**Meeting Minutes**

**Date:** February 13, 2014

**Start Time:** 7:10pm

**End Time:** 7:50pm

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

Andrew Hamilton, Travis Hunt, and Dr. Patricia Roden

**Members Absent:** N/A

**Topics** **Discussed**

* Programming Language Proposal
* Client Questions #4
* Software Requirements Specification Document
* Test Cases

**Decisions and Actions Taken**

The client will review the team’s programming language proposal and will respond with approval or disapproval at a later date.

The client and team discussed the fourth set of client questions. The client’s responses are attached. The next client meeting will be after class on February 18, 2014.

The team will complete the software requirements specification document by the end of the week. The team will review the draft and email it to the client for review by the February 18, 2014 meeting. The team should have a complete software requirements specification document by February 20, 2014 but no later than February 25, 2014.

Andrew shared his format of and examples of test cases with the client. The client has approved the general format.

Chad will be updating his high-level backend design to be in accordance with the client’s new requirements.

Michael will send out the updated software requirements specification document Friday (February 14, 2014) night and the team will be responsible for responding by Saturday (February 15, 2014) night.

**Supplementary Information**

**Client responses to questions:**

1. Who are the target users of this software? Could you describe them in detail?

**Drew Aaron:** CS 310 and possibly CS 311 class. Assume users have had CS 155 and CS 245 and know binary and hexadecimal.

**Michael Beaver:** CS 310 and possibly CS 311 students. Assume students have had CS 155 and CS 245. Students should have some programming experience and knowledge of binary, hexadecimal, ASCII, and EBCDIC.

**Clay Boren:** CS 310 and possibly CS 311. Students have a little programming experience.

**Chad Farley:** CS 310. CS 311 possibly.

**Andrew Hamilton:** CS 310 and maybe CS 311. Prerequisites: CS 155 and CS 245.

**Travis Hunt:** CS 310, possibly CS 311. Prerequisites: CS 155 (programming) and CS 245 (binary and hexadecimal).

2. How are the users from question #1 going to be using this software? What is this software’s general purpose?

**DA:** To replace the DOS software currently used in CS 310 as the primary means to write assembly code.

**MB:** This will be replacing the DOSBox version (CAS). This will be *the* version used in CS 310 (and CS 311).

**CB:** The software is going to replace the current software that is being used.

**CF:** Replacing 32-bit DOS version. Intended use is teaching tool for use on class assignments.

**AH:** To replace ASSIST/I (CAS). Will be the software they use to learn assembly for IBM/360.

**TH:** Use in place of old environment of ASSIST/I *fully*.

3. What specific deliverables are required? By what time(s) and date(s) are they to be delivered?

**DA:** April 29, 2014 is when to deliver both documents and presentation to both clients and guests, who will critique. A lot of documentation is to be delivered. A way for client to present the software to students.

**MB:** April 29, 2014. Deliver both documents and presentation to client and guests. Deliver printed documentation: Source code, emails, meeting minutes, designs, software requirements specification document, client questions and answers, user’s manual digital and hardcopy, a way for the client to efficiently and appropriately present to students. Also include test files, test indices, and test reports.

**CB:** Deliver all documents, source code, user’s manual, and final product by April 29th.

**CF:** Delivered on April 29th (documents/presentation). Printed documentation (emails, minutes, questions, user’s manual, software, etc.).

**AH:** Delivered on 4/29 (presentation on same day). Printed documentation: Source code, emails, spec. doc., questions, user’s manual digital and hardcopy, way to give to students, testing documentation.

**TH:** April 29th: Documents and presentation to all present. What: Source code, email, minutes, design, spec. doc. questions, user’s manual, way to disperse among students, testing.

4. Would you like the ability to set and run to breakpoints when debugging programs? Or would you rather the debugger be forced to step over each line, starting from the beginning?

**DA:** Breakpoints shelved to the cake-icing section.

**MB:** Breakpoints are on wish list. Replicate what was originally done in ASSIST/I.

**CB:** Breakpoints would be nice if time allows.

**CF:** Mimic original software debugging methods.

**AH:** Step through. Breakpoints would be nice but not required.

**TH:** Optional breakpoints; required: replicate what ASSIST/I does.

5. What specific debugger functionality is required?

**DA:** See #4.

**MB:** See #4.

**CB:** [no response]

**CF:** See #4.

**AH:** Step.

**TH:** See #4.

6. Should the text editor force characters to uppercase, or should it allow lowercase as well?

**DA:** Force uppercase.

**MB:** Always uppercase characters.

**CB:** Force to uppercase.

**CF:** Force uppercase.

**AH:** Uppercase only.

**TH:** Force uppercase all the time.

7. Would you rather the software run quickly with less memory or run slowly with more memory?

**DA:** No problem with limits, but document them. Professor answer: Check past programs and use a static number.

**MB:** No problem with limitations, but *must* document them for users. Study old programs to get a general idea of limits for static structures (more efficient execution).

