The DSN Programming System

W. D. Hodgson
TDA Engineering Section

This article describes the DSN Programming System. The Programming System provides the DSN with implementation tools and practices for producing DSN software. This article provides a general description of the System, as well as its key characteristics, status, plans, and expected benefits.

I. Introduction

The DSN Programming System is a unified body of practices and implementation tools for producing DSN software on time, within budget, conforming correctly to functional requirements, and of predictable and low life-cycle cost. The system incorporates software methodology, software standard practices, language and data base standards, including standard languages, implementation aids, and management aids.

The DSN Programming System is needed because:

(1) DSN service continues to require more software

  (a) End user requirements continue to increase, both in quantity and quality.

  (b) Network functions need to be computerized to stabilize maintenance and operating costs.

(2) DSN software costs are continuing to be an increasing fraction of the DSN budget. Transferred software in the DSN is increasing at 15% per year.

(3) The DSN customer, the end user project, requires that DSN software be delivered on time and with full negotiated capability. The Programming System provides assurances, as well as the means for the DSN to meet the end user requirements.

II. Key Characteristics

The DSN Programming System includes the production and maintenance of DSN Software Standard Practice documents, the implementation of DSN standard programming languages according to pre-defined functional requirements, and the implementation and evaluation of the initial concept of the distributed DSN data base (in the Configuration Control and Audit Assembly), as well as software implementation aids and management aids.

There are six complete DSN Standard Practice documents[1] for software that cover implementation including documentation; Software Implementation Guidelines and Practices, Preparation of Software Requirements Documents (SRD), Preparation of Software Definition Documents (SDD), Preparation of Software Specification Documents (SSD), Preparation of Software Operators Manuals (SOM), and Preparation of Software Test and Transfer Documents (STT).

[1]These documents are currently available on request from the author.

A seventh Standard Practice is currently in development that will cover the DSN data base, including implementation guidelines and practices.

The Software Implementation Guidelines and Practices describe the overall Tracking and Data Acquisition (TDA) software methodology, policy, and software management plan. The other Standard Practices provide supporting detail as summarized below.

The Standard Practices on Software Requirements Documents and Software Definition Documents cover the early activities of requirements identification and software architectural design. These are the conceptual phases where decisions made during these formative stages tend to have profound effects on the overall costs, commitments, and the general approach. The SRD and SDD de-emphasize formality and permanency. The SDD describes the software architectural design, which permits a ±10% accurate estimate to be made of the cost and schedule of the remaining implementation, which is the program construction. The architectural design can, in fact, be the basis for a bid by an outside contractor for a fixed price implementation. Timing is critical and early reviews facilitate redirection, if needed. Also, informal documentation during these phases facilitates changes as may be needed prior to final approval. The documents covering the preparation of SSDs, SOMs, and STTs require that they be produced concurrently with the program construction and testing activities. They document the true "as-built" and "as-tested" computer program and are used to operate and maintain the transferred (delivered) program. These documents tend to be more formal and are updated, as needed throughout the useful life of the software. The standard practice on data bases is to provide guidelines for standardizing and implementing the operational DSN Data Base. The DSN Data Base includes the data throughout the DSN in continual use for ongoing business, operations, and commitments.

The standard programming languages reduce implementation and sustaining costs because they are both standard and high-level languages. They permit implementers to concentrate on TDA design problems, not on machine details. Furthermore, having standard languages preserves implementer and manager skills and furnishes immunity of DSN applications programs to hardware and operating system changes. The machine-independent design principles being applied reduce the costs of providing the standard languages on various computers the DSN uses. Two standard languages have been identified: a non-real-time language for Operations management use, augmented by a File Editor for data base modification and update, and a real-time language.

The non-real-time language machine-independent design is being completed and implemented on the PDP-10, on the Univac 1108, and on the DSN standard minicomputer (Ref. 1). The File Editor functional requirements are being defined in preparation for the machine-independent design and subsequent implementation in the Configuration Control and Audit Assembly (Ref. 2). In later years, the real-time DSN standard language will be similarly implemented, initially for the DSN standard minicomputer, and then for DSN Control and Computation Modules (CCMs), the DSN standard microprocessor components.

The implementation aids are to be selected and then implemented, based on the methodology research coordinated under the DSN Advanced Systems program and on the experiences of the Mark III DSN Data Subsystem Implementation Project (MDS). These may include automated finished graphics, computer-based implementation document construction, a program design medium, and perhaps others. The Software Standard Practices will be revised to incorporate the proper use of these aids as they are transferred to Operations, acting in unison with the methodology and languages.

The DSN Programming System is organized in the following structure. (Figure 1 also gives a graphic representation of the same information for clarity.)

