Towards a software index - standards of program documentation

IABSE
AIPC
IVBH

COLLOQUIUM on:
"INTERFACE BETWEEN COMPUTING AND DESIGN IN STRUCTURAL ENGINEERING"
August 30, 31 - September 1, 1978 - ISMES - BERGAMO (ITALY)

Towards a Software Index — Standars of Program Documentation

Répertoire de logiciels — normes pour une documentation de programme

Software Index — Richtlinien für eine Programmdokumentation

M.V. JONES

National Executive Engineer
ACADS
Melbourne, Australia

1 INTRODUCTION

It is perhaps unfortunate that computing is still something of a glamour industry and programs are often written with little forethought or planning. This enthusiastic zeal to produce a working program does not generally extend to the documentation, which is frequently an afterthought. Indeed, it has been advocated that the User Manual should be finalised and printed before any program coding starts.

Not only is this neglect of documentation unprofessional, it is also short-sighted, since the "user" documentation is the primary means of assessing a program's worth. Increasingly, this is being realised and users are rightfully demanding adequate standards for Program Manuals; however this cannot be achieved without cost. Reti (Ref.1) states that "undocumented programs can be used only by their writers, and even they become rather uncertain after some time".

This plea for good documentation does not in any way detract from the other requirements of a computer program, namely that it be accurate and reasonably efficient. Indeed, a well documented program engenders confidence in a user and helps to ensure that he obtains his results expeditiously. Nor does good documentation in itself obviate the need to have access to expert professional advice on such aspects as modelling of the "real" problem to fit a computational solution. This also extends to interpretation of results such that they can be used correctly.

In order to make the best use of computing aids, engineers require to know what programs are available and, more importantly, which are reliable and well supported. Thus the need is for a comprehensive software index containing meaningful details of programs under standard headings.

2 EXISTING PUBLISHED DOCUMENTATION STANDARDS

(a) ACADS

The Documentation and Checking of Computer Aided Computations has been discussed previously by the author (Ref.2) and the ACADS publication mentioned therein (Ref.3) has since aroused world-wide interest. Some 600 copies have been distributed and the document is referenced in Amendment No.4 (1976) of the Victorian Uniform Building Regulations. When this is gazetted and becomes law later this year, the Document will have the same standing as an SAA Code. The document is also being considered for reference in the proposed Australian Model Uniform Building Code (AMUBC).

(b) A.C.I. - American Concrete Institute

The A.C.I. Committee 118 recommendations (Ref.4) whilst being commendable, do no more than outline recommended documentation. Turk (Ref.5), in commenting on the A.C.I. recommendations, advocates that the engineer responsible for a program should seal the program listing, saying in essence "This program, as herein listed, will do what it purports to do, within the limitations and restrictions stated".

(c) A.S.C.E.

The A.S.C.E. publication (Ref.1) details recommended standards for four suggested levels of Documentation.

(i) Program Identification
(ii) Engineering Documentation
(iii) System Documentation
(iv) Operating Documentation

These differ somewhat from the recommended ACADS Standards (Ref.6) and level (iv) above would seem to reflect the large predominance of IBM 1130 use among North American engineers. Rouse, Longinow and Chiapetta (Ref.7), in commenting on the A.S.C.E. standards, suggest that the recommendations are too broadly worded and that a more specific checklist should be provided. The ACADS Standard (Ref.6) reflects this philosophy for the one level of documentation detailed therein. The Authors (Ref.7) go on to state that in order to survive in the real world of budgetary and time constraints and of stubborn programmers set in their ways, a programming standard must:

(i) Provide specific subject headings or

questions to encourage specific answers
(ii) Avoid repetitious questions
(iii) Encourage a document-while-you-work or "self-documenting" philosophy.

(d) U.K. Experience

A joint Computer Committee has been established in the U.K. (Ref.8) with a brief to examine all aspects of computer applications in the civil engineering industry. Their first task is to decide on the form of documentation standards, and these are due for publication soon.

