View metadata, citation and similar papers at core.ac.uk brought to you by CORE
provided by NASA Technical Reports Server

N73-19211

AUTOMATIC EDITING OF MANUALS

Dr. Robert P. Rich
Applied Physics Laboratory, Johns Hopkins University

Documentation for a computer program is usually understood to include some or all of the following items:

(1) Program listing
(2) Flowcharts
(3) Problem description
(4) Programmer's reference manual
(5) Analyst's reference manual
(6) User's manual
(7) Management information

The documentation problem that one encounters arises from the difficulty of getting all of these items prepared in a timely fashion and the near impossibility of keeping them all correct and mutually consistent during the life of the program.

A useful approach to the problem is to collect all of the necessary information into a single document, which is maintained with computer assistance during the life of the program and from which the required subdocuments can be extracted as desired.

Implementation of this approach requires a package of programs for computer editorial assistance and is facilitated by certain programming practices that are discussed in this paper. Experience shows that this approach not only provides documentation at a reasonable cost but also facilitates program implementation and management, especially for large programs requiring a team effort.

THE INFORMATION PACKAGE

The present approach to program documentation was made possible by the existence of a general-purpose information package for the management of files of textual material. This package was originally developed for document retrieval with the IBM 1401in 1962. It has been rewritten with successive improvements for the IBM 7094, CDC 3300, and IBM 360; this last version, INFO 360, is discussed in this paper. The package contains three major programs: the EDIT program for maintaining standard files, the PRINT program for printing a standard file, and the SEARCH program for selecting records from a standard file.

Preceding page blank

The Standard File

For a specific application, INFO 360 works on standard files. Each file consists of a number of standard records. Each record consists of a format code followed by a comma and then by the body of the record, which is a string of characters whose number cannot exceed 4000. The master file contains the text of the current draft of the document. Most of the records are text records, with a hyphen as format code:

-,This is a text record.

Another type of record that is important is the title record, with format code T*d*, where *d* is a single decimal digit specifying the level of the title:

T3, 37. Title at level 3.

These records are used as section headings within a document; the body of the record begins with a section number (37 in the previous example).

The EDIT Program

The EDIT program permits a master file to be established or modified in a way that is specified by a file of changes. One or more records may be inserted, deleted, or modified. The details of how the changes are specified are not of concern here, but it should be noted that a good secretary can learn to type the changes without much difficulty. The cross-reference feature of the EDIT program is important in the present application. As an example, consider the section numbers of title records. The EDIT program can be instructed to renumber all the sections in the sequence 1,2,3, . . ., throughout the file. It also corrects cross-references, replacing the old section number with the new one, so that cross-references by section numbers remain correct if the file is modified.

The PRINT Program

The PRINT program outputs a standard file in a variety of formats that are determined jointly by the format codes of the individual records and the values assigned to various print parameters. Text records are broken into lines (with hyphenation and justification) to fit the specified margins. The listing is broken into pages to fit the assigned page length; page headings and numbers are printed as requested; and footnotes and white space for figures are appropriately positioned, no matter how the page breaks fall. In addition, multicolumn printing may be specified.

The treatment of the title records is especially interesting. When the PRINT program encounters a title record, it uses this record in three ways:

(1) The body of the record is printed in the text at its point of occurrence and set off by lines of asterisks.
(2) The body of the record is saved as a page heading, to be repeated at the top of following pages until it has been replaced.
(3) The body of the record, with the current page number added, is saved for inclusion in the table of contents.

When printed, each title record is indented by an amount proportional to its level.

This treatment of title records by the PRINT program, together with the automatic correction of cross-references by the EDIT program, provides an extremely powerful cross-referencing mechanism for the complex type of document involved in the present application.

THE MONODOCUMENT

What is being proposed is that the complete documentation for a particular program be included in a single document, the monodocument for that program, and that this document be maintained with the assistance of a computer employing INFO 360 or any local equivalent. When such a program is begun, the monodocument consists of only an outline that may be in the form of title records at appropriate levels so that the indentions will preserve the correct outline form in the table of contents.

If several people are to work on the program, then the first coordination meeting might result in the assignment of responsibilities for the various sections. Each such assignment is recorded temporarily as the text of the section, to be replaced by the actual text when it becomes available. As each section is completed, it is put into the monodocument by the EDIT program; the section then becomes immediately available for proofreading and further correction by author or editor and for reference and negotiation by all members of the team. Once a section has been approved and proofread, it remains correct until changed. Hence, all people involved can concentrate on the currently active sections of the document.

As agreement is reached on such matters as file formats, subroutine specifications, and programming conventions, they are incorporated into the document. Since at any given time, each person is using a copy of the same draft, it is much easier to maintain consistency. Each programmer has the responsibility of ensuring that his part of the program remains consistent with the other relevant parts of the monodocument. The cross-reference capability makes this part of his job such simpler. If the monodocument grows in this way as the program is written, it will be completed when the program is completed, and the documentation problem will have been solved.

