**Meeting Minutes**

**Date:** January 27, 2014

**Start Time:** 8:00pm

**End Time:** 10:30pm

**Members Present:** Drew Aaron, Michael Beaver, Clay Boren,

Chad Farley, Andrew Hamilton, and Travis Hunt

**Members Absent:** N/A

**Topics** **Discussed**

* Identifying a lifecycle process to adopt
* Identifying potential programming languages
* Organizing the team into subgroups
* Establishing preliminary standards and protocols
* Second round of client questions

**Decisions and Actions Taken**

The team discussed the pros and cons of the Waterfall Method and the Hybrid RPM-Waterfall Method. The team decided that adopting an agile method would cost too much time to learn and integrate. According to Chad, the Waterfall Method is a good method, except for the major flaw that it is extremely difficult to return to prior design phases to address problems. The Hybrid RPM-Waterfall Method allows for more client interaction at the beginning of the design lifecycle. The team has elected, therefore, to adopt the Hybrid RPM-Waterfall Method.

The team discussed the pros and cons of various programming languages. The team seems to favor a combination of C++ and C# at this time. The backend could be developed in a combination of C++ and C#, and a frontend could be written in C#. The compatibility of C# with C++ makes linking the backend and the frontend potentially much easier. At this time, the pros offered by C++ and C# appear to outweigh their cons and the pros offered by the other proposed languages. For more information, see the corresponding table in the Supplemental Information section.

The team will initially begin as a 3-3 frontend-backend split until the initial prototype is presented to the client. Once the prototype is accepted, the team will form a 2-4 frontend-backend split as more attention is paid to developing the backend and the other team members begin modeling and developing the frontend. As the frontend effort becomes easier, the team will assume a 1-5 frontend-backend split to invest as much effort as possible into backend development. As the backend nears completion or working status, the team will revert back to a 3-3 or 4-2 frontend-backend split, as necessary, to integrate the backend into the frontend.

Chad is investigating a file-locking system (mutual exclusion) for GitHub. He will report his findings at a later date.

The team established a Facebook group to facilitate group discussions. The team will not post pictures or code, and the team will have to exercise caution when posting events. Facebook’s terms dictate that it assumes the rights to (read: ownership of) pictures, code, and events posted. Hence, the Facebook group will be reserved purely for discussion purposes.

The team established preliminary communication protocols. Team members will carbon copy (CC) Michael on all project-related emails for the official record. Subject lines of any project-related emails should be preceded with “CS 455.” Team members are reminded that these emails will be collected as part of the project portfolio. As such, all emails should be checked for errors and retain professional language.

Version control will be maintained via the GitHub desktop application and the built-in Git tools in Visual Studio 2013. Before working on project code, team members should fetch commits from the GitHub server. If commits are available, then the team member may pull the commits and work on the modified code. At the end of each working session, the individual team member will make a local backup of their work. When committing changes, the team member should include the file name and a short description of the changes in the required commit comment.

It was determined that it is still too early in the development lifecycle to establish design, documentation, and programming standards. The team will develop these standards at a later date as the project starts to mature.

Drew will provide GUI mockups and screen captures of the MARIE environment for the January 28 meeting with the client.

The team still needs to establish the following: schedule deadlines; design standards; programming standards; and, documentation protocols.

The team developed a list of questions for the next client meeting, which will be Tuesday, January 28, 2014 at 7:15pm in the main classroom. The next team meeting date, time, and location will be decided at the conclusion of the client meeting on Tuesday.

**Supplemental Information**

|  |  |  |
| --- | --- | --- |
| **Potential Programming Languages** | | |
| **Name** | **Pros** | **Cons** |
| C++ | Familiar, powerful, potentially good for backend development, intuitive code, great for OOP | The client is hesitant toward C++, difficult to work with pointers and dynamic memory, somewhat difficult to work with threads |
| C# | Fusion of Java and C++, less restriction, easy implementation of pointers, good for concurrency, “easier” threads, great for server and network code, can run C++ code | Code is less intuitive, code is messy, C# is not as well-suited for OOP, lack of strings by default, memory management can be tricky |
| Java | Good for GUI development, secure code, easy to learn, JVM | Unwieldy, lack of C++-style pointers, cumbersome, memory issues, perhaps not the best option for backend development, JVM |
| SNOBOL | Good for writing parsers, very powerful, good for data manipulation, fairly easy to learn | Only Chad has experience with SNOBOL, not a mathematical language, could be difficult linking with another language |
| VB.NET | Easy to learn, easy GUI development, intuitive, plenty of example code available, fast to develop with, inheritance is easy, case-insensitive | Slightly less control than C++, OTHERS TO BE ADDED |

**Questions for the Client:**

1. When will the subset of ASSIST/I instructions be made available?

2. What is the “identifier” for the “Final Run” command in ASSIST/I?

3. What is the maximum size of memory for the IBM/360 with ASSIST/I?

4. Does the GUI need an input field, such as the one in MARIE?

5. Do you want output to be changeable (ASCII, Hex, Decimal) as with MARIE or as ASSIST/I would normally represent it?

6. Would you like a “view .PRT file” button or option?

7. Where would you like the output to the screen to be? Would you like it to be the traditional command prompt style or like a modified version of MARIE?

8. How do you want memory dumps from crashes to be displayed?

9. On the detailed project description, which specific features are required? Which specific features are optional?

10. What exactly do you mean by “set tabs to implement correct code layout” in the detailed project description?

11. What are your thoughts on the following names: ASSIST/J; COBRA; ASSIST/II; ASSIST/I++; and, ASSIST/101?