SOFTWARE DOCUMENTATION AND STANDARDS

J M JOSE
T VISHWANATHAN
INSDOC
New Delhi

Presently, software costs dominate the hardware costs in computer systems. Desire to produce high quality reliable software at low costs has led to the evolution and adoption of engineering principles in the design and development of software. This paper deals with software documentation and standards which occupy an important place in software engineering process. The need of a software librarian as a part of software engineering team is discussed. International standards in software engineering deal with only guidelines. Evolving a standardisation process for an organisation using these guidelines is dealt with. Finally some quality indicators for software documentation are presented.

INTRODUCTION

The last decade of computing recognised software as a new driving force and, in fact, software surpassed hardware as the key element to the success of many computing systems. But the dual problem of high cost and low quality of software remains the main concern of the software engineering community even today. Quality is sacrificed because of cost constraints, or costs are driven up in order to achieve higher levels of quality. Deficiencies ignored during the development stage result in huge costs during the maintenance phase of software systems. It is claimed by Boehm[1] that more than one-half of the total life-cycle costs are incurred during the maintenance phase.

During the past decade software development has evolved into an engineering discipline. Software engineering techniques deal with software as an engineered product that requires planning, analysis, implementation, testing and maintenance. Although solutions to current problems in the domain of software have advanced with the acknowledgement of life-cycle perspective of the development process, management’s lack of commitment to educate existing talent in the use of evolving methodologies, techniques and tool continu-
es to contribute to high-cost, low-quality software.

Citing the excessive software maintenance costs, the software engineering community has been able to explain the need for a disciplined development process, controlled through meticulous attention to configuration management and governed by a set of fundamental principles. In practice, these principles are associated with the objectives for a given software development activity and are measurable through associated attributes of the product.

Among these principles is the software documentation: i.e. the recording of the requirements, design, specifications and implementation decisions as they occur with the commitment to convey purpose, content and clarity.

SOFTWARE DOCUMENTATION

Software engineering techniques facilitate an engineering approach to software development. The classical life cycle approach goes through different phases such as requirements specifications, design, coding, testing and maintenance during system development. The documentation of all these activities is necessary. A document is defined as a written record of the completion of a phase of work. Software documentation is a major component of software engineering activity as depicted in Fig. 1. Software documentation comprises the manuals and other descriptive information that portray the use, operation and other major components of the software.

The purposes of software documentation may be categorised into:

- Inter-task/phase communication
- Quality control and project control
- Historical reference
- Instructional reference
Documentation serves as a communication bridge between the user, the designer and the builder of the system as shown in Fig. 2. It has long been recognised that poor or inadequate communication among personnel is a major problem area in software development process. The system analyst communicates with many user personnel during the investigation and analyses of the problem area. The system analyst in turn communicates the program requirements to the programmer, who must specify how the computers must run the program and so on. Without adequate documentation, the initial idea from the user becomes distorted resulting in the final product bearing little relation to the original intention. Documentation is important for quality assurance. As the product proceeds from concept to general design and ultimately into operational form, the only viable way to assess the product is through its documentation. Documentation functions as a source for historical reference and the provisions of a permanent record of work performed and used is an important reference for future modifications. Documentation is also used as an instructional device like user instructions or operator instructions.

A simple model of documentation process is shown in Fig. 3.

The first step in the documentation process is planning the document. It consists of three main steps:
- Define the audience, purpose, and subject
- Organise the material
- Design the format

The second step in the documentation process is to create a draft, or working version of the document that has been planned. Once the draft is finished, the next step is to edit it. The purpose of editing is to refine the draft into a usable systems document. When a useful version of the document has been drafted and edited, it has to be reviewed. All documents need to be tested against their document plan to make sure that they meet the criteria defined there. Once the document has been reviewed and perfected, it can be produced in a form that is suitable for distribution to the intended audience. Since information systems are more susceptible to change, revision and updating of system documents are important steps of the documentation process. The most important factors in good document design are:

- Orderly development of ideas
- Technical accuracy and completeness
- Logical arrangement of information

Software documentation can be defined as a collection of technical data or information, including computer listings and printouts, in human readable form. This collection describes or specifies the design details, explains the capabilities, or provides operating instructions for using the software to obtain desired results from a software system.