The Symbolic Program

The symbolic program itself is one of the major sections of the monodocument. It is easy to incorporate the symbolic cards for a checked procedure into the master file or to punch such cards from the master file. Hence, the monodocument can easily contain the program as it was checked out; in fact, it becomes the official record of the final version of the program. This is particularly helpful if the program has a long production life.

Although the general approach being discussed is fruitful for assembly language programs, the symbolic program is especially elegant when high-level languages are used because a moderate amount of commentary and proper style conventions make the program self-documenting in a very useful manner. It is assumed, of course, that the program is written in a modular fashion. Such helpful techniques as assignment of labels in lexicographic order, systematic indention to show logical levels, and explicit declaration of variables deserve more attention than is given to them in this paper. The fact that program commentary can contain cross-references to other sections of the monodocument is potentially very helpful but has not yet been exploited.

Flowcharts

A set of flowcharts is a traditional part of the program documentation package. Recent experience indicates that a program properly written in a high-level language, especially when cross-referenced to an appropriate functional description, is easier to understand than the flowcharts that purport to describe it. Those who hold this view would ignore flowcharts in the monodocument. (This opinion does not apply, of course, to the informal, working flowcharts that the programmer uses while he is writing the program.) However, if they are required, they can be prepared by AUTOFLOW after everything else has been done. Those who still feel that final, hand-drawn flowcharts are worth what they cost will, of course, continue to produce them; that job will not be made more difficult if the monodocument approach is used for the rest of the package.

Problem Description

Problem description means a description of the problem the program is to solve, the function it is to carry out. This section of the monodocument will obviously be one of the first ones actually written, although it may be modified from a brief qualitative description to a detailed technical specification in some instances as the work progresses.

Programmer's Reference Material

The programmer's reference material portion of the monodocument contains information of interest to a programmer who has to correct, modify, or explain the program. It is complementary to (and can be cross-referenced to) the symbolic program.

Analyst's Reference Material

The analyst's reference material includes details of interest to the user that are not included in the user's manual. For example, the convergence characteristics of the numerical algorithms and the nature of the approximations used are among the items included. Some material could equally well be placed here or in the programmer's reference material.

User's Manual

The user's manual includes the material needed to use the program: input formats, control cards, file designations, alarm messages, restrictions on ranges of input variables, etc.

Management Information

Management information, such as time logs of personnel assignments, computer time for checkout, and comparisons between original estimates and actual performance, can easily be kept current as the monodocument is periodically updated.

DISCUSSION

MEMBER OF THE AUDIENCE: What can be said of the system's ability to produce machine-readable charts, tables, and illustrations? Many document programs have a need for them.

RICH: Those displays that consist essentially of computer printout, such as AUTOFLOW charts, could very easily be imbedded in the document and updated in the same manner as text. Those displays that are produced by a plotting device or by a draftsman would be incorporated into the document when it is bound.

MEMBER OF THE AUDIENCE: Some of the IBM systems that are available have languages for producing machine-printable charts, illustrations, and diagrams that are not flowcharts.

RICH: The system that I discussed, as well as a number of other systems, have features that were not covered in my presentation. For instance, our system will not only accept tables but will even perform the arithmetic of tabular work.

Currently, we have over 200 pages of documentation for our system. For instance, the PRINT program has approximately 30 different format codes that indicate the different types of records: text and heading records, indexing records, records that leave space for figures and captions, etc.

MEMBER OF THE AUDIENCE: How does your approach handle the engineering aspects, such as representation of algorithms in text? A very important part of scientific programming is the ability to display the equations being used.

RICH: There are essentially two solutions to this. Space can be left in the text so that typed equations can be stripped into the document. This is quite troublesome. I take the view, however, that if we are going to achieve the documentation of a program, the problem is best described in a programmable notation. If the engineer wants an alpha, then I spell ALFA; if exponents are required, then two asterisks should be used, or some other representation that would depend upon the programming language. If an algorithm is clearly defined for the computer and an appropriate language is being used, the engineer can easily verify his statements. It is best to help the engineer write his formulas in a programmable language.

MEMBER OF THE AUDIENCE: Is it your feeling that another individual, a documentation specialist, should be working with the programmer and handling all of the evolutionary documentation?

RICH: When I work on a program, I take personal responsibility for the documentation. For the usual systems team (project engineer, physicist, analyst, programmers, etc.), a secretary and an individual willing to take responsibility are needed in order to achieve good documentation.

MEMBER OF THE AUDIENCE: What are the advantages of using the computer for documentation instead of an MTST or similar device?

RICH: The computer generates text that is truly machine-readable; this is not always the case with MTST's. For updating text efficiently, the computer approach is far superior.