**Meeting Minutes**

**Date:** February 4, 2014

**Start Time:** 7:10pm

**End Time:** 8:20pm

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

Andrew Hamilton, Travis Hunt, and Dr. Patricia Roden

**Members Absent:** N/A

**Topics** **Discussed**

* Client Questions #3
* Choice of programming language
* Preliminary test cases
* Prototype presentation and critique

**Decisions and Actions Taken**

The team consulted with the client. The client responded to the team’s third set of questions. The client’s responses are attached below. The team has yet to identify *five* *major* questions to ask the client. The team needs to step back and reevaluate the current state of the project and the project requirements.

The team needs to consider how students are going to acquire the software, how they are going to interact with the software, and how the software is going to be delivered to the client (and students). The team needs to ask the client about programming languages and be near ready to formally propose a programming language. The team will officially propose its chosen programming language at the next client meeting. Initial test cases should be ready for presentation by February 10, 2014.

The team presented its initial prototype to the client. The team needs to propose a list of syntax highlighting schemes for approval by the client. The team also needs to propose color schemes for the actual windows forms to delineate sections of the user interface. The colors need to have better contrast. The Program Status Word and the Condition Code need to be reorganized in some way (i.e., PSW not wrapped and CC not in an independent field). Also, note that the CC is only two bits and is part of the PSW. The system needs to have options for printing the .PRT in addition to printing the program listing (i.e., the source code itself). The program listing can be printed in portrait mode, whereas the .PRT would ideally be printed in landscape mode.

The specification document needs to detail which deliverables will be physical and which will be digital and how those deliverables will actually be delivered.

The delete and insert line functionality in the user interface may be better serviced by a text field rather than a textbox. Text fields maintain spaces for each character by default (e.g., 80 blank spaces per line), whereas each row of a multiline textbox does not maintain spaces for each character *until* a character is entered.

The team will determine the next meeting time and location after the next class meeting.

**Supplementary Information**

**Client responses to questions:**

1. Do you want assemble-time errors to be handled by error detection or by error avoidance?

**Drew Aaron:** Do it just like ASSIST/I; find multiple errors like ASSIST/I.

**Michael Beaver:** Simulate ASSIST/I *exactly* (very rigid). Assemblers assemble in a WYSIWYG (What You See Is What You Get) manner.

**Clay Boren:** Neither.

**Chad Farley:** Absolutely not!

**Andrew Hamilton:** No.

**Travis Hunt:** No.

2. Should error detection be implemented as ASSIST/I implements it?

**DA:** See #1.

**MB:** ASSIST/I will report multiple errors. The emulator should work *exactly* like ASSIST/I.

**CB:** Yes.

**CF:** Just like ASSIST/I.

**AH:** Yes, detect all errors.

**TH:** Yes, find multiple.

3. Suppose a user wishes to modify the maximum size of available memory. What is the upper bound on memory size you would like? (ASSIST/I bounds the memory to 9999 bytes.)

**DA:** Yes.

**MB:** 9999 bytes.

**CB:** 9999 bytes.

**CF:** Yes.

**AH:** Yes.

**TH:** Yes, 9999 bytes.

4. Do you want to support the use of the asterisk to refer to the location counter (see: Andrew)?

**DA:** Only if we have time. Shelve it.

**MB:** Not required at this time (wish list).

**CB:** Not required.

**CF:** Optional (most likely not).

**AH:** Icing on the cake.

**TH:** Optional.

5. The IBM /360 has a few floating-point registers. Would you like us to emulate these as well? If so, to what degree should they be implemented?

**DA:** Same as #4. The client likes it, but it is not necessary (wish list).

**MB:** Not supported by ASSIST/I. Floating-point registers would be nice, but they are not required at this time (wish list).

