NAU–CS Team Project Self-Reflection Worksheet

Overview: At the end of a project, it's useful to go back and reflect on how the project went, how the team functioned, how effectively you used tools, and so on. This worksheet is designed to guide you in this process, and capture the outcomes.

How to fill this out: Hold a final team meeting, after you've turned in the last deliverable and the heat is off. Order a pizza, crack open a beverage. Then sit down as a team and go through the following worksheet, discussing and filling in each section. Type up and the result, and email the document to your team mentor.

Grading Metrics: You will not be graded on the *content* of this document per se. That is, if for instance, your self-assessment concludes that you "didn't use version control tools effectively", then this shortcoming won't affect your grade; the point is that it should be an honest assessment. What you *will* be graded on is *how well* you fill in this document: thoughtful self-analysis gets a perfect score; cursory/lame/vague self-analysis will score low. We instructors use this document to help us think about how to encourage more learning and better teaming on projects, so please help us out!

Team Name: Team Controller

Team members: Zachary Parham, Brandon Udall, Bradley Essegian, Dylan Motz

Course number and name: CS 486c - Capstone Experience

Semester: ___Spring 2024___Date this reflection completed: ___5/3/2024__

Software DESIGN PROCESS

How did your team structure the software development process? Did you choose a particular formal model (SCRUM, Agile, etc.). If so, which one and why? If not, did you explicitly agree on an informal process…or was it just pretty random. Explain briefly.

The team structured the process around the Agile process. We decided on this approach because of the limited time to develop the application and also it would provide the clients with weekly updates to show our progress.

How did it go? Now briefly discuss how satisfied you were with this process. Did it work well for this project? Why or why not?

The approach went well. Dividing the project into bi-weekly sprints then dividing the programming issues into these sprints allowed us to keep up good progress towards the end goal. It helped us reduce the programming problem to sub problems which were not all that intimidating when they were small enough.

What changes might you make in your development process if you have it to do again? More structure? Less? Different process model?

Overall, the agile process of development that our team decided on worked out really well in the end. It held us all accountable to resolve issues by the end of our biweekly sprint, and made it easy to divide up the work for the whole semester so that we would finish on time. We don't believe that we would change much about our development process because of this.

Software DEVELOPMENT TOOLS

What software tools or aids, if any, did your team members use to support or organize software development? For each of the following categories, list the tool(s) used, and briefly describe how the tool was actually used. If you didn't use a formal tool, explain how you handled the matter with informal means.

- Source creation tools: IDEs, text editors, plugins, anything used to edit/create source.
- Version control: How did you manage your codebase?
- Bug tracking: How did you keep track of bugs, who was working on them, and their status
- UML modelers and other miscellaneous tools:

For the main technologies, we decided to use QT for a GUI paired with C++ for the backend. The team decided to use the QTCreator IDE to facilitate active development of both the GUI and the backend. This is the given IDE for QT development and worked very well. For version control and bug/issue tracking we use GitHub and Git. The team found this approach very easy as all of us had experience with Github/Git in the past. The branching here was key for development along multiple different issues simultaneously.

How did it go? Comment on any problems or issues related to organizing the coding process. How might you have managed this better? Were some tools you used superfluous or overkill? What tools or mechanisms would you try next time to deal with those issues better?

Overall it went well, if we could change one thing it would be clearly defining the roles of the team up front and specializing each of us into a different part of the project that needed to be developed. Eventually we all found specialties which each member could focus on but it would have been nicer to have these defined up front.

The tools we used were almost ideal. Qt has a massive library and we selected useful components as we went including testing, serial communication, settings (to store variables cross-session). We used discord for communication and github to track progress and store our code base.

TEAMING and PROJECT MANAGEMENT

Without getting caught up in detailed problems or individual blame, take a moment to think about how your team dynamics worked overall. Here are a few questions to guide you:

How did you organize your team? Did you have some clear distribution of team roles (leader, technical lead, documentation lead, etc.) up front? Or was it more just "everyone does everything as needed"?

