Sharing Sea Grant Models with the Public

Joy E. Moses-Hall

College of Marine Studies
University of Delaware
Newark, DE 19716

Robert A. Dalrymple

Center for Applied Coastal Research
University of Delaware
Newark, DE 19716

September, 1992

Contents

Part I

PROBLEM: Should Sea Grant make its numerical models available to the public? If so, how?

As a governmental body, the Sea Grant Program feels an obligation to produce a product that it can tangibly hold out to the public. The numerical models developed under various Sea Grant–supported projects at many different institutions could be useful to coastal scientists and engineers in industry nationwide. But the models produced by Sea Grant researchers are not ready-to-use in the sense that they are often user-unfriendly. The difficulties are two-fold:

1. Some of the models are just not suited for public use. Although they shed light on narrow scientific questions, they may be too far from the applications stage to use practically. On the other hand, they may be too specific; designed and tested for only one isolated spot in the world. Considerable effort would be needed to generalize such models.

2. Numerical codes by themselves are not all that helpful to anyone other than the developer unless they are well documented. Most of the models funded by Sea Grant are undocumented beyond a few comment lines. Adding documentation, then, means adding to the cost of producing the model. Estimates of just how much cost vary widely, and depend on the amount of documentation needed and the expertise of the intended user. One scientist, whose models are mostly of the type described in 1, estimated an additional ten to twenty percent of time/money to produce a user's manual. An engineer, who has produced models and documentation available commercially, estimates much more. Some engineers have found that users are not nearly as expert as anticipated.

   If costs were extremely limiting, a fee could be charged to the users of Sea Grant funded models, large enough to support the production of the documentation necessary for many of the models to be made public.

1 Public Access

Making the models available to the public is less painful than making them suitable for public consumption. The broadest availability scheme would include both electronic and hardcopy modes. A catalog and an archive would then have to be maintained, at the very least.

Sea Grant could choose to set uniform standards for its programs and documentation, such as the Army Corps of Engineers standards, but these would require oversight, scientific

programmers, and a degree of commitment in funds and manpower that may be beyond the scope anticipated. Further, increasing the level of principal investigator (PI) commitment to program documentation, diverting effort from science and engineering, is a significant trade–off, which should not be taken lightly. Therefore, this avenue will not be pursued further here.

1.1 Worksheet and Catalog System

As an alternative, a more self-sustaining system could be established. The basic element of such a system would be a worksheet, such as the sample shown in Appendix 1. It would be submitted by the PI at the conclusion of a funded project, when the model was completed. It provides all of the information needed, along with an abstract such as the one routinely required by Sea Grant, to catalog the model. Potential users scanning the worksheet would know right away if it met their needs and hardware requirements.

The first half of the worksheet identifies the model and its requirements. If the programs come compiled, for instance, it is less vital that the user have the programming language software.

The second portion of the form tells the user how to get the model itself, via a network or bulletin board or through the mail, and the last part informs the user about the difficulty associated with using the software. By listing the specific categories of documentation provided, the potential user can infer the friendliness of the system. A model with many documentation options available to the user will, in general, be easier to use than a model with only one or two document types.

From this worksheet, a catalog of all of the models provided to Sea Grant can be made. This catalog could be on-line or just in a notebook, which is periodically published. The catalog would also include a disclaimer, absolving Sea Grant and the modelers of all blame if users (or the model) go awry.

The catalog would require very little maintenance. The worksheets would have to take entered into the catalog for each new model. Appendix2 shows an example of how the Table of Models for the catalog might look. The program name and purpose, computer type, and programming language would be the same as those on the worksheet. The names of the Principal Investigators would be both a source of assistance and a 'brand name,' so to speak, to the user. The level, or degree of difficulty, of the model (the Level column) would indicate to potential users the ease of use of the software. The level would be determined directly from the documentation boxes checked off on the worksheet. Any model which includes tutorials or help screens would be the easiest to use, and so would have a Level of 1. Any model which provides only the computer code and a scientific theory section, and perhaps a list of references, would earn a 3. Anything in- between would be a 2.

