Electronic Documents Give Reproducible Research a New Meaning

RE1.3

Jon F. Claerbout and Martin Karrenbach, Stanford Univ.

SUMMARY

A revolution in education and technology transfer follows from the marriage of word processing and software command scripts. In this marriage an author attaches to every figure caption a pushbutton or a name tag usable to recalculate the figure from all its data, parameters, and programs. This provides a concrete definition of reproducibility in computationally oriented research. Experience at the Stanford Exploration Project shows that preparing such electronic documents is little effort beyond our customary report writing; mainly, we need to file everything in a systematic way.

In 1990 we began experimenting with electronic documents that merge our scientific software with our word-processing software. A year later we manufactured a CD-ROM containing a new textbook, Joe Dellinger's doctoral dissertation, and two progress reports of the Stanford Exploration Project. We distributed these CD-ROMs[1] to sponsors and many friends at the 1991 SEG meeting.

In 1990, we set this sequence of goals:

- Learn how to merge a publication with its underlying computational analysis.
- Teach researchers how to prepare a document in a form where they themselves can reproduce their own research results a year or more later by "pressing a single button".
- Learn how to leave finished work in a condition where coworkers can reproduce the calculation including the final illustration by pressing a button in its caption.
- Prepare a complete copy of our local software environment so that graduating students can take their work away with them to other sites, press a button, and reproduce their Stanford work.
- Merge electronic documents written by multiple authors (SEP reports).
- Export electronic documents to numerous other sites (sponsors) so they can readily reproduce a substantial portion of our Stanford research.

We met all these goals and set new ones:

- produce all new documents in this form, including lab reports in formal classes and "lab notebooks" of research progress.
- make incremental improvements in electronic-document software
- seek partners for broadening standards (and making incremental improvements).

Our basic goal is reproducible research. The electronic document is our means to this end. In principle, reproducibility in research can be achieved without electronic documents and that is how we started. Our first nonelectronic reproducible document was a textbook in which the paper document contained the name of a program script in every figure caption. The program scripts were organized by book chapter and section so they could be correlated to an accompanying magnetic tape dump of the file system. The magnetic tape also contained all the necessary data to feed the program script.

Now that we have begun using CD-ROM publication, we can go much further. Every figure caption contains a pushbutton that jumps to the appropriate science directory (folder) and initiates a figure rebuild command and then displays the figure, possibly as a movie or interactive program. We normally display seismic images of the earth's interior, but to reach wider audiences, Figure 1 shows a satellite weather picture which the pushbutton will animate as seen on commerical television. We include all our plot software as well as freely available software from many sources, including compilers and the LaTeX word processing system. Naturally we cannot include licensed software, but with the exception of Fortran and C compilers and the UNIX system itself, our publication includes source code for everything needed. The CD-ROM, at 680 megabytes, is so large we have had room for many executable programs on popular brands of workstations. The presence of these executables gives our readers a fast start.

Nearly everyone would rather read a paper book than the bitmapped page images on a screen that you see with an electronic document. But the illustrations in the electronic book are mostly in color, many are movies, and some are interactive. So the electronic book gives the reader a better understanding of the results. We typically use an interactive movie program to compare seismic sections where successive frames include processing with various parameters. The movie medium is much more informative than comparing seismic sections side by side. 3-D volumes are much better exhibited by movies than static paper illustrations. We are delivering a volume of software that is accessed like a book.

[1]SEP-CD-1 is available from Stanford University Press, $15 plus shipping, tel 415-723-1593

Reproducibility

The principal goal of scientific publications is to teach new concepts, show the resulting implications of those concepts in an illustration, and provide enough detail to make the work reproducible. In real life, reproducibility is haphazard and variable. Because of this, we rarely see a seismology PhD thesis being redone at a later date by another person. In an electronic document, readers, students, and customers can readily verify results *and adapt them to new circumstances* without laboriously recreating the author's environment.

Verification of reproducibility

Judgement of the reproducibility of computationally oriented research no longer requires an expert—a clerk can do it. To verify reproducibility of research, four steps are required:

1. Issue the "burn illustrations" command.
2. Activate a search program to hunt for any figures labeled "reproducible" remaining in the file system. (This might occur if authors do not correctly design their "burn target".)
3. Issue the "build illustrations" command and wait for its completion.
4. Print or view the document to verify the presence of the original illustrations.

> Our basic goal is reproducible research and the ability to publish it in reproducible form. The electronic document is our means to this end.

Effort

Authors producing an electronic document are required to file everything in a standard way and use certain names in their command scripts. Otherwise it is no harder to make an electronic document than a paper document. To propagate the methodology to the other 14 members of our group, we used only a one hour training session. Of course, our group was already accustomed to our common software collection for making paper documents.

Free

We have chosen to base our work on freely redistributable software because of the traditional academic need to publish. Work that we do that is based on proprietary software and unconventional hardware is labeled "nonreproducible" in our documents although the hooks often exist beneath the surface for readers who have the nonstandard resource. The only word processor meeting our requirements is TEX which although it produces unsurpassed displays of mathematics, is not as easy to learn or use as popular commercial word processing software.