Leader - Zach
Front-End - Dylan
Back-End (Controller Simulator) - Brandon
Back-End (Data Manipulation/Testing) - Zach/Bradley

How did you communicate within the team? Comment on each of the following communication mechanisms:

- Regular team meetings? If so, how often?
    - Team meetings occurred virtually every Monday at 5PM
    - Issue refinement meetings occurred virtually every other Friday at 1:30PM
- Impromptu team meetings? If so, roughly what percent of total team meetings were of this sort?
    - ~10%
- Emails to all members? If so, explain briefly: about how often, what used for?
    - Mainly for emailing clients/mentor
        - Once a week
    - We never emailed each other since we used Discord for communication instead
- Software tools? Were any of the software tools you mentioned above (e.g. bug/issue tracking) using to communicate and organize tasks, e.g., in lieu of emails or other discussion?
    - Used Github to organize our issues + deliverables for each sprint, as well as the Notion board for the calendar
- Other communication channels used? Facebook, wiki, text messages, phone conferences, etc.
    - Used Discord for all our communication

How did it go? Did you feel that intra-team communication overall went well? Were there breakdowns, e.g., where someone didn't know something was due, didn't realize a task had been assigned to him/her, did not know about a deadline, etc.? Without getting into details, simply comment on whether such breakdowns occurred, what the overall cause was, and how serious (if at all) the consequences were.

The communication strategy was effective. We used a more professional tone in the form of emails with the client and mentor. The team used discord to communicate and organize tasks every day so this meant that the team was well organized and kept on top of deadlines and deliverables.

What could you do better? More structured leadership? A more formal task assignment/tracking system? Using better/other communication mechanisms? Generally just think about what you all would do next time to improve communication and avoid breakdowns mentioned.

Because we never really encountered any "breakdowns", there isn't too much that we would change with our communication mechanisms. All team members were aware of the work they needed to complete each week, as well as being well-informed about upcoming meetings and deliverables from a neatly structured Discord server.

Nice work! Congratulations on finishing your project! Please enter all of your answers in this document and send it off to your instructor and team mentor, and submit it on Canvas.

Some closing thoughts…

Spend a little more time on your own percolating on the answers you gave in this self-reflection exercise. Being effective as a project team is ***not easy***, and is a skill that we all have to work on continuously. There is rarely any single or simple reason why a project was a bumpy ride; usually it's a combination of factors…of which is YOU. Regardless of project or team, there are things that could have been done differently to make it flow better. Recognizing those things through thoughtful reflection post-facto is the key to improvement!

Brandon Udall - What I really liked about this project was that it introduced me to agile and the Qt framework. Both of these are useful tools which are commonly used in industry and I feel as though I grew as a software developer throughout the course of this project. Also this capstone gave me the chance to meet the other members of Team Controller which has been a valuable experience.

Dylan Motz - This project was fun to work on as it introduced me to new technologies. The Qt framework definitely made this project more enjoyable as it provided us with everything we needed and made development easier. Also getting

more experience using github was very nice as I am much more comfortable with using github now. Overall this capstone helped me become a better software engineer and it was fun working with the other members of Team Controller.

Zachary Parham- This project made me realize how much I really enjoy project management. I found that organizing a small team of software developers around a central development goal was very satisfying and makes me excited for all of the roles where I can learn and grow my leadership skills. This project also helped me develop my interpersonal skills when working with clients and project owners. Working in a very corporate environment, these will certainly come in handy.

Bradley Essegian - This project introduced me to serial communication, which was the part that I was most intimated by based on the initial project guidelines. We got to create our own serial protocol during development of the Connection class and I learned a lot of valuable things about serial communication. Researching the Qt framework and its libraries was significant, and I'm excited to continue using the framework beyond its serial and GUI capabilities.

Stay tuned as Team Controller is still working on some more projects after college!