There are standard procedures available for the documentation of activities done during different phases of software development process. The changes made to the system during a particular phase of development should be reflected in all the related documentation components. This is known as concurrent documentation. The principles of concurrent documentation of software implies a continual change to all documentation elements affected by changes at any stage of development, as depicted in Fig. 4. Concurrent documentation is particularly important within the maintenance phase because examination of existing documents may be the only means available to the maintenance personnel to understand the internal details of the software and the original intentions of the developers. As with other software engineering principles, concurrent documentation is crucial to the software development process, and later, to the maintenance activity. Hence, the extent to which concurrent documentation principle is observed must be measurable and controlled. This leads to the problem of determining how one might evaluate the quality of documentation produced during the software development and maintenance phases.

Software documentation can be classified into system documentation and user documentation as depicted in Fig. 5.

System documentation conveys the requirements, design philosophy, design details, capabilities, limitations and other characteristics of a system. User documentation conveys to the end user instructions for using the system to obtain the desired results, for example, a user's manual. System documentation, in addition, can be categorised into four applicational areas as shown in Fig. 5.

**Software Requirement Analysis documentation**

Requirement analysis is the process of allocating software function, in the overall system development process. It enables the system analyst to
VISWANATHAN AND JOSE

Figure 4

Software Documentation

User Doc.

System Doc.

Requirements

Design

Program

Testing

Figure 5

Plan

Data structure

working

Req. Spec.

Design

Listing

Test Spec.

Figure 6
specify software function and performance, indicates software’s interface with other system elements, and establishes design constraints that software must meet.

Software Requirements Specification is produced at the culmination of the analysis task. The function and performance allocated to software as part of system engineering are refined by establishing a complete information description, a detailed functional description, an indication of performance requirements and design constraints, appropriate validation criteria, and other data pertinent to the requirements. In general a requirement document should include:

Introduction It states the goals and objectives of the software, describing it in the context of the computer based system.

Information Description It provides detailed description of the problem the software must solve.

Functional Descriptions A description of each function required to solve the problem is presented.

Validation Criteria It specifies what classes of tests must be conducted to validate function, performance and constraints.

Bibliography and Appendix Bibliography contains references to all documents that relate to software. The appendix contains information that supplements the specifications.

ANSI/IEEE Std 830-1984 specifies in detail what are the components of a good requirements document. The guide describes alternate approaches to good practice in the specification of software requirements. To enable the reader to make knowledgeable choices, extensive tutorial material is provided. This covers the attributes of a good software requirements specification itself, as well as specification methodologies and associated formats.

Design Documentation

Software design is a process through which requirements are translated into a representation of software. Software design is conducted in two steps. Preliminary design is concerned with the transformation of requirements into data and software architecture. Detailed design focuses on refinements to the architectural representation that lead to detailed data structure and algorithmic representation for software.

A design document consists of scope of software, reference documents, design description, modules, file structure, global data, requirements cross-references, test procedures, etc. The documents should specify the necessary information content. A software design document is a representation of a software system that is used as a medium for communicating software design information. ANSI/IEEE Std 1016-1987 specifies an organisation for a software design description.

Program Documentation

Program documentation comprises the records of the detailed logic and coding of the constituent programs of a system. It is prepared by the programmer and aids:

- Program development and acceptance
- Troubleshooting
- Program maintenance
- Machine conversion
- Programmer change over

For the purpose of program documentation, the following guidelines are useful:

- heading comments for the main program and each subroutine to provide the reader with an overall information about the program structure and logic, and with a concise description of the functions of each component.
- current comments strategically placed within the program to identify logical divisions or blocks, to clarify programs statements, and to point out deviations from the standard programming language, if any.

Program documentation should contain details such as: who developed the code and when, who made the changes and why, what changes are made and when in a chronological order.

