An Automated Documentation System for a Large Scale Manufacturing Engineering Research Project

Howard M. Bloom
Leader, Software Systems Group
Carl E. Wenger
Electronics Engineer
National Bureau of Standards
Metrology Building, Room A127
Washington, D.C. 20234

The Automated Manufacturing Research Facility (AMRF) being implemented at the National Bureau of Standards (NBS) involves the development of a software system integrating the various information processing, communications and data storage functions required in a totally automated manufacturing environment. The project contains a five year software development effort by more than thirty research staff organizationally partitioned into many units working concurrently on different parts of the system and supplemented by software acquired through procurement or contractual effort. As the facility is implemented in modular blocks, new software development will be undertaken as research into factory automation technology continues.

In such a research environment, a system is needed for maintaining documentation for the software life cycle for the following purposes: (1) tracking progress of individual module development, (2) allowing for the availability of up-to-date information on module description to other members of the project who need to interface to a given module, (3) developing a cross reference of module and data element relationships, (4) providing a documentation format that includes specific reporting requirements to upper level management, and (5) generating working level documentation that can be easily modified and serve as an up-to-date reference for anyone with interest in the project or special subproject.

This paper describes the structure of a software system that is functioning in a research environment and satisfies the following design constraints: (1) provide minimum additional effort on the part of the system developers, and (2) reflect the structured thinking of the developer during the software life cycle.

*Contribution of the U.S. National Bureau of Standards, not subject to copyright.

ACM
SIGDOC

Second International Conference on Systems Documentation

April 29-30
Seattle, WA

• The key to such a management system is the early documentation of the system modules identified through a decomposition of all functions from the integrated system down to the lowest level subroutines. Each component module of the system has a group of seven documents to track it through the life cycle. The decomposition provides each developer with an understanding of where his module fits into the overall system and permits module documentation to be produced that can be limited to just a few pages. The documenter is encouraged to enter information as it becomes known so that the system always reflects the latest status of the research effort.

The automated system is being developed as an on-line interactive menu-driven system that allows the developer to enter information about his modules. The final system version will be capable of the following functions: (1) full-screen editing, (2) menu driven interface between the user and the system, (3) cross referencing between module and data elements, (4) generation of reports, (5) generation of work schedules, (6) monitoring of system milestones, (7) checking for appropriate information, and (8) managing a data element dictionary.

Background

The National Bureau of Standards was given a Congressional initiative in October 1980 to build an Automated Manufacturing Research Facility (AMRF) where issues involving the batch shop manufacturing environment could be studied (Simpson, 1982). The two major focal points were to be (1) the establishment of guidelines for interface standards between the various system modules in a manufacturing system and (2) the extension of the dimensional traceability ("Deterministic upon the premise that an accurate monitoring of the manufacturing process can replace or reduce the need for inspecting the product.

The research project involves the effort of approximately 70 persons working on various aspects of automation technology which must be integrated together as a defined facility implementation. The researchers come from a variety of disciplines--computer science, electrical engineering, mechanical engineering, industrial engineering, etc. -- and are applying technology to a wide range of manufacturing areas. Examples of systems include hardware intensive sensory interactive robot control systems, network communication, smart machine tools, vision systems, etc. Software intensive systems include data base management systems, network protocols, computer aided process planning, computer-aided-design, manufacturing planning and control systems, scheduling, process monitoring and artificial intelligence, graphics, and simulation. Because of the wide range of applications, many languages will be used such as FORTRAN, PASCAL, FORTH, LISP, C, varieties of assembly languages, and special purpose languages such as APT, SLAM, PRAXIS, etc.; in addition many computers will be used varying in size from microcomputers to minicomputers and special purpose computers such as controllers, data base machines and AI/graphics machines.

The project will support and draw upon both university and industrial efforts that will either develop or furnish software to be included in the research facility. The AMRF will ultimately serve as a testbed research for NBS, industry, and universities to work together on solving the critical manufacturing problem stated above. Hence even though the environment is research in its nature, it was recognized that a strong software management program was essential to the software development effort and that at the heart of such a program would be an effective documentation structure.