The Availability column would list the network name, bulletin board name, or just Hardcopy. If the models are to be in a public archive, the PI would be responsible for sending a copy of model and documentation to the appropriate computer, or sending out hardcopies

upon request. As the funding agency, Sea Grant could require an archived copy of the model before advancing journal page fees, for instance. Maintenance of the catalog and archive would then amount to only the entry of the worksheet information into the Table of Models.

The catalog itself would then begin with the disclaimer and Table of Models. Following the table would be two pages for each model: the worksheet page, and the abstract. All of the information a user needs to get started would be available in a catalog form, with little extra effort on the part of Sea Grant.

1.2 Case Study: Army Corps of Engineers

The U.S. Army Corps of Engineers has committed considerable energy into developing software and documentation for the public. The Automated Coastal Engineering System and the Coastal Modeling System (still under final development) are good examples of a coordinated effort to produce and disseminate engineering models.

The Corps' standards were set more for the benefit of the Corps than for the public. Corps programmers were developing models for use by other Corps personnel—a closed system where the developers and users are a known group. Standardization improves efficiency for the users, who are also Corps personnel. Public access to the software and documentation is more of a peripheral benefit. This differs from Sea Grant's position in that the program developers have no direct link to the users. Also, because the user group's abilities are unknown, any documentation provided will make assumptions about the user that may be inaccurate.

Corps documentation is very detailed. The Automated Coastal Engineering System (ACES) comes with a tutorial, pull-down screens, test problems, references, and scientific theory. The models have been integrated with each other so as to form a package unit of similar style, so that once the user has mastered one model, the others are familiar.

The Coastal Modeling System will eventually have similar pull-down screens. Currently, it runs on a Cray supercomputer but is being modified for workstations. Developers prepare documentation according to guidelines developed by Corps staffers, some of whose *sole* responsibility is documentation and organization of the package. Mary Cialone, of the Research Division at the Coastal Engineering Research Center, indicated that the researchers generally write the scientific theory part, with realistic examples and an appendix of records (input and output), variables, and ranges of values, while the staff writes the documentation for supporting software. Even the written documentation conforms to a standard style. Occasionally, staffers write all of the documentation for a model segment, working from the code and comment lines and with review by the developer.

Sea Grant could also adopt rigid guidelines for software development, but this may raise costs and tempers more than necessary. Corps software is guaranteed an audience; some Sea Grant models may lie dormant, unused by or unfit for the public. Sea Grant's models span a wider range of programming talents, at a variety of institutions with many different programming styles and many different resources. Some developers may prefer to do all

of their programming on a workstation; others may need a supercomputer's abilities. A vast documentation system would laden a small organization like Sea Grant with prohibitive costs. Instead of requiring all software to conform to a specific style, with common input formats and the same utility commands, simply detailing what those formats are for each model would at least forewarn the user.

1.3 Case Study: Center for Applied Coastal Research

The University of Delaware's Center for Applied Coastal Research has accumulated a number of numerical models that it is prepared to make accessible to the public. Although the Center plans to charge a fee for many of the more sophisticated models, logistical problems similar to those of Sea Grant make it a prototype.

Some of the Center's models are already available commercially. The researchers have developed documentation and even offered short courses for interested users. The University is on the Internet network, and plans to archive the models on a workstation directly accessible through Internet. The models with a fee will, of course, not be directly accessible, and will require intervention by someone who can verify receipt of payment to actually send the model and documentation.

The Center sees most of its users as mostly fellow academicians or experienced engineers. As some of the models are complicated, and much of the documentation is put together from the point of view of the developer, the models sometimes are difficult for anyone other than a researcher in an allied field to use.

The Center plans to keep a catalog, as described in section II, which will be physically available at the Center. It will be updated as new models and updates mature. Users will be able to request a copy of the current catalog.