The Institution of Structural Engineers has also established a Special Study Group on Computer-Aided Design and Numerical Methods (Ref.9). One of their tasks is to update and publish the National Computing Centre Software Index of Structural Programs (Ref.10).

3. EXISTING SOFTWARE INDEXES

Numerous Software indexes exist (Refs. 10-15) but these are usually not rigorously maintained and updated. Access to the indexes is often difficult and deters engineers who want a reasonably prompt response to their enquiry. A search time turnround of 1/2 day - 1 day would generally be considered satisfactory by a potential program user. However, an on-line enquiry system would be ideal.

Another serious defect of existing indexes is the sparse or useless information provided and the lack of indication as to what documentation exists for a particular program. This latter was highlighted in a recent CEPA Report (Ref.16) which revealed that, in reviewing 39 Software catalogs containing 5213 engineering programs, no less than 99% of programs were deficient in information about their state of documentation.

4 NEED FOR COMPREHENSIVE PROGRAM DOCUMENTATION STANDARD

From the preceding section it can be seen that, although many indexes exist, the information contained therein is very often not of much use. This is because there is no comprehensive, and generally acceptable, standard for program user documentation. The ACADS Document (Ref.6) is aimed at providing a workable standard for such documentation.

5. ACADS PUBLICATION (TECHNICAL NOTE DO/1) - HISTORY

The initial drafts of the ACADS Policy Document 74/1 (Ref.3) contained, as an Appendix, details of a one page "Program Résumé". This was also referred to in the conference paper (Ref.2).

However, during further Committee discussion, it became evident that the one page Résumé would not be appropriate and that it should be superceded by three separate one page Abstracts. Details of two of these were included as Appendices 1 & 2 of the Policy Document 74/1 (Ref.3) published in August 1974.

Technical Note DO/1 was published in January 1975, and details all three Abstracts in addition to outlining the four recommended levels of Documentation. The Note was revised and reissued in March 1976.

6 ACADS TECHNICAL NOTE - PHILOSOPHY AND CONTENT

There should be four types, or levels, of documentation - each one suited to a particular degree of interest. The four levels are:

(i) Program Description - a few lines
(ii) Program Abstracts - three types, of one page each
(iii) User Manual
(iv) Full Documentation

All programs should have at least the first three levels of documentation and ideally the fourth too; however this is often not achieved in reality. Nos.(i)-(iii) are vital for the potential user to determine the most suitable program for the particular application.

A comprehensive User Manual is essential for the program to be used with professional competence. Numerous erroneous computer runs occur through the use of out-of-date manuals. User Manuals should be dynamic documents, with regular Revisions to correct typographical errors, ambiguities and to reflect improvements in program capability. Publication and, more importantly, maintenance of manuals costs money and users must be prepared to pay realistic prices for this service. This they are willing to do, provided that lower levels of Documentation (Nos.1-2) exist and the purchase of a User Manual is necessary only after thorough perusal of the Abstracts.

A Software index could consist of Program Descriptions or Abstracts depending on user requirements and the configuration of the computer system used to retrieve them. However, the ability to retrieve both the Descriptions and Abstracts would be best.

(a) Program Description

This is a brief summary of a few lines suitable for inclusion in a comprehensive program list. For use as a software index the description should consist essentially of keywords and ACADS has published (Ref.18) recommended formats for storing these on standard 80 col. punch cards. The ACADS Description includes an ACADS reference number, which provides the link to the Abstracts.

(b) Program Abstracts

These are three distinctly separate items, each preferably not more than one A4 page in length. In practice, for all the Abstracts already collected (nearly 150) it has been found possible to adhere to this restriction.

The three abstracts are called:

(i) Program User Abstract
(ii) Program Status Report
(iii) Program Implementation Abstract

The first two are of interest to a program user, whereas the third is of interest only to a programmer, or someone who wishes to consider mounting the program.