Remote access

There are two ways of accessing an electronic book. One way is to get our CD-ROM and run it. Another way is to have an account on one of our machines and open a window over the network using X windows. Electronic documents installed in California have been read over the network from Colorado and Hawaii. The remote reader sees the same bit-mapped screen of page images of the book that a local reader sees, and can press the interactive figure buttons. The software then runs at Stanford and the remote reader controls it. Things run a little slower because of communication delays. Our initial trials were marred by lack of compatible fonts on the cooperating machines, but we believe this defect in X-windows has been removed by recent releases. We are not yet promoting this method of document access, but we believe it has a great future.

At the present time, a minimum hardware configuration costs 5-10 thousand dollars.

Interactive vrs. reproducible

An illustration can be either interactive or not and it can be either reproducible or not. At first we confused these concepts.

To insert an illustration in a document, the author must provide a pathname to a postscript plot file with a name either of the form `NAME.ps` or of the form `NAME.ps.save`. The caption of the illustration is marked $[R]$ denoting reproducible when the plot file name is of the form `NAME.ps` and it is marked nonreproducible $[NR]$ when it is of the form `NAME.ps.save`. The burn-illustration commands destroy file names of the type `NAME.ps`. For a nonreproducible figure to be marked reproducible requires two unlikely events: first a mislabeling, and second, a failure to run the burn and build cycle on the completed document.

An *interactive* illustration has a pushbutton in the caption. The pushbutton is the interface between the word-processing program and the command scripts. The label on the pushbutton holds the pathname to the command script and the name of the figure. The word processor executes a "change directory" to where the command script is, and then launches it. What happens next is determined by the conventions and style of the author.

Our most recent research progress report (SEP-73) has 18 authors and 246 illustrations. Somewhat more than half the illustrations are reproducible. There are 165 captions with pushbuttons and 81 without. Nonreproducible illustrations are mostly line drawings from a draw program. Other nonreproducible illustrations used our parallel computer or had been done at other sites. Most of the pushbuttons simply rebuild and replot as described below, but many increasingly show movies.

Cake

UNIX systems use something called a `makefile` to maintain software. The remainder of this paragraph explains the

makefile concept for those unfamiliar with it. The final result of a research calculation is a plot file to be included in a document (and perhaps a movie to be launched when the figure caption button is pressed). To get these final results, many source files are needed such as data, main programs, subroutines, and parameter files. Between the source files and the plot files are intermediate files such as object files, executable files, and processed data files. After a plot file has been built, it should be younger than the intermediate files which should be younger than the source files. The makefile contains the commands needed to build everything and knows the expected age relations among the files. When the age relations are as expected, then a call to rebuild a plot file will report that the plot file has already been built. Changing the date of any file, say by editing a program or a parameter file, will cause out-of-date files to be rebuilt the next time the plot file is called for. Thus, when a reader presses a button in a figure caption, the figure is rebuilt if it is absent or out-of-date.

We find that makefile methods are highly nonstandard from one UNIX system to another, and that a publically available variant of **make** called **cake** works on all systems and provides more powerful ways of maintaining documents. One thing we particularly like about **cake** is that unlike **make**, it does not require intermediate files to be present. Thus, after we finish building a plotfile, we can clean up "**cake clean**" by removing all the intermediate files such as objects, executables, and processed data. (Authors could use **make** instead of **cake**, but the price they would pay on the next call for a plot is that **make** would insist on rebuilding all the missing intermediate files.)

Makefiles and cakefiles contain rules for building files from other files. For each rule, the created files are called "targets" and they are derived from files called "dependencies". We have built about six major documents with over a thousand illustrations. This experience has been distilled to 2-3 pages of common cake rules called `SEP.idoc.rules` that we typically `include` in every figure-building cakefile.

CD-ROM versus magnetic tape

We originally thought of producing electronic documents on magnetic tape instead of CD-ROM. Reasons to publish on magnetic tape are: More people have magnetic tape drives. Tape technology is better understood by more people. We can easily make small numbers of magnetic tapes.

Reasons to produce our interactive publications on CD-ROM are: When we hand a CD-ROM to someone and they put it in their drive, they have instantaneously mounted 600 Mb (megabytes) of our file system in their computer. They did not first need to clear 600 Mb of space in their computer and they did not need to wait two hours for the tape to read. CD-ROM is more suited than magnetic tape to please the recipient. Thus CD-ROM is a better publication medium. CD-ROM readers are likely to become *more* numerous than tape readers because manufacturers have begun distributing software on CD-ROM and the drives themselves are much cheaper than tape readers. CD-ROM is more durable than tape and more compact.

Manufacturing costs are variable, but in June 1992 we received an offer which amounted to manufacturing the first four disks for $150 each, thereafter the price would drop by a factor of a hundred to $1.50 per disk. Production times are about a week.