2 Conclusions

Sea Grant should be able to make its models available to the public without undue effort. Any successful PI who suggests that one of the benefits of the work will be a public domain model could be sent a copy of the worksheet (Appendix 1) and perhaps accompanying documentation instruction. At the conclusion of the project, Sea Grant would expect to receive a copy of the model, its documentation (including abstract), and the completed worksheet. Submission could be electronic if that is convenient. Sea Grant would have to dedicate a computer to this task, and monitor it, so some cost will accumulated.

All the Sea Grant Program would have to do to maintain the system would be to enter the worksheets and abstracts into the catalog and update the Table of Models. An on-line catalog would require less handling; a hardcopy catalog may have to be published periodically and disseminated.

Part II

Documenting Software–General Principles

Software requires documentation if it is to be used properly by anyone other than the developer. Not only does the documentation introduce the user to the capabilities and operation of the model, it frequently determines whether a particular program is used in the future.

The following deals with the documentation of software in general for wide public use. The requirements we provide below are for the ideal case. For most Sea Grant sponsored models, the level of effort necessary to produce such documentation would be prohibitive; therefore it should serve as a guide, rather than a requirement. Additionally, new technologies for documentation may make documentation easier in the future. For example, the program WEB, with its processors WEAVE and TANGLE, provides the ability to integrate the documentation of a program directly in with the code. Since this actually can make programming easier, it is expected that there will be a greater use of these programs in the future. (Typesetting of the documentation and the code in TEX or LATEX is carried out by WEAVE. When the program is to be run, TANGLE automatically strips out the documentation, leaving the code in a form to be compiled.)

Before writing documentation, certain assumptions must be made about the user. If the user is anticipated to be an experienced computer operator, for example, the documenter might leave out simple computer instructions such as how to access the network or install the program. However, it is usually safest to make minimal assumptions about the user's technical background, and given unlimited resources, all software would include documentation of the following kinds; although some aspects of these may not be necessary, depending on the model.

User's Guide

- Abstract
- System specifications
  - Hardware
  - Software
- Installation
- Tutorials
  - Hardcopy
  - On-line
- On-line Help
- Examples
  - Graphics
- Error Discussion
- Glossary
- Index

Reference Manual

- Theory
  - Purpose
  - Review
  - Methodology
  - Examples
- Code
  - Comment Lines
  - Automatic Documentation
- Index

Quick Reference cards

Keyboard Overlay

After all of the documentation has been prepared, it must be tested for accuracy. If the instructions don't work, neither does the program in the eyes of the user. The developer should double-check everything. The best test is to have someone unfamiliar with the software use the documentation to try to learn it. Any confusion or missteps this person accumulates must guide a revision of the documentation.

1 User's Guide

The User's Guide introduces the user to the software. It provides some computer-handling instructions to ensure the user is up-to-speed, as well as describing the technical theory that the software models. The language should be simple and clear, with frequent subheadings. It should be a teaching tool, showing the user how to manipulate the software. Eventually, the user should be able to wean himself or herself from the user's guide and rely on the other documentation for quick reference or if problems arise.

1.1 Abstract

The abstract should be a short (1 page) description of the software in terms of what it produces. It should help the user zero in on the exact model he or she needs to extract a desired result. It should state the mathematical assumptions and constraints. This piece of documentation is already required from Sea Grant PI's, and would be in the catalog of models, as well.

1.2 System Specifications

The first chapter should review all the necessary hardware and accessory software requirements for the model to run. The information, which may be tabular, must tell the user what equipment and settings he or she needs to run the software. The chapter should be organized around two headings: the computer hardware and the accessory software.

1.2.1 Hardware

Hardware would include the amount of permanent memory the software occupies and the typical type of computer for which the software was intended (such as IBM mainframe, Sun workstation, PC or Macintosh). If a modem, a tape drive, or other peripheral is needed, this section would state the size and specifications. A list of the printers supported by the software, if applicable, would also appear.

1.2.2 Software

The accessory software section should tell the user what programming language is used, and whether a compiler is needed. The user needs to know the type of operating system the software assumes, such as MS-DOS, Windows, or UNIX, and any standard software, such as statistical subroutines or graphics packages, the software uses.