Documentation Philosophy

Software documentation is certainly not a new problem and much effort has been spent on defining standard types of documentation (FIPS Pub 38). However the emphasis has invariably been placed on management information systems or large scale weapons systems that will be used in a production environment. Much has been written on the ways in which documentation for production systems should be prepared, controlled and maintained (Neuman, 1982). A major difference between the research and the typical production development effort is in the resources allocated to documentation. A good project manager will include in his budget funds for documentation as well as an allocation for staff time required for documentation of his milestone commitments. The major documentation emphasis in research efforts is in the area of publishing technical papers on the results and usage of the system.

The research environment of the AMRF project may be considered to be the major constraint affecting the software management efforts. In order to function effectively, the following goals must be achieved: (1) additional effort on the part of the system developers that would delay the research results must be minimized, (2) the structured thinking of the developer during the software life cycle should be reflected, (3) useful information about the development of particular module must be communicated to the staff without overwhelming them with paper or meetings, and (4) a level of documentation must be provided that can satisfy a multitude of readers who are interested in various aspects or views of the project.

The second issue above - the software life cycle - is a difficult concept to institute in a research environment. The functional requirements for systems very often do not originate with the end user (or sponsor) but are tentative goals for the software developer and are subject to change as the design is implemented. Requirements are often driven by the design decisions and demonstration milestones.

Since the staff is research oriented, it is often the case that the programmer does not have a lot of experience in generating code for a particular application and must spend time "rapid prototyping" to gain experience with various programming techniques to better understand the problem being undertaken. Even if structured design and coding practices are followed, there is a real concern on the part of the designer that he must spend his time documenting stages in development that could be completely changed.

The key to a useful documentation system is the early identification of the components of the system from the major subsystems down through the lowest level subroutines. An example of such a decomposition for the AMRF is given in Table 1. This decomposition allows each developer to understand where his module fits into the overall system and permits module documentation to be produced that can often be limited to just a few pages. The documenter is encouraged to record information as it becomes known so that the documentation data base always reflects the latest status of the research effort.

The purpose of the documentation varies with the level of the system. At the top level, documentation represents a formal description of the effort to be undertaken, the resources required, the overall design architecture, the acceptance test procedures and the manner in which the system is to be constructed. This documentation should be completed as soon as possible as it represents a "contract" between the developer and the supervisor of the project's goals. The top level documentation should not change significantly during the life cycle. This information can serve as the basis for developing reports that will be made available for people outside the project who have an interest in the effort. The lower levels of the system should also be documented, but not necessarily as quickly or as completely since the design of the actual implementation may vary greatly as insight is gained into the project.

In summary, the top level of documentation in a research project is important for internal communication and for interface requirements specifications between modules developed by different groups. In addition, the documents are used in the review process to aid in the "walk-through".

<u>Documentation Types</u>

The documents described in this section when completed will give a history of the entire project. Each document has a standard format that is meant to be used as a guideline for preparing the necessary information. It is important that all system developers follow the same guideline in order to ensure that the proper information is recorded and that it is in a form that is convenient for retrieval by all project staff.

Each level of decomposition of the system has a complete life cycle of its own, which, is linked into the life-cycle documentation at the next higher level. The design phase at one level drives the functional requirements phase of the next lower level.

The ten documents normally required in a software development project have been reduced to seven:

1. Functional Requirements Document

This document initiates the software development effort, and includes the description of the following information: requirements definition, system description, data requirements, interface specifications and resource requirements. This document also identifies the next level of modules in the decomposition so that the next set of documents can be identified. This document must be prepared as completely as possible for the highest level system before any additional development efforts are undertaken. It serves as the basic agreement between user and the supervisor or project leader and delineates the magnitude of the effort. Information not available at the beginning of the project can be furnished at an agreed upon later date.

2. System Development Plan

This document shows the results of the software planning function. It includes a description of all hardware and software resources, their estimated cost and expected dates of usage. The life-cycle checklist is also produced to show the tentative review dates for each required document or module.

3. Module Design Specification Document

This document specifies in detail how the module is to be implemented. It includes a description of the control logic, interface specifications, internal storage layout, data base requirements and formats for the input/output functions. A description of the messages, both prompts and error description that are generated by the system is also included.

4. Data Base Requirement Document

This document describes the logical layout of the data elements to be managed by the system. The information should include the relationship between the data elements, and the format of each element. The physical structure of the data base is also defined. Procedures for backing-up and restoring the data base are specified. (The data base manager can help in preparing this section.) Any supporting application software that must be interfaced to the data base management system should be specified.