**CB:** Limiting memory is okay. Just remember to document all changes.

**CF:** Design based, but if placing limitations, they must be specified. (Hybrid focused on past programs used in CS 310).

**AH:** Document limits for users if we have a maximum size.

**TH:** (Developer’s choice. Question for professor.) May put limits on items, but must specify for users. Look at past programs (can use static limit).

8. What is an acceptable or tolerable rate of error or failure?

**DA:** No crashes not due to user error.

**MB:** *No* crashes allowed. Graceful “shutdowns” are tolerable.

**CB:** Crashes are not tolerated.

**CF:** No crashes acceptable. “Graceful” escapes allowed.

**AH:** Never.

**TH:** No unhandled errors/crashes.

9. How would you prefer the software be licensed? Should rights be relinquished once development has concluded? Should the software be completely free and open source for anyone to modify and maintain upon deliverance?

**DA:** Not open source, but controlled by department.

**MB:** Not free, open source software. CSIS Department will retain rights.

**CB:** The department should retain the rights.

**CF:** License is given to the client/department.

**AH:** Not open source. Owned by department.

**TH:** Allow license to the department.

10. Do you have any “security” concerns for this software? If so, could you detail them?

**DA:** No, no logins/passwords, etc.

**MB:** No.

**CB:** No.

**CF:** No.

**AH:** No.

**TH:** No.

11. What is the maintenance plan for this software? Will the current team be relinquished of maintenance responsibilities upon deliverance of the final software?

**DA:** We are relinquished, but may be asked later for help by a later class, but not required.

**MB:** Will *not* be responsible but may be contacted by future teams for courteous consultations.

**CB:** The team will be relinquished of all responsibilities.

**CF:** Responsibility of maintenance is waived upon delivery. (Potential callback to assist future development).

**AH:** Not responsible but possibly needed to help someone else extend.

**TH:** No.

12. How do you want the software to be delivered to users? Should it be portable, packaged in an installer file, or should there be options for either option?

**DA:** Leave open—may need an installer, but may not. Must be as portable as possible.

**MB:** Portable execution is preferred for ease of delivery (i.e., run from a flash drive). This requirement may change as development progresses.

**CB:** Executable is preferred, but that is subject to change.

**CF:** Ease is important for user. Favor portability. (Open question, may redact.).

**AH:** Executable is preferred. Still open.

**TH:** Executable file is preferred, but may be an installer if needed. Formally determined after further design.

13. According to the ASSIST/I User Manual (p. 496), the XDUMP instruction was “intentionally left out of ASSIST/I because of the interactive debugger.” Should XDUMP be included in ASSIST/UNA? Note that ASSIST/UNA will also have “interactive debugger” capabilities.

**DA:** Leave it off since we display registers and other content.

**MB:** Leave out XDUMP in favor of graphical displays of registers and memory contents.

**CB:** Omit the XDUMP instruction.

**CF:** XDUMP is omitted.

**AH:** Can leave off.

**TH:** Exclude XDUMP because of memory/register display.

14. There are the instructions N, NR, O, and OR listed on the ASSIST instructions to be implemented. However, we are unfamiliar with these instructions. Could you elaborate on their type (e.g., RR), function, and format (i.e., parameters)?

**DA:** AND: N (RX), NR (RR). OR: O (RX), OR (RR).

**MB:** Bitwise AND: N (RX), NR (RR). Bitwise OR: O(RX), OR (RR). Note that the RX versions must have the second operand on a fullword boundary.

**CB:** AND: N (RX), NR (RR). OR: O (RX), OR (RR).

**CF:** AND: N (RX), NR (RR). OR: O (RX), OR (RR).

**AH:** AND: N (RX), NR (RR). OR: O (RX), OR (RR).

**TH:** AND: N (RX), NR (RR). OR: O (RX), OR (RR).

15. How would you like users to be trained to use the ASSIST/UNA software?

**DA:** User’s manual and online help. Client will give training.

**MB:** Presented with digital user’s manual and online user help. Instructor will provide training sessions.

**CB:** Client will train the users.

**CF:** Handled by client/manual.

**AH:** User’s manual.

**TH:** Instructor/user’s manual.

**NR and OR Examples:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Register 2** | **Register 7** | **Operation** | **Register 2** | **Register 7** |
| 0011 | 0110 | NR 2,7 | 0010 | 0110 |
|  |  |  |  |  |
| **Register 2** | **Register 7** | **Operation** | **Register 2** | **Register 7** |
| 0011 | 0110 | OR 2,7 | 0111 | 0110 |