1.3 Installation

Installation may be purely electronic or may require handling of diskettes. If the software is available on a network or bulletin board, particularly if the documentation is shipped separately, the installation chapter should include how to access the network or bulletin board, the directory where the model is stored, and all the specifications and instructions to download the software onto the user's machine. Diskette installation instructions should tell the user how to uncompress the software, if necessary, and in what order to load the files.

At the end of the installation procedure, the software should be ready to run.

1.4 Tutorials

A tutorial is a step-by-step description of the operation of the software. It includes computer prompts and appropriate responses. It should lead the user through the main part of the model, including any preparatory steps such as constructing an input file in the correct format. This is the section that teaches the user how to run the program by shear repetition. Eventually, the user will forgo the simple instructions here and use only the reference materials.

1.4.1 Hardcopy

For a book-form tutorial, a good outline format is best, so as the user becomes more experienced, he or she can skip over the steps he or she has mastered. The sequence of steps in the program should be followed in the tutorial. The user then will be able to orient himself or herself within the sequence, such as towards the beginning or end of running the model. Cross- referencing of commands and options may be helpful.

1.4.2 On-line

On-line tutorials are the same as hardcopy tutorials, except they appear on the computer screen on demand. They may be accessed by icons or menu options.

1.5 On-line Help

On-line help may include help screens, pull-down screens, or other formats. This kind of documentation is frequently the easiest for users to use and the most difficult and time-consuming for developers to produce. There may be considerable overlap if both on-line and book-form documentation is provided. On-line help that is difficult to access may require its own hardcopy documentation, a situation to be avoided.

1.5.1 Help Screens

Help screens provide a brief summary of the actions of commands or keys, usually accessed by a specific keyboard command. They quickly point the user to the information needed, without having to flip through pages, but may be cumbersome if a long sequence of options or specifications are offered. They are rarely cross- referenced.

1.5.2 Pull-Down Screens

Pull-down screens are a combination of help screens, tutorials, and data entry. They are interactive, leading the user from step to step and routine to routine as dictated by the input responses. Some rely on menus, or categories of settings, that can be adjusted at any time to reflect changes the user wants. Others ask for input in a step-by step fashion, as the logic of the program needs it, but in an intuitive (and English!) format. The screen will usually specify any particular measurement system or format required by the processor. For example, after starting a program, a choice screen may ask the user which product the user wants, and when the user selects, say the wind stress field, the next screen to pop up will ask the user for parameters only for that particular operation.

1.6 Examples

Examples serve two purposes: to serve as a tutorial for any branches of the software not covered in the main program tutorial, and to expose the user to sample results across the realm of the model's capabilities. If the model has a limited domain, the examples should show results at both the maximum and minimum extremes, such as at minimum and maximum grid steps. Examples should walk the user through from initial data input to model results. Examples should also be given of the input and output files, with the columns and rows clearly defined and the units specified. Differences from row to row, such as a time step, should be identified.

1.6.1 Graphics

Throughout the documentation, but particularly in the examples section, illustrations should set off the words of the text. Flowcharts and diagrams breakup the text and often summarize a lot of information. And in the examples section, graphs of output help the user to interpret results.

1.7 Errors

Any software available to the public should have been debugged, but users might still occasionally get error messages. The developer should include a review of errors that could arise from inappropriate user inputs. For example, a time step that is mismatched to the spatial grid could cause a model to blow up. The developer should describe the computer's message when this happens, as well as how to adjust the time step.

The developer might also want to include some statistics on the reliability of the model, comparing its results with observations, for example. Tolerances, especially in iterative schemes, should be noted.

1.8 Glossary

A glossary of terms that might be new or ambiguous can be helpful to both novice and experienced users. By carefully defining any term that might be misunderstood, such as coastline, to pin down the meaning to accepted standards, such as the mean high water line for coastline, the user is more likely to use the software effectively.