(1) Methodology research

(2) DSN non-real time language

  (a) MBASIC[2] machine independent design (MID)

  (b) MBASIC[2] MID extensions (MIDX)

  (c) MBASIC[2] batch compiler

  (d) MBASIC[2] reference manual

  (e) MBASIC[2] implementations

  (f) File editor

(3) DSN real-time standard language

  (a) Design

  (b) Implementation

(4) Data base implementation and standards

(5) Implementation aids and management aids

(6) Software standard practices

  (a) Guidelines and practices

  (b) Software Requirements Document preparation

  (c) Software Definition Document preparation

  (d) Software Specification Document preparation

  (e) Operators Manual preparation

[2]A trademark of the California Institute of Technology.

(f) Test and Transfer Document preparation

(g) DSN data base implementation guidelines and practices

III. Schedule Overview

The DSN Programming System schedule of major deliverable milestones is described in the following paragraphs.

The first six software standard practices have been completed since early FY77. The seventh (Data Base Implementation Guidelines and Practices) will be complete in early FY78.

The Machine Independent Design (MID) of the non-real-time standard language is also complete with one implementation (on the DEC System-10 computer) transferred to operations and in use.

The first volume of the software methodology textbook (Ref. 3) has been published and is available from (among other sources) Prentice-Hall. The second volume is drafted and will be published in early FY78.

It is planned that the non-real-time standard language (MBASIC) interpreter and batch compiler is to be complete and operational on three different computers (MODCOMP II, Univac 1108, and the DEC System-10) in late FY80, followed shortly thereafter by a standard file editor implementation on the same three machines.

The standard real-time programming language will be complete in FY80 for the DSN standard minicomputer, followed one year later by implementation for the DSN standard microprocessor components (CCMs).

An overview of the Programming System schedule is shown in Fig. 2.

IV. Benefits

Many benefits are expected or are currently being realized because of the application of the Programming System. Table 1 shows in what way each element of the Programming System provides benefits. Some of the benefits can only be predicted based on current software literature (since the Programming System tool is not yet available) but many have actually been measured. For example, for the implementation of the MBASIC language (itself a test bed for the Programming System), costs can be directly compared to the costs of a previous implementation which did not use the Programming System tools. The comparison in this case showed that a machine-independent design was obtained at no additional cost over the old way of machine-dependent design.

V. Summary

All the elements of the DSN Programming System have been defined. Those elements that exist now are being used in Implementation and Operations. The effects on software life-cycle costs and productivity are beginning to be understood. The transfer of all the elements of the Programming System to operations is planned to be complete in 1982.

References

1. Tausworthe, R. C., "Software Production Methodology Test Bed Project," *Deep Space Network Progress Report 42-33*, pp. 186–191, Jet Propulsion Laboratory, Pasadena, Calif., June 15, 1976.

2. Bryan, A. I., "A Distributed Data Base Management Capability for the Deep Space Network," *Deep Space Progress Report 42-33*, pp. 32–36, Jet Propulsion Laboratory, Pasadena, Calif., June 15, 1976.

3. Tausworthe, R. C., "Standardized Development of Computer Software, Vol. I – Methods; Vol. II – Standards," Jet Propulsion Laboratory, Pasadena, Calif., July 1976.

Table 1. Programming System benefits

Benefits	Elements						
	Standard practices and methodology	Standard NRT interpreter and compiler	Standard RT interpreter and compiler	Standard file editor	Implementation aids	Data base standards	Machine-independent design
Software reliability	b	a	b		b	a	b
Accurate scheduling	b	a	b		b		
Accurate costing	b	a	b		b		
Reduced implementation costs (program development)	a	a	b	b	b		a
Reduced operations costs (OPS people time)	a	b	b	b		b	
Reduced sustaining costs (show 1108 vs PDP, keeping within specs)	b		b		b	a	a
ECO engineering (improvements to software)	a	b			a	a	a
Lower computer production costs (CPU)		c	c				
Reduced training/retraining costs	a	a	b	b	b	b	b
No new file editors				b			
Better management and implementor communication	b	b	b	a	a	a	a
Reduced test time	b	a	b	a	a	a	a
Application software insulated from O.S. changes		a	b				
Same engineer for multiple tasks	b	a	b		a	a	a
Portable software	b	a	b	b	a	b	b

[a]Predicted information only

[b]Some empirical evidence exists, from both DSN and external data sources

[c]Benefits only if interpreter used for development activities and compiler for production