**CB:** Would be nice to have.

**CF:** Optional (most likely not).

**AH:** Icing on the cake.

**TH:** Optional.

6. ASSIST/I is a two-pass assembler. Would you like the emulator to follow this standard?

**DA:** Two pass. Must be!

**MB:** *Must* be a two-pass assembler. The nature of the IBM/360 assembly language dictates it.

**CB:** Yes.

**CF:** Must be like ASSIST/I.

**AH:** Yes, DS and DC are at the bottom. One pass is impossible.

**TH:** *Must* be two-pass.

7. The ASSIST/I .PRT files display and print better in landscape mode. Would you prefer .PRT files to be displayed in landscape mode, portrait mode, or to have options for both?

**DA:** If it is feasible (open-ended). Review later. Document that it is better in landscape and why we chose what we did.

**MB:** If feasible. Report later when feasibility of viewing styles is better known. Would love landscape, but go with what works best and document that decision.

**CB:** Use what works best.

**CF:** If possible, we should default to landscape (left open-ended by the client). *Design option*.

**AH:** Find out later. *Design option*.

**TH:** Open-ended. *Design option*.

8. Would you like the ASSIST/I assembler options to be project-specific or global (see: Chad)?

**DA:** If feasible. Explain later.

**MB:** Explain later how project-specific options will be saved (see Jan. 31 minutes). The proposed format will be approved or disapproved by the client.

**CB:** [no response]

**CF:** Open-ended (most likely speed related).

**AH:** Maybe (depends on how we do it).

**TH:** (Ask about file structure as a whole?)

9. As a follow-up to #8, would you like the ability to save project settings, options settings, and other relevant project information to a “project” file (e.g., myProject.una)?

**DA:** Would have to see if the memory for speed tradeoff is worth it.

**MB:** What overhead is involved with regard to memory and speed? Report later.

**CB:** Optional.

**CF:** See #8.

**AH:** Maybe. Optional.

**TH:** Possibly. Optional.

10. As a follow-up to #8, what are your thoughts on the extension \*.una for project files?

**DA:** Good.

**MB:** See #9. The .una extension would be good *if* project files are used.

**CB:** Optional.

**CF:** Yes, *if* #8 is implemented.

**AH:** Yes.

**TH:** *If* there are project files, then yes.

11. As a follow-up to #8, if you prefer global options, would you like them to persist between sessions (i.e., options settings saved to a configuration file)?

**DA:** If feasible. Explain later.

**MB:** See #8.

**CB:** Optional.

**CF:** Inconclusive.

**AH:** Optional.

**TH:** See #8-10.

12. What extension would you like for the source file? Perhaps the files could use the .TXT or the .UAS (UNA Assembly Source) extension?

**DA:** Unique, like .una or .uas. The .una extension is preferred if there is no project file.

**MB:** Would like a unique file extension. The .uas extension is fine *if* the .una extension is *not* used for project files.

**CB:** Use a unique extension.

**CF:** Use the .una extension if #8 is not implemented; otherwise, use the .uas extension.

**AH:** Preferably a unique extension, like .una or .uas extensions.

**TH:** If not using .una for above, then use .una extension. Else, use the .uas extension.

13. Would you like the user to be notified in the event of a run-time error? If so, then how?

**DA:** Yes, in a more detailed way than just “crashed,” though.

**MB:** Present these in an error message in the output (i.e., XPRNT) dialog.

**CB:** Yes.

**CF:** Yes.

**AH:** Put in dialog box.

**TH:** Yes, in popup XPRNT dialog.

14. Memory dumps from crashes are to be saved to the .PRT file. How would you like memory dumps from XDUMP to be displayed to the user?

**DA:** Dialog.

**MB:** XDUMP dumps are presented in the output dialog (see #13).

**CB:** [no response]

**CF:** Yes.

**AH:** In dialog.

**TH:** To dialog.

15. In ASSIST/I, the debugging (“Run”) mode saves a .PRT file after execution. Would you like the emulator’s “Assemble and Debug” run option to save a .PRT file or not?