Test Plan Documentation

The preparation of a program test plan is a fundamental technique of program validation methodology. A test plan is essentially a schedule of sequence operations to test a program. The test plan incorporated in the program manual should nominally comprise:

- A summary of method
- List of test cases, sequence of appli-
Figure 7

Figure 8
SOFTWARE DOCUMENTATION AND STANDARDS

cations and expected results
Listing of a test data

ANSI/IEEE Std 829-1983 specifies the contents of a test documentation. The standard addresses the documentation of both initial development testing and testing of subsequent releases. The documents outlined in this standard cover test planning, test specifications, and test reporting. The test plan prescribes the scope, approach, resources and schedule of testing activities. It identifies the items to be tested, the features to be tested, the testing tasks to be performed, the personnel responsible for each task, and the risks associated with the plan. The standard shows the relationships of these documents to one another as they are developed, and to the test process they document.

SOFTWARE DOCUMENTATION AIDS

The most common type of software documentation aid is the automatic flowchart generator or flowcharter. These fall into two main groups. The first group operates on ordinary source programmes together with a few special parameters to produce detailed logic flowcharts. The second group accepts special parameters plus a program description and from these, an outline flowchart is produced. The first group of flow charts are called post coding flowcharts, and are valuable aids for program maintenance. The second group of flow chart can be called as precoding flowcharts. A flowcharting package can also produce reference listings.

Automatic Software Documenter is one such package developed by the firm COSMIC, Georgia and works on HP-9000 systems. This software provides automated drafts of documentation directly from C source code in the Unix environment. This is specifically designed to simplify and streamline program documentation. BOS/HLP is another package which helps to create on-line documentation interactively. Documentation Aid from Allen systems group generates and helps to maintain job documentation, provides on-line retrieval, and cross-referencing. All flowcharters available transcribe the sequential program flow. Some, however, have the option to present the logic in a different form. For example, AUTOFLOW from Applied Data Research Inc. automatically analyses the program, determines which logical units of code are related, and, whenever possible, places these blocks of logic together on the chart, regardless of their input sequence. DOCUMATIC from Data usage Corporation, USA does not produce detailed flow chart as such but an outline system pictorials(inputs, outputs, and process box) togetherness with a narrative description of processing.

USER DOCUMENTATION

User documentation is the hardcopy or on-line documentation provided with an application program. It tells the user how to get the program up and running, trains the user in its operation, and provides reference information once the user becomes experienced.

User documentation is extremely important. It will make the difference between whether a program can be used effectively or not. User documentation is increasingly becoming a selling point for software products.

User Documentation development process consists of the following steps[3].

1. Form the documentation development team
2. Make user documentation match
3. Prepare documentation plans
4. Develop documentation standards
5. Develop management plan
6. Create documentation
7. Conduct technical review
8. Conduct user evaluation
9. Create final draft.

The larger and more complex the program, it is more likely that it will need several separate documentation components to fulfill various needs of its audience. The documentation developer must decide what components are needed in the function to be fulfilled by each.

User documentation comes in two forms: internal and external. Internal documentation consists of help screens, directions and other explanatory information that the operator can access within the program. External documentation is outside the program itself. Its most common form is user's guide.

All user documentation has the same purpose: to explain the features of the program and help the operator gain proficiency in using it. The most common forms of external documentation probably are the program user's guide, quick reference card, quick reference guide, and job performance aid. The most common forms of internal documentation probably are the help screen, tutorial and guided tour.

A user documentation system is a set of interdependent documentation components designed to

Vol 39 No 1 Dec 1992

129
support the user in learning and using the program. The various components have different purposes and support this goal in different ways, but all, in one way or another support the ultimate goal.

The major components are

- On-line help (provide fast access to key commands and procedures)
- Tutorials (trains new users)
- Reference manual (provides detailed reference information)
- Job performance aid (summarizes key procedures)
- Guided tour (provides program overview)
- System set up information (tells how to configure program)
- Quick reference card (provides key reference information)

ANSI/IEEE Std 1063-1987 is the standard for Software User Documentation. This standard provides minimum requirements on the structure and information content of user documentation. It deals primarily with the technical substance rather than the style. Users of the standard may develop their own style manual for use within their organisations to implement the requirements of this standard.

SOFTWARE LIBRARIAN

Concurrent documentation facilitates consistency in documents. During maintenance phase and other phases of development process, the changes attributed to one document should reflect in other documents also. Otherwise, inconsistency will result and defeats the purpose of documentation.