The concept of the one page Abstract is not new, and indeed seems to be generally accepted by many organisations. However other authorities usually see it as just one Abstract to be used by users and programmers alike. It is ACADS experience that one page is insufficient to provide the required amount of meaningful information. Furthermore, it is important not to confuse a potential user with details which are of interest only to a programmer.

(i) Program User Abstract

This is the most important of the three and contains details of the technical content of the program. From this a potential user should be able to judge the likelihood of the program's applicability to a particular problem. Appendix A gives the details recommended for inclusion in a User Abstract.

In practical use, it has been found that after thorough perusal of a User Abstract the enquirer is very sure as to whether he definitely does, or does not, wish to proceed to the next level of Documentation, i.e. the User Manual. The information provides him with an easy clear cut decision, and this is especially important for, not only is a charge made for most User Manuals, but a reasonable investment of time is needed to study the contents. This contrasts with a one-page Abstract which can be studied in under 5 minutes.

The User Abstract is essentially an objective and factual document which should only require revision when a new program version is introduced, i.e. new enhancements or user features are introduced. Program amendments of a less significant nature than these are termed updates, and it is essential that all programs are precisely identified by both version and update number.

(ii) Program Status Report

This is a one page report on the current reliability of the program, comments on its User Manual and amount of usage and a rough guide to costs. Some of the contents of the Status Report are less definitely objective chan the User Abstract. Comments on User Manuals, in particular, are subjective and even run costs can be a cause for debate.

However, although the User Abstract is the most vital of this trio of one-pagers, it is the Status Report which is probably of most interest to the potential user. From it, he can get more of a "feel" for a program, particularly its ease of use (from comments on the user manual) and amount of usage to date on "live" jobs. Appendix B gives the details recommended for inclusion in a Status Report.

A Status Report refers to a particular copy of a program on a specific computer and other versions, even if "identical", on other machines, require their own Status Reports. The Report reflects every program modification, as witnessed by a changing program identifier, and is thus very dynamic in nature. The Report and program should be updated together and thus it is best stored on a file together with the program and listed every time the program is run. As with the other Abstracts the Report should confine itself to one A4 page, but with several Update Records (see Appendix B) this will not be possible unless only the latest updates are detailed.

It is desirable for the Status Report to be issued by an organisation independent of the program maintenance organisation or Licensee. ACADS is providing a role in this area.

(iii) Program Implementation Abstract

This Abstract is for a programmer to assess the requirements for implementation of a program. It also indicates the availability, cost and form in which the program is supplied. Appendix C gives the details for an Implementation Abstract.

(c) User Manual

This is to enable the user to code the input, interpret the output and to satisfy himself of the technical content of the program. For many programs it may be suitable to have two manuals: A User Guide and User Reference (or Theoretical) Manual. ACADS aims to publish a Technical Note recommending details to be included in User Manuals.

(d) Full Documentation

This would include full details of the program listings, block and flow diagrams, special equipment requirements, details of checking and testing and a complete record of all program modifications. It could be termed the Programmer's Manual. It is again ACADS intention to publish recommendations on this item.

7 CONCLUSIONS

Inadequate documentation and the lack of usable standards is inhibiting the professional use of computers in engineering. ACADS has published such a standard for the first two levels of primary user documentation and has already collected details of over 100 programs (Ref.17) in the recommended format. This is the first stage towards establishing a Software Index, and program authors and maintenance organisations are urged to document their programs in accordance with the published standard. This will help to ensure their wider publicity and reduce the wasted effort involved in needless duplication.

8 ACKNOWLEDGEMENTS

Acknowledgement is made to the Council of ACADS for permission to publish this paper and extracts from the ACADS publication on Documentation. Recognition must also be given to the Committees and individual members of ACADS whose formulatory work and constructive criticism have contributed to the preparation of that publication.