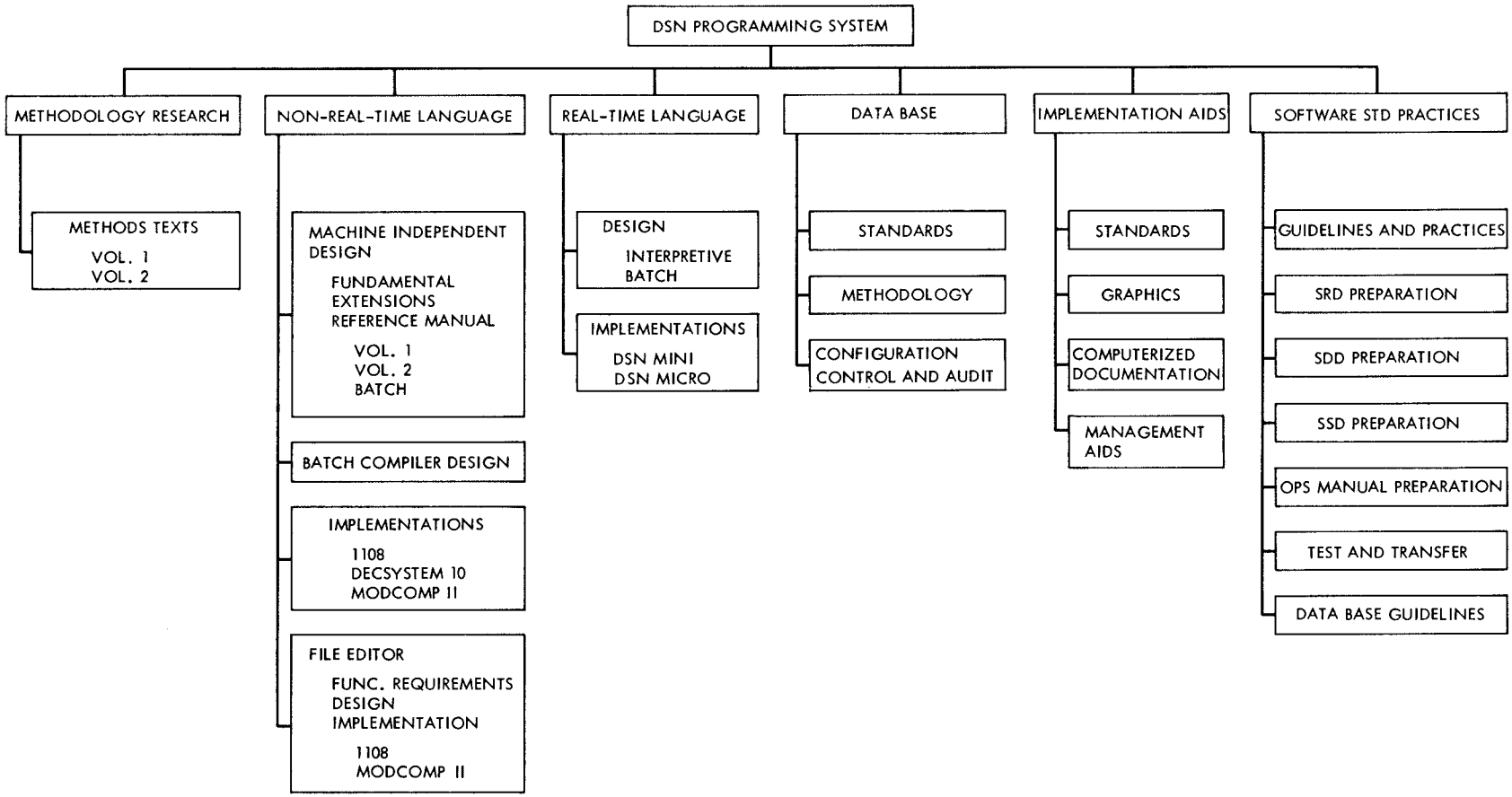
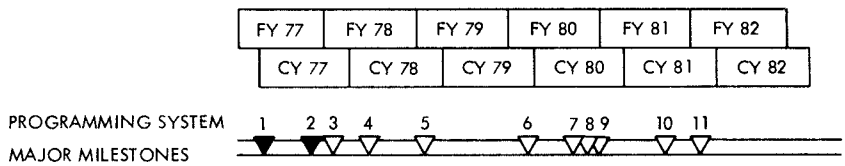


Fig. 1. DSN Programming System



1 COMPLETION OF SIX SOFTWARE STANDARD PRACTICES

2 COMPLETE MACHINE INDEPENDENT DESIGN OF NON-REAL-TIME STANDARD LANGUAGE (MBASIC[a])

3 COMPLETION OF SOFTWARE METHODOLOGY TEXTBOOK, VOL. II (VOL. I COMPLETED IN FY 77)

4 COMPLETION OF DATA BASE STANDARD PRACTICE

5 REAL-TIME LANGUAGE SPECIFICATION

6 FIRST BATCH COMPILER OPERATIONAL

7 FULL IMPLEMENTATION OF THE MBASIC LANGUAGE (BATCH AND INTERPRETER) ON THREE DSN COMPUTERS

8 FULL IMPLEMENTATION OF A STANDARD FILE EDITOR ON TWO DSN COMPUTERS

9 IMPLEMENTATION OF REAL TIME STANDARD LANGUAGE ON DSN STANDARD MINICOMPUTER

10 IMPLEMENTATION OF REAL TIME STANDARD LANGUAGE ON DSN STANDARD CCMs

11 COMPLETION OF PROGRAMMING SYSTEM IMPLEMENTATION[b]

[a] TRADEMARK OF THE CALIFORNIA INSTITUTE OF TECHNOLOGY

[b] SUBSEQUENT IMPLEMENTATIONS CONDUCTED AS PART OF SPECIFIC COMPUTER MEASUREMENTS

Fig. 2. DSN Programming System overview