**DA:** No.

**MB:** Do *not* save a .PRT file when debugging.

**CB:** No.

**CF:** Not.

**AH:** No.

**TH:** *No .PRT file*.

16. The project description lists the option of “exiting from a file without saving the file.” Would you like the user to be prompted to save the file before exiting, like in modern IDEs?

**DA:** Yes.

**MB:** Yes. Use Ctrl+KQ in ASSIST/I.

**CB:** Keep the same as ASSIST/I.

**CF:** Ask if the user is sure about exiting.

**AH:** Yes.

**TH:** *Yes*.

17. In the project description there are features “Delete line(s)” and “Insert lines(s).” Are these necessary, separate actions, or could these not be achieved simply by using backspace / delete and the return key? If these are necessary actions, could you elaborate on how you envision them being implemented and functioning?

**DA:** Shortcut to delete the entire line that the cursor is sitting on. Enter key is sufficient for inserting lines.

**MB:** Shortcut to delete a line on which the cursor is currently located. Refer to the ASSIST/I User Manual for further details. The “enter” key is sufficient for inserting new lines.

**CB:** See page 488.

**CF:** Shortcut to delete line at cursor. *Note*: consider using text fields instead of textboxes.

**AH:** Shortcut to delete line. See p. 488 in the ASSIST/I User Manual.

**TH:** Shortcut to delete the line with the cursor. See p. 488 in the ASSIST/I User Manual.

18. Do you want the emulator to track and report warnings and errors? If so, should the emulator be able to handle *all* possible ASSIST/I warnings and errors or a relevant subset?

**DA:** Yes, all warnings and errors. May be provided with a list of some sort at a later date.

**MB:** Report errors and warnings as ASSIST/I reports them. Handle *all* errors and warnings. A list of errors and warnings may be made available by the client.

**CB:** All.

**CF:** Yes to all.

**AH:** Yes. The client will provide more information.

**TH:** *Yes*.

19. Would you like the option to print .PRT files from the IDE?

**DA:** Yes, default to print in landscape.

**MB:** Yes, in addition to the option of viewing the .PRT file. Default print to landscape.

**CB:** Yes; print to landscape.

**CF:** Yes, buttons on toolbar for print *and* open .PRT.

**AH:** Yes, landscape.

**TH:** *Yes* (landscape).

20. Could you elaborate on the “configuration of assembler (specify path and arguments)” requirement?

**DA:** Is everything going to have to be in the same [working] directory, like with ASSIST/I? Prefer to not have in the same folder, unless infeasible.

**MB:** Can the user specify paths, or do *all* documents have to be in the same working directory? Prefer *no* path restrictions (i.e., not just one working directory). Example: $ENTRY FILE

**CB:** Ability to specify directories.

**CF:** Allow specification from user for saving and loading files.

**AH:** Input files from one directory. Get rid of file path restriction.

**TH:** Separation of Assembler to project location and $ENTRY files.

21. The ASSIST/I editor allows for 79 characters to be entered per row. Should the editor allow 79 characters to be entered per row?

**DA:** Cut off at 80 characters, like ASSIST/I.

**MB:** Cut off rows at 80 characters.

**CB:** [no response]

**CF:** Yes, just like the emulator.

**AH:** Cut off at 80.

**TH:** Cut off at 80.

**Additional client meeting notes:**

* Take a step back. The team needs to address *at least* five major topics.
* Big picture items and implementation.
* How are students going to obtain and interact with the software?

**Highlights from Client Critique of Initial Prototype:**

* The user interface needs a color scheme that “pops” more. There needs to be clear delineations between sections of the interface.
* The option for user-defined syntax colors is to be shelved at this time.
* The memory representation on the user interface is good.
* The overall design is acceptable.
* Get rid of the Condition Code field.
* Unwrap the Program Status Word field and stretch it horizontally.
* If possible, highlight the Condition Code *within* the Program Status Word field.
* The user needs options to print the source code, in addition to the .PRT print option.
* The .PRT file will be printed in landscape mode, but the source code should be printed in portrait mode.