Stanford Exploration Project, Report 70, December 18, 1997, pages 1–392

Electronic book update

Jon F. Claerbout[1]

keywords: *algorithm, interactive, graphics, work stations*

ABSTRACT

I am writing a new textbook (and revising an old one) that will appear in electronic form as well as paper form. On the workstation screen appear pages with pasted-in figures and push buttons in the figure captions. A pushbutton takes you to a directory where the figure gets made. From there, some figures spring up into interactive programs and others into batch programs. Reading is simplest when remote sites use networks and X to read my Stanford copy of the book. I am also attempting to distribute the book on tape but the main drawback there is the volume of complicated software required to be installed, including cake, ratfor, seplib, vplot, TEX and LATEX, XView, and the X window system.

SYNOPSIS

In report SEP-67 I wrote a paper entitled, "Active documents and reproducible results" that claims a revolution in education and technology transfer will follow from the merging of word processing and software command scripts. In this merging, called an *active document* (a-doc), the author attaches to every figure caption a pushbutton or a name tag usable to recalculate the figure from all its data, parameters, and programs. At my present stage, when plot files are absent or out-of-date the a-doc software regenerates them. So I see an a-doc as software that reproduces a document including the computations underlying the results and the conversion of those results to figures in the document. An a-doc provides a concrete definition of reproducibility in computationally oriented research. Given suitable interactive software, an *active* document is easily converted into an *interactive* document (i-doc) and we have begun some experimental work on i-docs.

The main lesson learned before SEP-67 was that authors and researchers should keep a meticulous accounting of how each figure is made, so that when the time comes, the figure regeneration procedure can be merged with the figure caption information in the word-processor software. Technically, the figure regeneration procedure we use

[1]**email:** not available

is like the well-known "makefile" method of building UNIX system software.

CURRENT STATUS

I am writing a new textbook entitled, "EARTH SOUNDINGS ANALYSIS: Processing versus Inversion" (PVI) and I am revitalizing my 1985 textbook, "Imaging the Earth's Interior"(IEI). Both books can now be viewed on a workstation screen, including all the figures. The new book should be publishable by the end of the year.

Nearly all the figures in PVI have underlying program support. Of 140 figures, 40 are supported by interactive programs, and about 70 are supported by batch programs, most of which run in a matter of seconds. The few figures without program support are mostly drawings and indications of ideas rather than demonstrations of them.

The documents themselves are written in LaTeX which can be printed on a laser printer, or the page images can be displayed on a workstation using Dirk Grunwald's `xtex` (freely available via anonymous ftp from foobar.colorado.edu 128.138.243.105) that enables us to include pushbuttons in the manuscript. Steve Cole adapted this software to X on Sun workstations so I am delighted to be seeing bitmap pictures of pages on the screen of my workstation, including pasted up figures. With `xtex` we can place a pushbutton anywhere in the text (see Karrenbach and Nichols, 1990, page 150) and the pushbutton will launch any command in the system. This provides more flexibility than we can begin to exercise, so I began (with help of Martin) by having each figure caption provide a pointer to the figure building file.

All the figures in PVI now contain pushbuttons, which either launch the interactive programs in an appropriate initial state, or for the batch programs, check to see whether the plot file is up-to-date with the software that generates it; then if not, they recompute it; and finally they invoke the local plot display program which has color and zoom. For 23 figures, the `ed1D` interactive program [Claerbout, 1988] is invoked, and for 16 figures the `Zplane` program [Claerbout, 1987] is invoked. We have not fully defined the kind of "workbench" that should surround batch-program figures. Although we want to offer the reader some flexibility, it is not clear how much. Obviously, maximum flexibility comes from leaving the manuscript and logging in at the figure construction directory. The beginning of a possible "workbench" environment is suggested by Steve Cole's Xvpanel [Cole, 1990]; but more experimentation is needed before installing it in my books.

For the older book, IEI, the program support is in place for 30 of 170 figures For those 30, pushbuttons will be installed soon. The rest of the figures are in a raster scan format. About 30 more figures should get program support over the next year. Many of the scanned figures are sketches not warranting an underlying program. About 40 of these sketches are being being redrawn with a simple draw-type program to escape from the crudeness of scanning. Others contain hyperbolas or other elements that are beyond simple draw-type programs. Still other figures deserve program support, but providing it may be beyond my time and energy. The scanned figures now occupy about 100Mb, which is more than ten times everything else put together. Luckily, the `compress` program works well. Of the two books, IEI is more suitable for movies than PVI, but PVI is more suitable for interaction. With next quarter's teaching being from IEI, some figures should turn into movies.

