Self 4.1.5 Release Notes

Michael Abd-El-Malek and David Ungar

1 Introduction

The Self project has not been active at Sun since 1995. Although some of its innovations (especially those concerned with implementation technology) have found their way into the world, other Self inventions such as the language design, the user-interface, the programming environment and the transporter are still not yet widely known. In order to make it possible for more people to experiment with these ideas, and to support our own work, we have ported Self to Mac OS X. Kristen McIntyre helped with the initial port. This system is far from polished, consider it a beta version. In particular, it includes none of the clever implementation tricks that gave Self its advanced performance on the SPARC™ processor, however it runs acceptably on a 500 Mhz G3. And, though we have used it to do many hours of work on Powerbooks, it is a bit flakier than we would like. However, we are releasing it so that friendly and determined users can experiment with the system, and so that Self devotees can exploit the portability framework and VM cleanup work we have done. Following are some brief explanatory notes. Send questions to `david.ungar@sun.com`. You may also want to try sending mail to the Self interest mailing list, `self-interest@egroups.com`.

Now it is Winter 2002, and we are putting out 4.1.5. We wish we had more time to file off the rough edges, but believe there is enough value added since 4.1.4 to put out another release.

2 Enhancements to Self 4.0

2.1 Mac Compatibility

Since Self 4.1.2, Dave has ported the virtual machine and ui2 to Apple's Carbon compatibility library. Previous releases of Self (until 4.1.4) ran on OS 9 and OS X. In this release, we have dropped support for OS 9. You need OS X if you want to use Self on a Mac.

2.2 Portability Framework

Since Self 4.0, Dave has added a portability framework to the virtual machine and partially ported it to Mac OS X and the PowerPC architecture. The port is only partial because it only implements enough to run Self on the Mac, not enough to realize the full performance of the Self VM. In particular, Dave ported the memory system, OS interface, runtime system and the non-inlining (simple) compiler (the NIC), but not the polymorphic inline caches (PICs) nor the simple inlining

(smart) compiler (the SIC). Also ported is the frame conversion code needed to create debugging frames from unoptimized frames, but not the frame conversion code used to create optimized frames from unoptimized frames, nor the code that creates unoptimized frames from optimized frames.

With the portability framework in place, it should be easier for others to port the virtual machine. The portability framework consists of additions to `makeDeps`, a tool that manages C++ header files, and of idioms employed when writing partially- or fully- machine-dependent classes.

2.2.1 `makeDeps`

See the included Self 4.1.4 Release Notes document for information on `makeDeps`.

2.3 UI2 on Mac

Although UI2 was originally designed for a three-button mouse, the Mac has just a one-button mouse. So, option-click on the Mac is used for a middle-button click, and command-click is used for a right-button-click. To change these, find the whichButton: method in the initialization category in the traits ui2MacEvent object.

In OS X, the UI2 can use Carbon or X, if you have an X server. In the Self console, a `desktop open` will open a Carbon window by default. `desktop open` calls `desktop openOnDisplay:` with an empty display string. If you call `desktop openOnDisplay:` with a non-empty display string, then it will be assumed you want an X window. Thus, to force the use of X as opposed to Carbon, use the `desktop openOnDisplay:` message with a non-empty string specifying the display.

3 Enhancements and changes to Self 4.1.5

Since Self 4.1.4:

Platform-independent changes:

- The demo snapshot has been recreated. It now reflects the most recent Self system.
- The Self debugger has been generalized to be retargetable to debug other things (for example, machine-level debugging).
- The transporter has been enhanced to handle object vectors that aren’t copies of the prototypical object vector.

- We have implemented an experimental model for slots that hold methods. Under the traditional semantics, a method may only be stored in a constant slot, and the method is invoked whenever the slot is accessed. By executing `_NakedMethods: true`, the user can now select a relaxed model, which permits a method to be stored in an assignable slot. Accessing such a slot merely returns a reference to the method object.
- Added an argument count bytecode. A new instruction set has been defined and there is backward compatibility with the old instruction set. We will be placing information on this bytecode, along with the other bytecodes, on our web site soon (http://research.sun.com/self).
- `bigInt` now uses a binary base rather than a decimal base, so we can do bitwise operations.
- Support for handling 32 bit binary integers has been increased (see the `int32` and `int64` objects).
- Variations for transparent forwarding (useful for debugging) have been added (see `loggingSender`).
- `traits smallInt` has two extra methods: `numberOfOnes` and `roundUpTo:`.
- Fixed some memory leak bugs that would cause the system to slowly get bigger and bigger. (As a side effect, we have increased the time required to do a thorough memory cleanup.)
- Added support for the `ptrace` system call.
- Added a “Yank Out Outliner” menu item which lets the user create smaller views of the same object. (Warning: there are still some rough edges here.)
- Fixed bugs for frame buffers without color maps.

Solaris/SPARC™ changes:

- We wrote a new section explaining how to build the Self VM on the Solaris environment. Please see section 4.2.
- An inlining bug in SIC has been fixed.