5. Test Plan

The plan for testing the module is described by the following information: (1) tentative dates for carrying out a test, (2) resources required, (3) module capabilities to be tested, and (4) step-by-step procedures for carrying out the tests. The test data should be described in terms of its content and physical location. The expected output results must also be specified.

6. Development Journal

This document is used to list the history of events that have occurred in implementing the system. This includes problems, successes, solutions to problems, test results, etc.

7. System Library Document

This last document of the development stage includes all the information required to run and maintain the software. The operating environment needs to be identified in terms of hardware requirements and support software. A detailed identification of all files containing program information needs to be specified.

<u>Description of ADS</u>

An Automated Documentation System (ADS) is being developed as an interactive menu-driven system to assist the developer in the preparation of system documentation. The objectives of the system are as follows: (1) Provide an automated system for generating documentation for the software development life cycle, (2) Provide tables of system information that are useful in coordinating the relationship among system entities, (3) Provide tracking of the progress of the software project, and (4) Provide each member of the software project with a means of obtaining up-to-date information on the description of those systems being implemented by other members of the project.

The Automated Documentation System will be used to handle the automatic recording of working-level information that a system developer generates during the life cycle software development. The user will enter information as it becomes known to him into any of the seven document-type files mentioned earlier. The ADS must be capable of tracking the entities as they are entered, determining if they are already available as part of the system data base, and also note for the new entities that future documents are required. For example if the system has 3 modules, a document needs to be generated for each module.

The following list is the functional requirements to be considered for the first implementation of ADS:

(1) Documents will be stored in Standard ADS Document Format (SADF), allowing changes and additions to the document. (SADF is defined as the RUNOFF output print file format which is left and right justified.) The documents may be created from a terminal keyboard with prompting and full screen formatting, from existing files in SADF, or from existing files or parts of files in other formats.

(2) Inputing/editing of ADS test data will be done using full screen capabilities including forms, protected fields, and forms paging. ADS should allow documents to be completed on other systems such as word processing or text editors and then to be logged into ADS after processing by the Formatter and Parser modules. Documents should be logged out for editing by ADS or other text editors and made unavailable to other users except in a read only mode. When the author chooses, the document is formated, parsed, and logged back into the system.

(3) All field values for any document should be available for retrieval purposes. When appropriate, field values should be checked for validity and logical consistency.

(4) There should be a system command structure which allows mnemonics or menu selection of system or subsystem, document type, or ADS functions such as edit, input, print, etc. Variable HELP levels controlled by the user or by a user profile should lead the user through the use of the system.

(5) The system should maintain an index of all system and subsystem documentation files with the status of each. All required subsystem documentation files should be initiated automatically when the subsystem is first mentioned in a higher level Functional Requirements Document (FRD). All data field values already known to ADS should be automatically filled in for these documents.

(6) Provide for the generation of reports using any combination of fields

from any document.

(7) The system should allow multiple users to use the system at the same time, but not allow more than one user to update the same document at the same time.

The first version of the system has been implemented on the AMRF research computer. A description of the use of this version is given below and a sample terminal session using ADS on the VAX is shown in figures 1, 2, and 3. For the sample session, a '===>' is displayed to prompt the user for an ADS command or for data required before processing of the selected document begins. After processing of the document begins, a '->' prompts the user for text data to be input for the document. Underlined text data in the sample session are responses to these prompts and are terminated by carriage returns. Responses to some prompts without underlined text are carriage returns only.

The first part of one continued ADS terminal session on the research computer is shown in figure 1. The user executes the Automated Documentation System (ADS) by entering the command 'ADS'. An ADS sign on message followed by the main command menu is then displayed.

The 'XPR' command as shown in the ADS command menu toggles the expert mode switch to provide abbreviated displays. When an ADS session is completed by entering the 'EX' (or 'EXIT') command, the last setting of this switch is stored in the ADS user profile which is automatically created and maintained for each user. If ADS is exited with this switch set to the expert or abbreviated mode, the expert mode will be set for the next ADS session. If an illegal or blank command is entered, all command choices are displayed even though the expert mode is set. The expert mode was selected in the sample session, figure 1, by entering 'XPR' in response to the command prompt.