To ensure the quality and reliability of the software, one will have to be authorised to monitor and control the changes attributed at the maintenance phase of the software development. The answers to the questions, such as:

- How does an organisation identify and manage the many existing versions of a program (and its documentation) in a manner that will enable change to be accommodated efficiently?
- How does an organisation control changes before and after software is released to a customer?
- How does one ensure that changes have been made properly in all related components?

lead to the concept of a software librarian or configuration manager. The outputs of a software engineering process are collectively called software configuration, as depicted in Fig.6.

Software configuration management is a set of activities developed to manage changes throughout the software life cycle. It is a software quality assurance activity. ANSI/IEEE Std 828-1990 gives the standard for software configuration management planning. The purpose of this guide is to provide guidance in planning software configuration management (SCM) practices that are compatible with ANSI/IEEE Std 828-1983. The guide focuses on the process of SCM planning and provides a broad perspective for understanding software configuration management [4].

In a software development process a software librarian is responsible for keeping all the documents related to a software development process. He is a part of the software maintenance/development team (Fig.7) Whenever changes are made to one document, software librarian is responsible for making the related changes to all documents. He constantly interacts with maintenance staff, development teams, system supervisor, review team, testing team etc. The librarian acts as a controller, coordinator and potentially an evaluator of the software configuration.

The responsibilities of a software librarian are to maintain custody of programs, files, documentation, magnetic media; help collect and format software productivity data, catalogues and indexes reusable and reusable software module, and assist the team in research, evaluation, and document preparation.

He is also responsible for concurrent documentation, i.e. issue of these resources based on authorized schedule of use or special authorisation. He schedules changes to production applications after completion of testing and maintains records of the changes.

Even though a large number of software packages are available for document preparation and maintenance there is no software available to ensure the changes made in one document automatically leads to changes in other related documents.

INTERNATIONAL STANDARDS

For ensuring quality of software products, standards organisations such as IEEE and ISO have come up with standards for different software engineering activities. A list of the latest standards
from IEEE[2] are given in Appendix I.

The Standards from International Standards Organisation (ISO) give another set of guidelines for software documentation. The standard ISO 9000 is for quality systems and gives a model for quality assurance in design/development, production, installation and servicing. It has three parts:


ISO 9000-3:1991 (E) sets out guidelines to facilitate the application of ISO 9001 to organisations developing, supplying and maintaining software.

BS ISO/IEC TR 9294:1990 - Information Technology - Guidelines for management of Software Documentation gives the specific guidance on the Management of software documentation to those managers responsible for the production of software or software based products.

DEVELOPMENT OF DOCUMENTATION STANDARDS

There is no universal documentation system which is valid for all environments. Development of documentation standards should, to a large extent, be a process of adaptation rather than origination. The standards available from IEEE and ISO act as guidelines. For example, ANSI/IEEE Std 1002-1987 explains the Taxonomy used in software engineering standards, their functional and external relationships and the role of various functions participating in the software life cycle. This Taxonomy may be used as a method for planning the development or evaluation of documentation standards for an organisation. There are three phases in evolving documentation standards[5]:

- preparatory phase
- development phase
- post implementation phase-audit, maintenance and support.

Three major preparatory tasks in the preparatory phase are:

- workout project organisation and allocate resources.
- define scope of standards program.
- define the architecture or organisation of manuals.

Development phase as such contains the actual standard development process. It is necessary to:

- provide the groundwork for development of each standard.
- determine the formal and informal practices in effect.
- gather data for comparison of the various approaches or possible solutions.
- establish priority of needs and probability for successful implementation.

It also involves the draft preparation, distribution of draft etc. In post-implementation phase, procedures must be formulated to review and maintain the standards.

The output of the standard developments program should be:

- A manual for documentation standards
- A maintenance and support program
- A training and implementation scheme

DOCUMENT QUALITY INDICATORS

Arthur & Stevens[6] described a method for evaluating the adequacy of predefined, static sets of documentation relative to the system that it purportedly describe. The primary, high-level qualities of good documentation are: accuracy, completeness, usability and expandability. Qualities are the abstract characteristics of adequate documentation, yet, essential to its definition. As shown in Fig. 8, qualities are the nodes directly under and most closely tied to the even more abstract notion of adequate documentation.