1.9 Index

Many word-processing systems will all but index the documentation for the developer. The index summarizes the cross references and allows the user to narrow a documentation search for a particular item. It is most helpful in hardcopy documentation.

2 Reference Manual

The reference manual will be the volume the user turns to after he or she has some experience with the software to double-check a procedure or try something new. Errors may drive the user to the reference manual, as well.

2.1 Theory

The theory chapter is a review and discussion of the science and engineering behind the software. It includes the governing equations, assumptions, and boundary and initial conditions of the model. It may compare the criteria for this model with other models. It will discuss grid structure, if any. It will give acceptable ranges for parameters. Caveats, singularities, and instabilities should be discussed in detail, so that the user will know exactly how close he or she can come to those values, or how to circumvent them.

References to other works are acceptable, but vital steps should be presented. This chapter should guide the user through a scientific understanding of the processes that the computer automates.

The theory section is best presented in operational order, beginning at some basic level of understanding that the software developer feels is the minimum in order to use the software correctly. It is a good idea, in any teaching medium, to start a few steps before the user's knowledge drops off, to both orient him or her and to cement that knowledge.

2.1.1 Purpose

A brief statement of purpose should open the theory chapter. This statement should clarify for the user where in the model universe this particular program or set of programs fits in.

2.1.2 Review

This section explains some of the background leading into the model and differentiates between this model and any similar models. References given should help the user beef up his or her background in the problem to be solved by the model.

2.1.3 Methodology

The governing equations and boundary and initial conditions, as well as the assumptions the model makes should be prominent in this section. Specifications for parameters, whether set by the software developer or the user, should be discussed. The numerical method used for solving the equations should be addressed, and the grid space over which they are solved described.

Each process the model encompasses should be discussed. A multi- operational model which calculates products on top of products should have a section describing each product and how it was determined. For example, a software package that calculates a wind stress field and feeds that into a numerical solution for wave climate should discuss both how the wind stress field is developed and how the wave field is calculated.

2.1.4 Examples

A brief discussion of the uses, or examples of the results of the model, may follow the methodology. If, for instance, the model specifically creates a solution for a concrete structure but would work identically if the material used were brick, the modeler might reveal this here.

2.2 Code

The code is the software developers end product, and is occasionally the only documentation available for a model. A well-commented code may not only help the developer to return to the code for updates after he or she has begun to forget the intimate details of the code, but it can also make the documenting process easier by identifying all the sections, variables, and input formats buried in the code.

2.2.1 Comment Lines

Comment lines should explain all variables and parameters and their names, and introduce each section and subroutine of the program. They should stand as an outline of the program when taken alone, and summarize the entire code section by section. The written documentation can then be hung from this outline.

Inputs and outputs, their formats and filenames should be commented. If variable names evolve through a number of steps (as index bases change, for example), the start and end versions should be identified. References to published papers may be included in the comments. Comment lines can also act as delimiters of white space, setting off sections of code such as "plotting."

A desperate user may be able to locate the source of his or her frustration in the code itself if the comment lines truly describe the program. More expert users may wish to alter the program itself or add a subroutine for some specific function. Comment lines steer him or her to the portion of the program to be adjusted.

2.2.2 Subroutines

The use of subroutines within the program can clarify the flow of logic within the program. Computationally, subroutines are generally efficient, and if they are carefully commented

they can make the documentation more efficient, as well. Inputs and outputs and their formats and filenames should be noted in every subroutine, and their relationship to the variables and parameters in the main program noted. If the user supplies an input file or parameter, the contents of the file or the meaning of the parameter must be defined.

2.2.3 Automatic Documentation

Some utility programs can help with code documentation by following the logic of the program and identifying used and unused variables, flowcharts of subroutines, and indexes.

Toolpack

Toolpack, available on UNIX, has a statdoc command that can block out Fortran usage such as COMMON blocks and symbol characteristics. It can also produce a subroutine flowchart.

Toolpack can document information about a Fortran program's execution. Rundoc can block out more dynamic reports, keeping track of the number of times a segment of the program is executed. There are also commands that restructure the code to a more readable format.