The 'I' or input menu option is used to input new data to an existing document or to create a new document. If a carriage return only is entered for the system name and document type prompts, a list of the current system names and document types is displayed. After the system name and document type is selected, a prompt is displayed for each data field of the selected document. Single or multiple lines of text may be entered for each prompt. The input option was selected by entering 'I' in response to the command prompt in figure 1. After the input option was selected, ADS prompted for the System Name Code for which a document is to be processed. The user entered a carriage return only to get a list of systems for which document processing had been previously started. ADS was selected by entering 'ADS' as a prompt response.

The ADS terminal session is continued in figure 2. A carriage return only response to the Document type code prompt listed all the document types and their codes. The Functional Requirements Document was selected by entering 'FRD'. The actual text of the document begins after the dashed line in figure 2. Each '->' prompts the user for data to be included in the document. The underlined responses to the '->' prompts are terminated by carriage returns.

Text data may be entered directly for each '->' prompt. A carriage return or four other ADS subcommands may also be entered instead of text data in response to a '->' prompt. These four subcommands are 'EDIT', 'FIND', 'EXIT', and 'HELP'.

The 'EDIT' subcommand invokes the EDT text editor which may be used to input new data into the ADS document. When prompted for the edit file name, pressing RETURN without entering a file name will cause ADS to create a temporary edit file to be used as input data. When EDT is exited, the contents of edit file is inserted into the ADS document, and the '->' prompt for the next document section is displayed.

The 'EDIT' subcommand may also be used to insert an existing file into the ADS document by entering an existing file name when prompted. After editing of the file is completed, an 'EXIT' command to EDT will insert the just edited file into the ADS document and prompt for the next document section. This ADS feature is illustrated starting near the end of Figure 2 and continued in figure 3. A response of 'EDIT' to the '->' prompt and the file name response to the '===>' prompt invoked the EDT editor so that file [WENGER.ADS]DESCRPT.DAT could be edited using EDT and then inserted into the ADS document. The EDT command 'TYPE WHOLE', figure 3, listed the file, and the first 'EXIT' exited EDT so that the file was inserted into the document. The second 'EXIT' exited the ADS edit mode so that a new ADS command could be entered.

The 'FIND' subcommand prompts for a document section number. The section number to be found may be any existing section number either ahead or behind the current position. After the requested section is found, data may be typed in for that section or EDT may be invoked to insert a file in that section.

The 'EXIT' subcommand causes an exit from the ADS input mode to the main command menu, and the 'HELP' subcommand prints help information about how to answer the '->' prompt while in the input mode.

The 'E' command in the main ADS command menu allows use of the EDT editor to edit selected sections of a document. When the 'E' option is selected, a prompt is displayed for the first document section to be edited. Next a prompt is displayed for the first NOT included section. All document sections starting with the section number entered for the first prompt up to but not including the section number entered for the last prompt will be brought into the EDT editor. After an exit from EDT, the just edited sections will be replaced in the document. Use of the 'E' option is shown in figure 3. The user entered 'E' for the first '===>' prompt and a carriage return only for the second to edit a section of the document currently being processed. Entering '2.' for the first included section and '3.' for the first not included section entered the document section 2 into the EDT editor for editing. Entering 'EXIT' for the EDT '*' prompt saved section 2 back into the document.

The three remaining commands in the ADS command are 'L', 'P', and 'EX'. The 'L' command lists a document on the terminal, the 'P' command prints a document on the line printer, and the 'EX' command ends the ADS session. The command 'EXIT' may also be used instead of 'EX'.

Conclusions

The Automated Documentation System has been implemented and has become part of the AMRF software management program. It is still too early to tell what level of success will be achieved. The system has been heavily used and the documentation output has been very convenient for formating project descriptions. There are still many features of the system such as the cross reference capability and report generation that need to be implemented.

Acknowledgement

The authors want to thank Albrecht Neumann, of NBS, for his guidance and assistance in the area of software documentation. A special thanks to Charles McLean, of NBS, for his technical review of this paper.

References

1. Neumann, Albrecht J., Management Guide for Software Documentation, NBS Special Publication 500-87, January 1982.

2. A Guide to Software Requirement Specifications (IEEE Project 830), Preliminary Draft, June 15, 1982.

3. Guidelines for Documentations of Computer Programs and Automated Data Systems, FIPS Pub 38, National Bureau of Standards, February 15, 1976.

4. Hecht, Herbert, Requirements Documentation - A Management - Oriented Approach, FIPS Software Documentation Workshop, National Bureau of Standards, March 3, 1982.

