Inter-Office Memorandum

To	Mesa Implementors	Date	October 5, 1978
From	Barbara Koalkin	Location	Palo Alto
Subject	Proposal for Debugger Interface	Organization	SDD/SD

XEROX

XEROX SDD ARCHIVES
I have read and understood
Pages_________To_________
Reviewer_________Date_______
# of Pages_____Ref. 78SDD-190

Filed on: [IRIS] < KOALKIN > DUI.BRAVO

DRAFT

It is clear that more extensive use of windows/menus/selections would improve the debugger's user interface. This memo is a proposal for a new user interface for the Mesa 5.0 debugger; it has been drawn together from suggestions from the Mesa task list, Greg Shaw's suggestion list (based on the experience of ETM), the change requests classified as debugger wishes, experiments with the Tools Environment and the SmallTalk Browser, and discussion with experienced Mesa users.

Based on the following observations:

- We are aiming for the experienced systems programmer, *not* a clerical person with minimal training who will quit in 6 months.
- We must build on the capabilities (and code) of the current Mesa debugger as well as maintaining the present Mesa style of debugging.
- Speeding up the performance and increasing reliability is as important as improving the interface.

We should implement the following new facilities:

- A way of looking at the stack as a list of procedures and modules, where selecting the name shows its local and global variables and source text. An optimization that could be made is to only show the value of user selected variables and monitor their values each time you enter the debugger.
- A more visual way of changing contexts (configuration, process, and module) so that by selecting the name, it becomes the current context.
- Better use of selections to cut down on type-in. Show the value of a variable simply by selecting it; change the current context by selecting the name of the new context.
- More extensive use of the debugger's source file window. You can keep a marker in the source text for each of the breakpoints that have already been set, and have a command to show where breakpoints are allowed to be set.
- A macro/replay facility for replaying a session or repeating a series of commands.

Old things that need to change:

- Replace the Mesa.Typescript window with a stack/configuration window. If you want to see the typescript, you can use the Userscreen command or load the file into a scratch window.
- Fully integrate the window manager and interpreter into the debugger. Make interpeted expressions and selections valid input for all debugger commands.

* Expand the present selection scheme. Double clicks can expand to the next enclosing Mesa structure.

* Some sort of "property sheet" for infrequent, globally defined state. This would allow users to change the default settings on globals such as paying attention to case shifts in variable lookup, entering the debugger in worry mode, setting the default input radix, and whether input may come from the keyset.

* All windows should have a menu with the same standard set of window operations (move, grow, create, destroy, top, bottom); however, different types of windows need other more specific actions. For instance, the Debug.Typescript window does not need breakpoint capabilities.

Goals to keep in mind:

* Try to cut down on the edit-compile-debug loop. This has been helped by having the FTP and Chat facility in the debugger. We should also look into integrating the debugger into the Tools Environment and adopting the standard Tools User Interface to get the benefits of a simple editor and standard user interface facilities.

* Keep it simple. No on: too many windows, slow graphics, too many keystrokes, and the display flashing all the time.

Questions to look at:

* Should we give up on the typescript as an absolute log of the debugging session when we go to a more selection oriented, less teletype oriented, interface? It would affect the replay capability if all actions were not completely recorded someplace.

* What is the overhead involved in integrating into the Tools Environment and adopting the standard Tools User Interface? This would give us the benefits of using a standard accepted interface but introduces problems of maintenance and release control.

* Which menu style to adopt? The possibilities include: one big menu for all windows (as in the present Mesa debugger), two or more menus per window (as in the Tools Environment), different menus for different types of windows (as in SmallTalk), or fixed menus (as in Laurel).

* Should we implement an on-line help facility to aid our inexperienced users and infrequently used commands? For example, prompting for input parameters for commands and giving more complete error messages from the interpreter.