Word Processors

Word processing software used to type up the documentation may have some automatic features, as well. Indexes, tables of contents, and lists can often be generated, and automatically updated as page numbers change, after the documentation has been written by simple text marking procedures. Macro commands can also be helpful in the overall structuring of the document.

2.3 Index

The Reference section should be indexed separately from the User's Guide, since ideally they are separate volumes used at different stages of a user's experience. Again, the indexing can be automatic by many word processors.

3 Quick Reference Cards

Quick-reference cards are very brief references for the user after he or she has gained some experience. The quick-reference may just be a listing of commands or keystrokes and a one-line description of the function. It may be a simple outline of the software. The key feature is brevity. Even a simple flowchart that diagrams the order of operations a user must follow

may be considered a quick reference. It should be the kind of document a user can prop by his or her computer and get full use of it with just a glance.

4 Keyboard Overlay

A keyboard overlay is similar to a quick reference card, but is particularly suited to software with many keystroke commands. As a template, it should fit over or above the keyboard and match the commands with the keys.

5 REFERENCES (Documentation)

Holtz, Herman. *The Complete Guide to Writing Readable User Manuals*, Dow-Jones-Irwin, Homewood, IL, 1988.

Simpson, Henry, and Steven M. Casey. *Developing Effective User Documentation: A Human Factors Approach*, McGraw- Hill Book Co., NY, 1988.

Singleton, Margaret. *Automating Code and Documentation Management: The Intelligent Guidance of Change*, Prentice-Hall, Inc., Englewood Cliffs, NJ, 1987.

References (Army Corps of Engineers)

Cialone, Mary. *Coastal Modeling System Overview (Draft)*, Research Division, Coastal Engineering Research Center, Department of the Army, Waterways Experiment Station, Corps of Engineers, Vicksburg, MS, 1991.

Leenknecht, David A., et al. *Automated Coastal Engineering System User's Guide (Draft)*, Coastal Engineering Research Center, Department of the Army, Waterways Experiment Station, Corps of Engineers, Vicksburg, MS, 1990.

6 Appendix 1. Worksheet

Information Checklist for Software Catalog

Program Name: _____

Program Purpose: _____

_____

Computer Type: (Mainframe, Workstation, PC) _____

Memory Requirements: _____

Program Language: _____ Compiled? ___yes ___no

Utility/Library Packages Needed: _____

Principal Investigator: _____

Institution: _____

Address: _____

Phone/Email: _____

Availability: Software: ___Hardcopy ___Diskette ___Online

If Online: Network/Bulletin Board Name: _____

FTP: _____

Directory: _____

Filename: _____

Documentation: ___Hardcopy ___Diskette ___Online

___Tutorial
___Index
___Scientific Theory
___Error Discussion
___Course
___Help Screen
___Glossary
___Code
___Examples
___References

If Online or Diskette, work processor/language: _____

Is this an Upgrade: ___no ___yes, to (Program) _____

Cost: _____

Program	Purpose	Computer	Language	Documentation	P.I.	Level*	Availability
IBREAK	Numerical Model for Design of Impermeable Coastal Structures	Wkst	FORTRAN	Manual Code Examples References	N. Kobayashi A. Wurjanto U. of DE	3	Internet
RBREAK	Numerical Model for Random Waves on Impermeable Structures and Beaches	Wkst	FORTRAN	Manual Code Examples References	A. Wurjanto A. Kobayashi U. of DE	3	Internet
REF/DIF	Combined Wave Refraction/ Diffraction Model	Wkst	FORTRAN	Manuals Code Examples References Short Course	R. Dalrymple J. Kirby U. of DE	3	Internet
REFRACT	A Refraction Program for Water Waves	PC	FORTRAN	Manual Examples	R. Dalrymple U. of DE	2	Internet

*

1=Bachelor-level science/engineering, no programming. Tutorials or help screens.
2=Master-level, familiarity with program language. Sequential steps, examples.
3=Ph.D.-level science/engineering, programmer in program language. Code, references