5. Simpson, J. A. National Bureau of Standards Automation Research Program, Proc. Fourth IFAC/IFIP Symposium on Information Control Problems in Manufacturing Technology, October 1982.

Table 1

System Architecture for the AMRF

- Manufacturing Information Processing System

Support System A:	System Modeling Package
Support System B:	Software Development System
Support System C:	Graphics Support System
Support System D.	Expert Systems/Artificial Intelligence

1. Production Control System

    Facility Control System
    Shop Control System
    Cell Control System

        Group Technology Cell
        Material Support Cell
        Workstation Pool Cell

    Workstation Control System
    Vertical Machining Station
    Horizontal Machining Station
    Turning Station
    Cleaning fand Deburring Station
    Material Handling System
    Automatic Inspection station

    Equipment Control System

        Integrated Robot system
        Automatically Guided Vehicle
        Machine Vision System
        Carousel Storage Buffer
        Machine Tool Controllers

2. Network Communications System

    Network Interface Processor
    Network Management System

3. Distributed Data Administration System

    Data Base Management Systems

        Facility DBMS
        Shop DBMS
        Cell DBMS
        Workstation DBMS
        Equipment DBMS

    Data Transform Processor

    Integrated Data Dictionary System

7.1

```
 $ ADS

Welcome to the AMRP Automated Documentation System (ADS)

Version 1.01, last update: 08/04/82.

$ ADS

Welcome to the AMRP Automated Documentation System (ADS)

Version 1.01, last update: 08/04/82.


          ADS Command Menu

      I - Input data for a system document

      E - Edit an existing document using EDT

      L - List a document on terminal

      P - Print a document on line printer

     EX - Exit ADS

    XPR - Toggle expert mode

          Command ===> XPR

          Command ===> I

Please enter the desired system name code, or
press RETURN for a list of system names  and codes.

System Name Code ===>


Code     System Name

ADS      Automated Documentation System
CDI      CAD Directed Inspection
IMS      Integrated Manufacturing System
RCS      Robot Control System

System Name Code ===> ADS
```

Figure 1. Sample ADS Session.

```
Document type code ===>

Code    Document Type

FRD     Functional Requirements Document
SDP     System Development Plan
SDC     System Development Checklist
DBR     Data Base Requirement Document
MDS     Module Design Specification Document
STP     System Test Plan
SDJ     System Development Journal
SLD     System Library Document

Document type code ===> FRD


Document file does not exist. Press RETURN to create a new file
or enter any character. ===>

Enter the document section number you wish to start processing,
or press RETURN to start at beginning of document. ===>

--------------------------------------------------------------------

                Functional Requirements Document

1. General Information

1.1 System Name:  ->Automated Documentation System

1.2 System Identification:  ->ADS

1.3 System Version:  ->1.01

1.4 Organization:  ->Software Systems Group

2. Systems Overview

2.1  Objective:  ->

(1)Develop an automated system for generating documentation
for the software development cycle.

2.2 Description:  ->EDIT

EDIT file name ===> [WENGER.ADS]DESCRPT.DAT

    1    The Automated Documentation system will be used to handle the
```

Figure 2. Sample ADS Session, contiued.

```
*TYPE WHOLE

     1     The Automated Documentation system will be used to handle the
     2     automatic recording of working-level information that a system
     3     developer generates during the life cycle software development.
[EOB]

*EXIT

_DRA1:[WENGER.ADS]DESCRPT.DAT;2 3 lines

2.3  Design Goals:  ->EXIT

       Command ===> E

Press RETURN to process a FRD document for the ADS
system, or enter any character. ===>


File exists. Copying to TEMP file...

First included section. ===> 2.

First NOT included section. ===> 3.

     1     2. Systems Overview
*TYPE WHOLE
     1     2. Systems Overview
     2
     3     2.1  Objective:
     4
     5     (1) Develop an automated system for generating documentation
     6     for the software development cycle.
     7
     8     2.2 Description:
     9     The Automated Documentation system will be used to handle the
    10     automatic recording of working-level information that a system
    11     developer generates during the life cycle software development.
    12
    13     2.3  Design Goals:  ->
    14
    15     2.4  Techniques:  ->
    16
    17     2.5  Constraints:  ->
    18
[EOB]
*EXIT

       Command ===> EXIT
FORTRAN STOP
$
```

Figure 3. Sample ADS Session, continued.