I have devoted much of my time to cleaning up the books themselves and the underlying figure makefiles (actually cakefiles) and little of my time to designing ways of making the final materials more versatile and appealing to others. I hope to find a partner interested in the design of the "workbench".

Internal layout of documents

The chapters of the books have a unique mapping to computer directories. Inside any chapter directory is a file with the text of the chapter and also several topic directories most of which build figures. A figure directory may build a single figure or several. Each of about 70 figure directories declares:

1. that the figure is in the book PVI (or IEI),
2. the chapter name,
3. a list of figure names
4. commands that prepare those figures, and
5. that SEP definitions, compilation rules, and i-doc rules are to be included.

Besides the many figure directories are several directories that install executable programs which are used more globally in the book. Modern programming languages, especially C, facilitate the idea of variables being local, or global, or at intermediate stages. We are trying with partial success to segregate components of interactive documents in the same way. Logically, the subroutines described by a chapter should be files kept with the chapter, but at present they are gathered together for the whole book in a single library directory. When PVI is completed I will devote more energy to activating IEI and then the question will arise of how to handle parts that are common to the books. From a program maintenance view, there should never be duplication, which implies that a few chapters or appendices in each book should be identical or the books should be merged into a consistent whole.

We enhance the mechanisms of interactive documents in two ways, (1) in the LaTeX author's environment, and (2) in the figure building file (cakefile). The LaTeX environment invokes the figure building script (cakefile) but does so in only a standard way that we have chosen to be the command "Change directory to chapter X, figure subdirectory Y, and invoke the system command 'cake -DREADER figure-name.idoc' ". From there control leaves the manuscript environment and passes to the figure making cakefile which may interpret it arbitrarily, and if it is ignored, control passes along to the default cake rule which now invokes the vplot display program leaving the reader with (theoretically) only color and zoom to adjust.

Rule making

The figure building script is written in a document maintenance language called cake, which is an extension of the UNIX utility "make". You can define global "rules" to be included in any `cakefile` (collection of cake commands). These global rules give the software brevity and cohesion. Our biggest difficulty with these rules is that we are unsure of what they should be. I recently realized that the rules can be, and should be different for different people. For example, sometimes you want to put on your reader's hat and do what readers will be allowed to do. Other times, you want to put on your author's hat and install changes in the book. Still other times, you will want a hacker's hat for experimenting. Still another hat is "installer" where you are trying to bring the book up on another computer and want to see how many of the figures will build, which won't, and why. We need to develop the rules that enable a person to identify his/her role, reader, author, hacker, or installer, and then type one command like "cake" and see it all happen. We are still unclear about when and why a figure directory must be copied and work done in the copy, and when the original figure directory can be safely used.

REMOTE READING

There are two ways of reading an electronic book. One way is to import and install all the software. This way will be described in the next section. Another way is to run the book software here at Stanford and to read the book over the network using X windows. The network method seems radical, and we certainly didn't plan for it, but in the last few days before this report deadline, Joe Dellinger in Hawaii demonstrated that he can read my electronic books across the ocean. He sees a bit-mapped screen of page images of the book. He can turn the pages and "play with" the interactive figures. We'll have more to say about this in the future.

INSTALLATION = DRAWBACKS

The realistic possibility of *delivering* the electronic book depends critically on the ease of installation. The world is showering us with software, and we are limited mostly by what we can afford to learn, not what we can afford to buy. So, it is important to be able to set up an installation procedure that will not be too aggravating.

It is always impressive to drop a Macintosh disk into the slot and watch the software magically come up, but we are working with software that we design to run on machines of many vendors. Still, perhaps we should attempt to prepare an "all binary" book for a major platform or two. But the problem with binaries is that they may work fine for a year or so, but then the vendor will upgrade the OS and they break, whereas source code is more maintainable.

Hardware

First, on the hardware front, there is nothing but good news. Current workstations in the ten to twenty thousand dollar price range have more than adequate capacity. Since SEP-67 we at SEP have acquired the new exabyte-style tape drive where a hand-sized tape holds two gigabytes of data. That is about two-hundred times more than adequate to hold the text and software and about twenty times more than adequate for the scanned figures. Personal computer diskettes are much too small for this application. CD-ROMs have held promise for a long time, and perhaps they will have a role to play when (as we are being encouraged to believe) system software is routinely delivered on them.

