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: Phonetic Evolution

Team members: Krystian Bednarz, Kenzie Norris, Preston Lee, Sam Asher

Course number and name: CS486c, Computer Science Capstone Experience

Semester: Fall 2023 Date this reflection completed: 12/14/2023

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.*

Our team did not adopt any formal model for our software development process. We instead split up the project into multiple components, assigned team members to complete each component, and we met as a team to integrate those components into the main project. We accounted for when demos needed to be conducted by, and started programming weeks in advance prior to the presenting of our program in those demos. Informally, we decided as a team which components of the software had priority, and then did our best to knock them out in that order.

We used the initially assigned team roles as a guideline for divvying out initial work, and from there we got a better sense of what each team member excelled in. Initially we also had a team discussion where we took inventory of what everyone already had some knowledge/skill in. It helped that as one person worked on an area of the project more, they became more familiar with it over time and it was a lot easier for them to add more features and changes to that particular component.

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?*

At times this method worked well, though there were times when we all needed to meet together and dedicate full days to software development. We started working on project components in advance, so our workload was more spread out instead of condensed to the few days before a demo deadline. The only improvement to the development style we worked with would be to start programming even sooner. Many team members had heavy workloads and busy schedules with other classes, and this became a very difficult obstacle at times. It's hard to work around 4, at times, very different schedules.

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

More structure would help, maybe assigning personal deadlines for project components outside of the demos already set.

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.
  - Atom: Universal text editor, used for HTML and PHP files
  - VScode: IDE also used for editing HTML and PHP files

- Version control: How did you manage your codebase?
  - GitHub: Codebase hosting platform, used for holding branches for individual components, and sharing code between team members

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

    - Discord: team members posted in the chat regarding problems and kept tabs on them.
    - Task Report: We created our own custom version of the original task report format and used it to keep track of who was working on which task and any issues that they encountered.

- UML modelers and other miscellaneous tools:
    - VirtualBox: Virtual machine, used for code developments that were made more convenient on a Linux operating system
    - LucidChart: for UML, software architecture, version control diagrams.
    - OnlineGantt/Monday.com: for creating Gantt charts.

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?*

Connecting components to the database involved using a Linux machine, which was cumbersome for most team members who used computers running on a Windows operating system. Database management proved to be easier on Linux machines however, so our team members using Windows opted to use VirtualBox to replicate Linux functionality.

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"?*

We assigned each team member to their own specific roles. The roles we came up with were a Team Lead, Database Coordinator, Documentation Manager, and Release Manager. The Team Lead handled work distribution between members, client and mentor communication, and more visual aspects of project development (Presentations, videos, website graphics, etc.) The Documentation Manager handled document development, adhering the papers we wrote to the same format, while detailing project development plans throughout. The Database Coordinator handled backend functionality and data transference for all components. The Release Manager handled deliverable posting on our team website, while keeping track of the branches that held each project component. Everyone on the team was responsible for programming in some form, but these specializations made keeping track of who needed to do what much easier.

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

- Regular team meetings? If so, how often?
    - We made the effort to try to meet at least once a week every week. At times some team members could not make it to the regular meetings, but this was handled through direct messaging and meeting minute document creation for looking at later.

- Impromptu team meetings? If so, roughly what percent of total team meetings were of this sort?
    - Some weeks we met as a team more than once, usually no more than twice a week. Of the 25 or so meetings we had, about 5 were extra meetings, so 20% of the meetings were extras.

- Emails to all members? If so, explain briefly: about how often, what used for?
    - We rarely sent emails to each other, at least between team members. Instead we would do mass pings in discord as our main form of communication.

- 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?
    - GitHub was used to share code between team members
    - Discord was also used for code sharing
    - Project components were separated into individual branches on GitHub

- Other communication channels used? Facebook, wiki, text messages, phone conferences, etc.
    - Most if not all communication and documentation sending occurred on our private team Discord server. our task reports, important documents, and meeting minute documents were all linked in separate channels on that server.
    - Text based remote communication was held in a separate channel, used to talk about everything while our team was not together in person.
    - Team meetings occurred in a separate voice channel, where we talked to each other remotely each week. Sometimes mentor meetings occurred in the voice channel as well.

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.*

For the most part, team members were able to effectively communicate in a timely manner through our text chat on our Discord server. We updated our team whenever progress was made, when something needed to be done before some deadline, or when we needed to meet in person next. The only miscommunications that occurred resulted from assumptions being made that were not made known by asking about it beforehand. At one point a document was fully written by one member, but was not reviewed by the other members until the deadline, and it needed to be heavily revised in a hurry before submission. The document was submitted after an extension, but the team needed to spend the time to immediately fix the document, where it would have been better to instead report that the document needed revisions days prior.

At times some team members could not make it to the scheduled team meetings on time. This occurred due to non college-related issues happening before the meetings began. This wasn't much of an issue, as information talked about in the meetings was made known to the absent team members. In addition, absent team members always let the rest of the team know that they can't make it, so everyone was informed at all times.

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.*

An important lesson we learned regarding communication was to minimize the assumptions made about if something was actually finished or not. You can't say something is done yourself until everyone agrees that it is actually done. The assignment could have some glaring overlooked issue that needs to be addressed, so it's important to finish things early to give other members time to review to catch those issues. Especially when writing documentation, the more team members that you can have reviewing the work the better. Many times during this multi-person review process we caught errors that other team members had simply overlooked.

Better usage of github would have also helped with the development process. We often had different versions of the final website floating around, and had to make more concerted efforts to bring them all together, so that everyone was working on the most updated version.

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!*

Overall, the project had its ups and down. A few of these major aspects we could've improved in the future could be broken down into a few things.

Lack of PowerPoint Practice: Our group needed a lot more practice for the powerpoint, or at least some desensitization to the idea of the presentation. Some of our work moments happened when we were presenting our project in front of others, professors or even our client. Not everyone in the group felt comfortable with presenting, and this caused many issues during those moments.

Time Management: Despite us getting the project finished, the coding part of the project should have been started much earlier. The group had decided to spend time over the summer to get ahead on the project, but the result ended up with very little communication over the summer, and not properly working on the code until the Team Lead had to heavily encourage the group to sit and code for a set amount of time. As a team we should've been in charge of our own tasks and time management and keep each other more accountable.

Miscommunication in General: At times team members would assume parts of the project or documentation were still being worked on, while others assumed what was currently completed was good enough. Asking more questions and double checking with everyone multiple times to ensure everything is being done would remedy this.

We are happy with our product in the end and despite everything we made it out ok.