File systems

A CD-ROM disk is a read-only file. The entire disk holds about 680 Megabytes which could be broken into several tracks, but for most software purposes it would be handled as a single track. A UNIX file system is a file with a special form. Such a file can be "mounted" by the UNIX mount command and it is then accessible as a file system (you can snoop around on it using `cd` and `ls` commands). The UNIX `tar` command will take a UNIX directory tree and create a single file from it, but this file will not be in the proper form for a CD-ROM drive. For the material on a CD-ROM to be a file system that can be mounted, the file must be prepared by a tar-like program. One such program that we purchased (with a huge academic discount) for CD-ROM mastering is called **makedisk** from a company called Young Minds (YM). Before that we mimicked a CD-ROM with the UNIX **mount** command by mounting a file system in read-only mode.

How to overcome the read-only limitation

In our research we were not accustomed to read-only file systems. Our worst fear originally was that the software we put on a CD-ROM would be of little use until someone copied the whole thing onto their hard disk. This would make CD-ROM as inconvenient as tape. However, the following well-known technique makes everything easy:

Nearby is a tree with about 3500 leaves on it and in a nearby computer file system there are about 3500 files. In winter time all the leaves fall off the tree and then light passes directly through it because the leaves have much more surface area than the branches. Likewise the computer files take much more space than the directory structure. So the trick to using a CD-ROM as if it were a read-write file system is to first make a copy of the CD-ROM's directory structure on your hard disk. Like the tree branches, that isn't too voluminous. In this directory structure you do not put files, but pointers (symbolic links) to the files on the CD-ROM. Now whenever you need to change a file, you replace the link to the CD-ROM by a copy of the file on your hard disk.

The tree without its leaves is often called a "shadow tree". To make one we could use a well-known simple shell program called `lndir` (distributed with X) but instead we use a program called `cd_link`, free from Young Minds. It works much like `lndir` except that it performs the required name translation from ISO 9660 names (a CD-ROM standard) to our directory names. The only drawback is that it takes about 5 to 30 minutes to make the shadow tree. Luckily, you only pay this price the first time you load the disk.

The ISO 9660 Standard

From the Young Minds documentation we learn that the CD-ROM standard, ISO 9660, mandates that file names be restricted in several ways. Basically, 1. have no lower case and 2. fewer than 8 or 11 chars. `makedisk` maps our file names into ISO 9660 compliant names. Then after the user mounts his disk, the user needs to run a program `cd_link` (supplied by YM as C source) that appears to build (on the user's hard disk) a shadow tree with our names and symbolic links to the ISO 9660 named files on the CD-ROM.

We understand that manufacturers are planning CD-ROM drivers that should make CD-ROM usable on our target range of work stations without requiring the use of `cd_link` or a shadow tree.

A disadvantage of publication on UNIX machines is that superuser status is now needed to use the mount command. Thus, an amateur cannot pop a disk in and out as simply as on a Macintosh. It is rumored that experts are working on this problem, but we can expect to wait a while before a multivendor solution comes into view.

We recently received a free CD-ROM called "Software Store" which contains demonstrations of a wide range of software for sale. A little booklet came with the disk. Unfortunately the first four pages of this booklet are devoted to "super user" commands required to be run before we can see anything. Our CDs are a little easier, but not much.

Copyright

Free software does not mean public-domain software. Free software is copyrighted, but it comes with a public license that allows you to use it, copy it, and redistribute it provided you follow certain guidlines. The clearest and most widespread concept of free software is the brainchild of Richard Stallman, founder of the Free Software Foundation, Inc. (also known as GNU), 675 Mass Ave, Cambridge, MA 02139, USA, who will provide a copy of their most up-to-date public license. We believe the GNU concept is ideal for universities and companies that are consumers of software, but risky for companies that sell software because if you merge your software with publically licensed software, you may then be unable legally to sell the combination. We put our basic libraries and electronic book software under GNU license so the public can use them as an exchange standard, but we handle most other specialized materials in the ways that universities customarily do, i.e. by copyright and license.

I (JFC) have found some publishers that are willing to publish a traditional book even though there is a publically licensed electronic book. Inexpensive copy machines did not put publishers out of business and they don't expect clumsy workstations will either.

Workbench

The "interactive document" is a concept. Beyond this concept are realizations of ever-increasing completeness and utility. As underlying interactive software becomes more powerful and more friendly, we may see the electronic book become the "desktop" of interactive computing.

CONCLUSION

The human consequences of reproducible publication have hardly been considered. Yet this medium of scientific communication has now appeared. With workstations becoming widespread and software available, the burdens imposed on the author to create reproducible results are little more than the task of filing everything systematically. We are nearing a time when it will simply be the author's choice whether to keep detailed means to results confidential with the use of traditional publication or to communicate fully by using reproducible documents.

Figure 1: This figures is interactive. Pressing the button at the end of the caption will bring up a movie of satellite pictures showing the weather situation over California during one day in June 1992. CI060210 [NR]