Software

On the software front, the news is mixed. A bad trend is that we are finding more software packages to be helpful and even essential, and thus we have more components overall. A good trend is that vendors are working to increase compiler and operating system compatibility. And we are gaining more experience at building platform independent software. SEP has 4 binary platforms. The book is fully operational on the Sun-4 and the Convex. I have not addressed the byte-swapping issue required for complete compatibility with our DEC-3100. The books would probably run on our many Sun-3's but they are old and slow and some parts of their environment are degraded and not worth maintaining, except for book display.

cake. Dave Nichols reports that cake worked immediately on all our platforms.

ratfor is provided by AT&T, the inventor of UNIX. It was once part of the standard package, but has become an added cost extra, and it is not supported at all by some vendors, such as Convex. AT&T will sell the source, but not cheaply, and last I heard, they give no support. On the other hand, the DEC Ultrix fortran compiler will compile from either fortran or ratfor source.

seplib is the most popular I/O library invented and used at SEP. It is also formally supported by a group of French sponsoring companies, and is informally supported by SEP alumni at various sponsoring companies. Seplib is deeply dependent on internals of the UNIX operating system, and thanks to Stew Levin and Dave Nichols, it has survived the arrival of RISC, POSIX, and ANSI-C. Fortunately, we are getting organized with Benoit de Boullay at the French

Petroleum Institute to try to maintain it in an orderly way across more hardware platforms and prevent future environmental traumas. (Seplib is best documented by its on-line manual pages which *might* appear in this report. About the only description of it in SEP reports is found peripherally in Claerbout, 1986.)

vplot is SEP's intermediate plot language which all our programs write, which can be interpreted on all our devices. Vplot is principally the work of Joe Dellinger, though it is based on early work of Rob Clayton and Dave Hale. It seems to be robust and stable, though there are many opportunities for disruptions with future operating systems. The biggest current problem is that the filter from vplot to Xwindows has some aggravating bugs. Rick Ottolini wrote one version called X11pen, but the most popular version here now is Xvpen, written by Steve Cole in XView.

TEX and LATEX are fine word processing software, free I think, and they are getting easier to install. But the fonts can take a lot of space.

XView is a freely distributable tool kit from Sun Microsystems that is needed for the interactive programs.

X Windows is the key piece of software that has made our multi-platform environment possible. But it is not problem-free. Installation and maintenance are not trivial, especially when trying to support the various vendor-specific versions of X (DECwindows, Sun OpenWindows) as well as the portable MIT version.

I hope the above list of troublesome maintenance areas is complete, but fear there will always be another snag.

Copyright

I plan to explore the possibility of "copyleft" for the electronic book, and "copyright" for the paper book distributor. Copyleft is a legal invention of Richard Stallman at the Free Software Foundation. Oversimplifying, it means you can give it away, but not incorporate it in a product of your own, unless you give that away too. Publishers do not seem opposed to copyleft on the electronic document. They say that inexpensive copy machines have not put them out of business, and they don't expect clumsy and expensive workstations to do so either.

ACKNOWLEDGMENTS

I am thankful to Rick Ottolini and Steve Cole for helping me convert my old interactive programs from Sunview to XView. I'd like to thank Steve Cole for xtex, Sun, Xview assistance, and specific X comments in this manuscript. Martin Karrenbach and Pinghua Young for LaTeX assistance, and Dave Nichols for help in deciphering the many things that go wrong.

REFERENCES

Claerbout, J., 1986, A canonical program library: SEP–**50**, p. 281–289.

Claerbout, J., 1987, Interactive filter design in the Z-plane: SEP–**56**, 263–271.

Claerbout, J., 1988, Interaction with one-dimensional seismology: SEP–**57**, 513–521.

Claerbout, J., 1990, Active documents and reproducible results: SEP–**67**, 139–144.

Claerbout, J., and Nichols, D., 1990, Why active documents need cake: SEP–**67**, 145–147.

Cole, S., 1990, An interactive processing environment: SEP–**65**, 305–314.

Cole, S., 1990, Xvpanel: interactive programs in one line or less: SEP–**67**, 161–166.

Dellinger, J, 1989, Why does SEP still use vplot, SEP–**61**, p. 327–335.

Karrenbach, M. and Nichols, D. 1990, Progress towards the interactive book: SEP–**67**, 149–151.

Nichols, D., and Cole, S. 1989, Device independent software installation with CAKE, SEP–**61**, p. 341–343.