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 LoRa

Team members: Ryan Wallace, Benjamin Couey, Brandon Salter, Mohammed Alfouzan

Course number and name: CS476: Requirements Engineering/ CS486: Capstone Design Experience

Semester: Spring 2020 **Date this reflection completed**: 5/6/2020

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.

For the design process we didn't use a formal structure. Our informal agreement was that Ryan and Moe would work on the front-end half of the project (the LoRaMessenger library and demo app) while Brandon and Ben would work on the back-end half of the project (the proxy server and configuration service).

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?

With our informal process we managed to finish the project but it was a bit of a struggle.

We found that trying to seperate the project into a frontend and backend was counterproductive. There was a high degree of coupling between the proxy server and the LoRaMessenger library: neither worked without the other. As such, trying to seperate the development only obfuscating how the two halves would be communicating to one another.

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

As a team we definitely could have benefited from having more structure especially when it comes to due dates for tasks. Oftentimes someone would be assigned a task but would be quickly forgotten or untouched for a long period of time. Towards the end of the project we switched to a more advanced and robust Gannt chart that helped guide our team with due dates.

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.
  Android Studio
  VS Code
  Atom

The android team needed to use Android Studio out of necessity as it is the only real Android/Kotlin IDE. The proxy server team used an editor of their choice. Both teams used the Git plugin that was compatible with their editor of choice.m

- Version control:
  Git/GitHub
  GitKraken

Of each development pair for the Android Studio Library and Proxy Server, one person was in charge of version control. The Android Studio team used the built

in Version Control integrated with GitHub within Android Studio. This made keeping files up-to-date simple and caused very few merge conflicts. For each component within the Proxy Server, as someone would update or create a file, they would push their changes to their own dedicated branch within the Git repository. After which the version controller would be responsible for accurately merging the files. This is how our team managed the version control aspect of our project and kept our files up-to-date.

- Bug tracking: How did you keep track of bugs, who was working on them, and their status

We didn't do any bug tracking. We never got in the habit of making good consistent use of a Kanban board or similar organizational tool. Each person was responsible for fixing bugs along the way during each development phase.

- UML modelers and other miscellaneous tools:

  We have used draw.io as our UML builder tool.

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?

To improve our team in the future we would use a Kanban board to track issues and bugs along with monitoring the process of each task. Our team felt that most of the tools we used were necessary for the project.

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"?
Our team's organization was adequate and equally distributed.

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

- Regular team meetings? If so, how often?

We met on Discord or Zoom at least once a week, often more, and also frequently met after in person meetings with our team mentor or client. Later in the second semester, we also began having weekly in-person meetings that doubled as a time for code review and paired programming.

- Impromptu team meetings? If so, roughly what percent of total team meetings were of this sort?

We had a handful of impromptu meetings, but most of them were planned 24 hours in advance. I would say on average 20% of our meetings were "impromptu".

- Emails to all members? If so, explain briefly: about how often, what used for?

Emails to all team members were generally used when sharing documents with everyone, or CCing everyone when sending off an important email to the team mentor or client.

- 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?

We made very frequent use of Slack as a way to quickly send short messages to everyone on the team, coordinate meetings, etc.

Besides that, we gave names to our git commits and commented our code but this was never relied upon as a means of communication.

- Other communication channels used? Facebook, wiki, text messages, phone conferences, etc.

Occasionally, individual team members would communicate with text messages or private messages on discord.

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.

Communication overall went well. Everyone was always up-to-date on assignments and their due dates. Our team always gave everyone plenty of time to work on their assigned tasks. We never missed a deadline for any assignment.

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.

A more formal task assignment and tracking system would have been very helpful. I feel both to organize our work and hold ourselves accountable for what we had to get done.

Past that, we had an issue with Benjamin being a bit of a "do-everything-myselfer" which later down the road led to issues with him have a deeper understanding of much of the project than other team members. He also didn't do a good enough job of informing his team about all the things he did on his own. In the future, Benjamin needs to both hold himself accountable for explaining the parts of the codebase he developed.

Brandon's schedule was a bit difficult to work with; he was only available in the evenings which sometimes made things a little rushed to finish. In the future we might consider doing a better task management system, like a kanban board, for task assignment so that members with time difficult schedules could be assigned work ahead of time and check it during their free time, rather than wait for a meeting.

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

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!