Although intangible, the identified qualities most closely convey the meaning of adequate documentation. For the purpose of clarity, a brief description of all four qualities are given below:

Accuracy: The common definition for 'accuracy' is the freedom from mistake or error; a synonym is 'correctness'. Within the context of computer documentation, accuracy can be defined as the consistency among the code and all documentation of the code for all requirements. That is, accurate documentation should reflect the realized state of the system that it represents.

Completeness: The standard definition for
'completeness' is the possession of all necessary parts, elements, or steps. For the purposes of computer documentation, a set of documentation is complete if all of the required information is present.

Usability: The dictionary definition for 'usability' is the capability, convenience, or suitability of being employed. Relative to assessing documentation quality, usability is more appropriately defined as the suitability of the documentation relative to the ease with which one can extract needed information.

Expandability: A general definition for expandability is the ability to increase an object's extent, number, volume, or scope. A synonym is 'extensibility'. The rationale for including expandability as a desirable quality is to reflect the concept of underlying document maintainability, and in particular the ease with which the documentation can be added to and modified. In concert with the notion of document maintainability, a more precise definition for expandability is: the capability of the documentation to be modified in reaction to changes in the system. This quality is assessed through measures reflecting ease of modification.

SUMMARY

Latest standards in software engineering, with particular emphasis on documentation is discussed here. The needs of documentation and its various components are discussed in detail, from the viewpoint of software engineering. System documentation, an integral component of software documentation, conveys the requirements, design philosophy, design details, capabilities, limitations and other characteristics of a system while user documentation, another component of software documentation conveys to the end-user instructions for using the system to obtain desired results. The principle of concurrent documentation emphasizes that the changes made to the system during a particular phase of development should be reflected in all the related documentation components. This brings out the need for a software librarian, which has been discussed in detail in this article. A universal documentation system which is suitable for all types of organizations is not available, but each organization can develop its own standard documentation procedures from the guidelines given by the standards available.

For evaluating the adequacy of a document, a set of quality indicators such as accuracy, completeness, usability and expandability are to be given due importance.

ACKNOWLEDGEMENTS

Shri J M Jose is thankful to the CSIR for supporting this work.

REFERENCES

<table>
<thead>
<tr>
<th>ANSI/IEEE Std</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>729-1983</td>
<td>Glossary of Software Engineering Terminology</td>
</tr>
<tr>
<td>730-1989</td>
<td>Software Quality Assurance Plans</td>
</tr>
<tr>
<td>828-1990</td>
<td>Software Configuration Management Plans</td>
</tr>
<tr>
<td>829-1983</td>
<td>Software Testing Documentation</td>
</tr>
<tr>
<td>830-1984</td>
<td>Guide for Software Requirements Specifications</td>
</tr>
<tr>
<td>982.1-1988</td>
<td>Standard Dictionary of Measures to Produce Reliable Software</td>
</tr>
<tr>
<td>983-1986</td>
<td>Software Quality Assurance Planning</td>
</tr>
<tr>
<td>990-1987</td>
<td>Ada As a Program Design Language</td>
</tr>
<tr>
<td>1002-1987</td>
<td>Taxonomy of Software Engineering Standards</td>
</tr>
<tr>
<td>1008-1987</td>
<td>Software Unit Testing</td>
</tr>
<tr>
<td>1012-1986</td>
<td>Software Verification and Validation Plans</td>
</tr>
<tr>
<td>1016-1987</td>
<td>Software Design Descriptions</td>
</tr>
<tr>
<td>1028-1988</td>
<td>Standards for Software Reviews and Audits</td>
</tr>
<tr>
<td>1042-1987</td>
<td>Guide to Software Configuration Management</td>
</tr>
<tr>
<td>1058.1-1987</td>
<td>Standard for Software Project Management Plans</td>
</tr>
<tr>
<td>1063-1987</td>
<td>Standard for Software User Documentation</td>
</tr>
</tbody>
</table>