Mac OS X changes:

- The Self VM is now an Apple Project Builder (PB) project. We provide the PB project, under vm/mac_osx/vm_project. Please see section 4.3 for instructions on building and running Self on Mac OS X.
- An AppleScript droplet has been included that facilitates starting up Self. Now, Self snapshots can be double-clicked to launch Self or they can be dropped on this droplet. Please see section 5 for information on how to start up Self on Mac OS X.

- The creator type and file type of snapshots are now set correctly (to Self and Snap, respectively).
- Minor performance enhancements.
- Aligned memory allocation doesn't use `mmap` anymore. We use `malloc` instead.
- Fixed a semaphore bug that caused the system to freeze during garbage collection.
- Added support for cursor teleportation. Thanks to Kristen McIntyre.

Since Self 4.1.3:

- The slice outliner now provides more flexible senders, implementors, etc. views of your objects.
- Several memory leaks and Macintosh performance bugs have been fixed.
- The abstract syntax module had been deleted.
- The outliner has been refactored into a pluggable outliner hierarchy and a model hierarchy. This change would facilitate supporting other languages with the Self programming environment.
- When changing a method in the debugger, if the method is present in more than one object, the debugger will ask what you want to do rather than changing it everywhere as before.
- Activation mirror objects now also contain a reference to the relevant process, simplifying their usage.
- Browsing implementors, senders, etc. is now more versatile: slice outliners have been added. These can browse hierarchically, restricting the search to an object's ancestors, descendants, or both (its family). When used in such a fashion, the results are displayed in a visual containment hierarchy corresponding to the objects' inheritance relationships.
- Some of the methods that were in outliner objects, but were specific to modifying Self objects have been relocated to mirror and slot objects.
- Outliners how have a "yank" menu button. This operation allows you to extract a view on just a single method or category, like "spawn" in Smalltalk-80.
- I have written a general parsing framework, and written a Java and Self parser in it. (As of this writing (8/5/2001), I'm not sure whether the parser will make it out the door.)
- A slot in the prototype module object, "comment", has been renamed to "myComment" in order to support robust interactive editing of the comment. (The module outliner has been revamped and now supports comment display and editing.) If you have any Self source files of your own, you should use a text editor to rename the "comment" slot in the module before reading them in.

- In order to recategorize a slot in a given object, you should now “copy” the slot and drop the copy into the new category. The system will notice that the slot is merely being recategorize and not ask for confirmation.
- A cache is now maintained of those modules that are not contained in any other modules, that is the “top-level” modules.
- Several bugs have been fixed, including one that made it impossible to single-step a thread under certain circumstances.
- OS X saves moderately-sized snapshots so quickly that I have removed the garbage collection operation before a snapshot. You can always select “clean up memory” from the background menu.The virtual machine file name suffixes have been changed to be more compatible with modern C++ compilers.
- Some of the icons have been replaced with much nicer ones (thanks to Kristen McIntyre).

Since Self 4.1.2

- Self now runs on OS X.
- A bug in the SPARC spy has been fixed.

4 Self VM Build Instructions

This section only applies to people interested in building the Self VM. Many users will not need to do this; they can use the provided Self VM binaries. Those interested in working with the VM will need to know how to build the VM, and this section documents this process.

4.1 Common

1. You need a top-level Self directory. In your shell’s start-up script, set the environment variable `SELF_WORKING_DIR` to be this directory. As an example, we use `~/self`. If you do not use a C-shell, you must still set the `SELF_WORKING_DIR` variable in your `~/.cshrc`. This is necessary since the build-time scripts are C-shell scripts and they depend on the `SELF_WORKING_DIR` variable. So as an example, if you use the bash shell, and you set `SELF_WORKING_DIR` in your `~/.bashrc`, you still need to set `SELF_WORKING_DIR` in your `~/.cshrc`.
2. Download the Working_with_VM package.
3. `cd ${SELF_WORKING_DIR}`
4. `tar -zxvf <path to archive>/Working_with_VM.tgz`
5. `mkdir objects # in case it doesn’t exist`
6. `rm -rf vm     # in case it exists`
7. `mv put_contents_in_objects_dir/* objects/`
8. `mv put_contents_in_Self_dir/* .`
9. `rmdir put_contents_in_objects_dir`
10. `rmdir put_contents_in_Self_dir`

11. You can now delete the Working_with_VM package.

4.2 Solaris/SPARC™

First, follow the instructions in section 4.1.

Next, generate the dependency lists as follows:

1. `cd ${SELF_WORKING_DIR}/vm/svr4/generated`
2. `make lists`

The above step has to be done any time you add or delete a file to the project.

Now, you can start the regular build process:

1. `cd ${SELF_WORKING_DIR}/vm/svr4/optimized`
2. `make`
3. On a multi-processor machine, you can use parallel builds by issuing the command `make files` for the second and subsequent build jobs.

In the above instructions, I went into the optimized directory and this caused the resultant build to be an optimized build (which still included debugging information). If you want to a debug or profiled build, go into the `debug` or `profiled directories` and issue the make command there.

