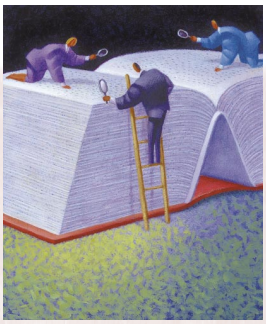
A Standard for Software Documentation

Vir Phoha, Northeastern State University

Standardizing software documentation could help reduce the time and effort spent in developing new software, increase the ease of porting software to different platforms, and help users understand software more easily. But there is no universally recognized standard for software documentation, in part because documentation style and content differ among programmers and sometimes differ for the same programmer under different circumstances. In addition, the choice of programming language and the nature of a program may dictate a particular style of documentation that might not easily apply to another environment.

While there is no universally recognized standard for software documentation, there is a standard for documenting engineering and scientific software. Developed by the American National Standards Institute and the American Nuclear Society in 1995, it is called ANSI/ANS 10.3-1995 Standard for Documentation of Computer Software. One of the standard's goals, as stated in the standard itself, is "to encourage better communication between developer and user, and to facilitate effective selection, usage, transfer, conversion, and modification of computer software." The standard is not a rigid set of specifications, but a guide that can apply to most software projects intended for internal or external use.



The ANSI/ANS 10.3-1995 standard provides a flexible, robust framework for documentation needs.

While the standard cannot cover all documentation problems, it is a good starting point, even for the most complex software. Similarly, while the standard provides recommendations for documenting scientific and engineering software, it doesn't offer guidance for online monitoring, control, or safety systems, and doesn't specifically address the unique requirements of consumer-oriented software. As a general guideline for clear, well-organized documentation, however, the ANSI/ANS 10.3-1995 standard can serve as a fine place for developers to begin a documentation methodology.

The standard itself is not only well written and fairly comprehensive, but it allows for individual developer differences and unique software documentation problems. In short, the ANSI/ANS 10.3-1995 standard provides a flexible, robust framework for documentation needs.

CONTENT RATHER THAN FORM

The ANSI/ANS 10.3-1995 standard—which makes recommendations for the content rather than the form of software documentation—suggests separating documentation content into four categories: the abstract, the application information, the problem (or function) definition, and the programming information. The standard also makes recommendations as to which items to include in the software package itself, such as floppy disks and manuals, and even how best to label those disks and manuals. But the most useful elements of the standard are the four content categories themselves.

Abstract

The abstract summarizes the software's major capabilities and purposes. The abstract's goal is to provide the user with enough information to enable an easy decision about whether the software is suitable for the user's needs. According to the standard, this information should

Where to Obtain the Standard

The standard costs $55 and can be ordered through the ANSI Web site (http://www.ansi.org) or by contacting ANSI at the following address:

American National Standards Institute
11 W. 42nd Street
New York, NY 10036
Phone: (212) 642-4900
Fax: (212) 398-0023

Editor: Charles Severance, Michigan State University, Department of Computer Science, 1338 Engineering Bldg., East Lansing, MI 48824; voice (517) 353-2268; fax (517) 355-7516; crs@egr.msu.edu; http://www.egr.msu.edu/~crs

include specific functions of the software, the required computer environment, and all the materials available to the user in the software package, including manuals.

Application information

The application information section summarizes the problems the software is designed to solve, and specifies how to use the software, especially for preparing input data and interpreting computed results. This section should be concise enough to enable effective use of the software and to serve as a reference source for user questions. It also includes a brief discussion of the comparative strengths and weaknesses of the software in relation to other competing or related software packages.

The application information describes

- the function of each major program option;
- the way the software handles files;
- the effect of data storage requirements on the software;
- the range of values and variables that can be used in the software;
- the control commands required to execute the program; and
- the nature of the output, including the way the output relates to the input.

The application information also provides information on how to recover from a crashed application, how to restart the application, and how to interact with the software's various data screens. This section provides basic information on how much of the computer's memory and processor power the application uses.

Functional definition

The functional definition addresses the algorithms or mathematical models used in the program. This section won't be of interest to those using the software unless users want to understand and validate the nature of the mathematical models the software is built upon. This section contains a very detailed description of the problems that can be solved with the software, including the outside limits of the software's capabilities and exactly how the software calculates and processes data.

The functional definition also describes the software's data libraries and the way the software processes were validated by the developer. The goal of this section, according to the standard, is "to provide sufficient detail to permit users to judge the suitability of any model for application to a particular situation."

The standard can even be helpful for budding software developers working in classrooms or garages.

Programming information

The programming information section is for people who implement, maintain, modify, and port software for external or local needs. While this section would likely not be included in consumer-oriented software, it is a useful part of the documentation process, especially because it can help track changes to software as it moves to different platforms or computing environments.

The programming information offers

- the language in which the program was written and details about any development tools used,
- details that would help port the software to different platforms,
- specific descriptions of the types of data files and internal devices the software uses,
- implementation requirements, and
- the control commands to install the software.

The programming information section includes complete diagrams that help illustrate a program's structure and logic, and includes flowcharts where appropriate. The programming information section also identifies ways to describe the software's network interface and data-transfer protocols.

The standard recommends including very detailed information in the programming information section about the environments on which the software has been tested, the specific memory requirements needed by the target platform, the special hardware functions the software uses, the unique networking requirements, the language processors, the subroutine libraries, and much more.

RELATED MATERIALS

The standard also makes recommendations for what should be included in the software package itself. The software package, according to the standard, is the "aggregate of all elements required for implementation or use of the software," including documentation, sample output, and transmittal material. The standard makes useful recommendations for how disks should be labeled, how the contents of the software should be identified in external documents, and how developers should include sample data sets.

There are also recommendations for an installation environment report, which—like the programming information section—would document all the computer environments on which the software has been tested.

While it might seem as if the ANSI/ANS 10.3-1995 standard is unnecessarily comprehensive in its recommendations, it is unquestionably useful. Offering a complete checklist of software and hardware details, the ANSI/ANS 10.3-1995 standard will also be useful to developers who have their own methods of software documentation. The standard can even be helpful for budding software developers working in classrooms or garages.

Programmers' workloads will initially increase when they adhere to the standard—because successful documentation takes time—but the payoff in terms of reuse and portability is clear. ❖

Vir Phoha is assistant professor of computer science at Northeastern State University, Tahlequah, Oklahoma; contact him at phoha@cherokee.nsuok.edu.