project we had ever done. This project taught us how to communicate effectively. We communicated through in person meetings, skype calls, slack, and through useful commit messages. It was important to ensure each team member understood their tasks for the week, and for each team member to

understand each other's progresses. We kept a master branch, which was almost always in working condition, and developed features/fixed bugs in other branches before merging into master. By the end of the project, we all learned a lot more about communication, and using git. Since we were unable to do any extensive planning due to our lack of experience, and needed to produce a product for the demos, we adopted the SCRUM process.As mentioned above, for most of the team this was the first big software project we had ever done. Throughout the semester we got a first hand experience of how fluid requirements are and how often they change and shift when you encounter different problems throughout the project. Although this project wasn’t the exact same as a typical software project since we don’t have a customer who has requirements, they were all just hypothetical at the beginning. With very little experience it was also very difficult for us as a team to predict how long tasks were going to take. But now I feel like we all have a better sense of the fluidity of requirements, that requirements change, and sometimes things take longer than expected, which may cause another feature to be sacrificed.

The social part of this project was extremely interesting and helped us a lot to know about the creation of a working group and how it develops. With some distance, a lot of experience can be taken from how easily the cross-functional concept of the SCRUM method has been naturally applied to our group. Every team member knew very quickly what domain he was interested in and consequently we could begin very fast with the work to be done. Also, and according to the SCRUM model, there was no need to give a specific role to each of us, as some natural leaders and organizers directly brought a clear guideline to the overall working of the group and the flow of the project. We could note that having only good programmers in a Software Engineering team was not enough to guarantee the success of the project, but the organization and the communication are as important.