4.3 Mac OS X

First, follow the instructions in section 4.1

Although Self uses Carbon, you still need two X libraries. These libraries are needed to allow Self to share windows with other X clients. Also, you can use X windows, rather than Carbon windows, under OS X (see section 2.3). The two libraries you need are libX11.a and libXext.a (notice these are the static versions). We use the libraries provided by the 4.2 release of XFree86 for Mac OS X (known as XDarwin). This package is freely downloadable at http://www.apple.com/downloads/macosx/unix_apps_utilities/xfree86.html. Our build process expects libX11.a and libXext.a to be located in /usr/X11R6/bin.

Building the Self VM under Mac OS X is now done almost completely in Project Builder. However, one thing has to be done in a console first.

1. `cd ${SELF_WORKING_DIR}/vm/mac_osx/generated`
2. `make lists`

The above step has to be done any time you add or delete a file to the project.

Now, you can start up Apple's Project Builder. Open the Self project file, which is in `${SELF_WORKING_DIR}/vm/mac_osx/vm_project`. Under the Project menu, select the Edit Active Target. Go to the Build Settings tab. Scroll down to the bottom until you see the Build Settings. Here, set the `SELF_WORKING_DIR` variable to correspond to the `SELF_WORKING_DIR` variable you set earlier.

The default build style is Development, which is somewhat slow. To build a much faster Self VM, click on the Targets tab in the left panel, and select the Deployment build style.

Cick the Build button to start the build.

Unless you tell Project Builder otherwise, the result of the build will be placed in `${SELF_WORKING_DIR}/vm/mac_osx/vm_project/build/Self.app`.

Note: Project Builder's recursive dependency analysis is broken (this is based on Project Builder version 1.1.1). For example, if `foo.c` includes `bar.h` and `bar.h` includes `fubar.h`, changing `fubar.h` may not necessarily cause `foo.c` to be rebuilt. As a result, changing a header file in Self may not lead to the correct source files being recompiled. We hope Apple fixes this significant bug soon.

Please see section 5 for instructions on how to launch the Self VM under OS X.

5 Running Self in Mac OS X

There are two ways to run Self in Mac OS X. You can either start Self from the command line or you can use the Self droplet we are providing.

5.1 Running Self from the Command Line

The Self VM binary is located in `${SELF_WORKING_DIR}/vm/mac_osx/vm_project/build/Self.app/Contents/MacOS/Self`. To run Self, just use:

```
<path to Self>/Self
```

or, to start up a snapshot:

```
<path to Self>/Self -s <path to a snapshot>
```

5.2 Using the Self droplet

The Mac OS X version of the Using_Self package includes an AppleScript application that simplifies starting up Self. You can drop a Self snapshot over the SelfDroplet application and this will create a new Terminal window which starts the Self application.

You can also associate Self snapshots with the droplet, so that double-clicking a snapshot will start it up. To do that, bring up a Show Info on any of the provided snapshots. Then in the Open With Application section, select the SelfDroplet application in `${SELF_WORKING_DIR}/objects`.

You should use the provided snapshots since they have their file type and creator code set correctly, and older snapshots may not. If you create new snapshots, these two properties are set correctly under Mac OS X.

6 Limitations

The Self Mac port does not include any of the performance tricks that Self is known for: there is no optimizing compiler and no polymorphic inline caching. I have used a 266Mhz G3 and found the system usable if the desktop was kept uncluttered. Some operations (such as implementors of a popular selector) were pretty slow. On the newer 400Mhz G3, it feels pretty spiffy most of the time.

7 Bugs

7.1 A few really upsetting bugs are left:

Sometimes when starting up on OS X, the Self UI2 window remains in the background and there is no way to activate it. I know of no other solution than to restart the machine.

The outliner work has destabilized things a bit; sometimes the module dictionary keeps refilling over and over again. When this happens, I recommend that you kill the scheduler (^C followed by “q”), restart the prompt (“prompt start”), and force a refill of the module dictionary (“moduleDictionary alwaysRefill”).

Also thanks to the outliner work, once in a while a debugger will appear when moving a slot.

7.2 The following minor bugs exist:

In UI2, when dragging a non-rectangular object (e.g. the trash can) the shadow does not conform to the object’s shape.

Once in a while, using the spy causes Self to freeze up. When that happens, you have to quit Self and start it again.

If you change a module and attempt to file it out, the transporter assumes that the application is located in a folder called `objects` and assumes that the subfolders that originally contained the source file are present.

7.3 Finally, there are several tasks yet to be done that are on my wish list:

UI1 is a user-interface that was built to show how cartoon animation techniques could enhance the user's experience. It is partly ported to Quickdraw, but much remains to be done.

When starting a snapshot on the Mac, if the screen size has shrunk, UI2 should ensure that your windows do not open outside the bounds of the new screen.

Although you can double-click a self source file such as `all2.self` to start Self and read the file, it does not work if Self is already running.

Porting UI2 to Quartz, the native graphics layer for OS X.

Finally, the icons were sketched very quickly and could really use a cleanup.

8 Conclusion

As you have read, this release embodies a lot of work with a lot of rough edges. We hope it works for you.