9 REFERENCES

1. RETI, G.A. - Engineering Computer Program Documentation Standards. Journal of the Soil Mechanics and Foundations Division ASCE, SM3, March 1973, pp. 249-266.

2. JONES, M.V. - The Documentation and Checking of Computer Aided Computations. Proc. National Conference on Computers in Engineering, Sydney 1974, pp. 116-119.

3. ACADS (Association for Computer Aided Design) - Recommended Standard for Documentation and Checking of Computer Aided Engineering Computations. Policy Document 74/1. ACADS, Melbourne, August 1974.

4. BERWANGER, C. - Recommended Documentation for Computer Calculation Submittals to Building Officials. ACI Journal, March 1973, paper 70-16.

5. TURK, A.R. - Computer Documentation. ACI Journal, May 1975, pp. N27-28.

6. ACADS - (Technical Note DO/1),Recommended Minimum Program Documentation and Details of Abstracts, Status Reports. ACADS Melbourne, January 1975.

7. ROUSE, J.D., LONGINOW A. and CHIAPETTA, R.L Engineering Computer Program Documentation Standards. Journal of the Soil Mechanics and Foundations Division - ASCE, SM12, December 1973, pp. 1174-1175.

8. NEW CIVIL ENGINEER - Computer Programs Documentation to be standardised. New Civil Engineer, 13th February 1975. pp. 53-54.

9. INSTITUTION NOTES - New Special Study Group. The Structural Engineer, February 1974, No.2. Vol.52,p.65.

10. NCC - Software Index. The National Computing Centre Limited (NCC)-U.K.

11. INGLIS, E.D. - A PRogram INDEX system for retrieval of information on software. PRINDEX, Univ. Melbourne Computer Centre, July 1975.

12. CETIS - Computer Program Index. COPIC, Ispra Italy, April, 1973.

13. AASHTO - Computer Systems Index. AASHTO, U.S.A., July, 1974.

14. BUSHNELL, D. - COMSTAIRS, A computerised information retrieval system for structural software. Structural Mechanics Computer Programs, Univ. Press of Virginia, U.S.A., 1974.

15. EWALD, R.H. - Computer Based Technology Transfer. ASCE - 6th Conference on Electronic Computation, Aug. 7-9, 1974, pp. 742-761.

16. CEPA - A Proposal for a National Institute for Computers in Engineering (NICE). CEPA, Rockville Maryland U.S.A., October, 1975.

17. ACADS - Computer Programs for Concrete Structures, 1975 Survey. ACADS Report No.4 ACADS, Melbourne,February, 1976.

18. ACADS - Program Descriptions using key words - Format for storing on Punched Cards. Technical Note DO/2, ACADS, Melbourne, March, 1976.

APPENDIX A

RECOMMENDED DETAILS FOR A PROGRAM USER ABSTRACT

PROGRAM USER ABSTRACT Date of Abstract

PROGRAM NAME: A 4-6 character code name to identify the program and date of first issue e.g. FRAMEM-OCT73.

Version No. A new version number is required when new enhancements or user features are introduced.

TITLE: A title, not more than 80 characters long (one punched card), describing the problem solved by the program, e.g. STATIC ELASTIC ANALYSIS OF FRAMES - STIFFNESS METHOD.

MAINTENANCE: Program Maintenance Organisation - This is essential to differentiate between proliferated versions of a program.

ACCESSIBILITY: Computer systems where the program is mounted.

CAPABILITY & LIMITATIONS: The purpose and range of applicability; special features and limitations; units and capacity limits.

INPUT: Summary of required input.

OUTPUT: Description of amount and form of output, including input data, intermediate and final results.

METHOD: Brief description of theory, including references where applicable; design codes used etc.

LANGUAGE: Program Language - precise definition: e.g. Fortran - AS1486, 1973. Indicate if the language is associated with a manufacturer or a specific computer: e.g. IBM 1130 Fortran.

AUTHORS: Program Authors, organisation.

DOCUMENTATION: Indication of existence of further documentation, e.g. Status Report, Implementation Abstract, User Manual, Programmer's Manual etc.

INFORMATION: The address and telephone/telex number of the contact for further information.

KEYWORDS: For classification and retrieval purposes.

PREPARED BY: The name of the organisation (and person) preparing this User Abstract.

CONFIRMED BY: The name of the organisation (and person) which has confirmed the accuracy of the contents : this would usually be the Program Maintenance Organisation or Author.

APPENDIX B

RECOMMENDED DETAILS FOR A PROGRAM STATUS REPORT

PROGRAM STATUS REPORT Date of Report

PROGRAM NAME: A 4-6 character code name to identify the program and date of first issue, e.g. FRAMEM-OCT73.

Version No. A new version number is required when new enhancements or user features are introduced.

TITLE: A title, not more than 80 characters long (one punched card), describing the problem solved by the program, e.g. STATIC ELASTIC ANALYSIS OF FRAMES - STIFFNESS METHOD.

MAINTENANCE: Program Maintenance Organisation - This is essential to differentiate between proliferated versions of a program.

COMPUTER: The particular computer where the program is mounted.

USER MANUAL: Publication No./Date of issue - for precise identification. Comprehensiveness, comprehendibility, ease of use/familiarity. Range of examples included and provisions for updating.

CHECKING: References on checking of the source program.

TESTING: References to test examples run and especially any comparisons with other programs or independent testing - e.g. ACADS Frame Report 1974.

USAGE: Rough guides as to how much usage program has had on actual engineering jobs.

UPDATE RECORD: Any change to a program should be indicated in the program identifiers. A new update number is used to denote bug corrections, minor changes to output formats and small internal amendments. New enhancements or user features (including changed input formats) require a new version number. This record provides a history of program changes and known bugs and these are ordered by update No. with the most recent update first.

UPDATE No. - Date of Update

Program Changes - program changes as they may affect the user.

Known Bugs - Bugs reported, including those from previous updates not yet corrected.

RUN COSTS: A rough guide to the cost of runs on this particular computer installation.

PREPARED BY: As on User Abstract.

CONFIRMED BY: As on User Abstract.

APPENDIX C

RECOMMENDED DETAILS FOR A PROGRAM IMPLEMENTATION ABSTRACT

PROGRAM IMPLEMENTATION ABSTRACT Date of Abstract

PROGRAM NAME: A 4-6 character long code name to identify the program and date of first issue, e.g. SAPIV-JNE73. Version No. A new version is required when new enhancements or user features are introduced.

TITLE: A title, not more than 80 characters long (one punched card) describing the problem solved by the program, e.g. GENERAL FRAME ANALYSIS - STIFFNESS METHOD. LINEAR ELASTIC STATIC AND DYNAMIC.

MAINTENANCE: Organisation responsible for co-ordinating bug reporting and corrections. Name, address and tel./telex no.

UPDATE: Update No. to identify the program precisely. A new update No. denotes any change to a program (other than a new version), e.g. bug corrections, minor changes.

COMPUTERS: The computer system on which the program is maintained and the other systems where it has been implemented.

LANGUAGE: Program Language - precise definition, e.g. Fortran ASl486, 1973. Indicate if the language is associated with a manufácturer or a specific computer, e.g. IBM 1130 Fortran.

EQUIPMENT: Minimum configuration required with pertinent details of:

CORE requirements incl. details of word length. CARD/PAPER TAPE readers, LINE PRINTERS, LOW SPED TERMINALS, PLOTTERS, MAG. TAPE/ DISC UNITS.

RUN TIMES: Typical times of a range of problems on different systems.

AVAILABILITY: CONTACT: Name, Address and tel./ telex No.

COSTS: Details of costs and cheque payment/currency.

PREPARED BY: As on User Abstract.

CONFIRMED BY: As on User Abstract.