BOEING

CAGE CODE 81205

THIS DOCUMENT IS:

CONTROLLED BY Functional Test Engineering, T-7640

ALL REVISIONS TO THIS DOCUMENT SHALL BE APPROVED BY THE ABOVE ORGANIZATION PRIOR TO RELEASE

PREPARED UNDER ☐ CONTRACT NO.
☐ IR&D
☒ OTHER

PREPARED ON

FILED UNDER

DOCUMENT NO. D532W003

MODEL 777

TITLE SETL Dataset Development Software Tools

☒ THE INFORMATION CONTAINED HEREIN IS NOT PROPRIETARY.

☐ THE INFORMATION CONTAINED HEREIN IS **PROPRIETARY** TO THE BOEING COMPANY AND SHALL NOT BE REPRODUCED OR DISCLOSED IN WHOLE OR IN PART OR USED FOR ANY DESIGN OR MANUFACTURE EXCEPT WHEN SUCH USER POSSESSES DIRECT, WRITTEN AUTHORIZATION FROM **THE BOEING COMPANY.**

ORIGINAL RELEASE DATE

ISSUE NO. TO DATE

ADDITIONAL LIMITATIONS IMPOSED ON THIS DOCUMENT WILL BE FOUND ON A SEPARATE LIMITATIONS PAGE.

Additional contributors are listed on the next page.

	SIGNATURE	ORGN	DATE
PREPARED BY	Sunil Velaga	T-7644	08 May 91
CHECKED BY	Shirley MacGoddin	T-7644	08 May 91
	(for mail) Robert K. Chapman	T-7644	08 May 91
SUPERVISED BY		T-7640	05-09-91
APPROVED BY	H. J. Lee	T-2640	5-9-91

NEW

ACKNOWLEDGMENTS

Those Involved in the Preparation of this Document:

James J. Boisseranc, A-2057
Gary W. Campbell, T-7644
John N. Esau, B-E32T
Edward L. Marvin, B-E32T
Robert C. Reynolds, A-2057
David A. Rodgers, B-E32T
Jay C. Spery, T-7644
David R. Swords, T-7645
Lowell C. Wickman, T-7645
James B. Willow, T-7645
Yiu-Tim Wong, T-7644

ABSTRACT

Specifies a plan to develop the model 777 dataset development software tools. Defines deliverable items, how they are to be developed and verified, and required documentation.

KEY WORDS

AECMA

automated test equipment

dataset

Integrated Functional Test System (IFTS)

Simplified English Test Language (SETL)

system functional test

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

DEFINITIONS

INTEGRATED FUNCTIONAL TEST SYSTEM (IFTS): An Everett business system that controls all aspects of system functional tests.

SIMPLIFIED ENGLISH (SE): An aircraft maintenance language, as defined by the Association Europeenne des Constructeurs de Materiel Aerospatial (AECMA), AECMA Document PSC-85-16598.

SIMPLIFIED ENGLISH TEST LANGUAGE (SETL): A subset of "Simplified English," as defined by the Association Europeenne des Constructeurs de Materiel Aerospatial (AECMA), AECMA Document PSC-85-16598.

SYSTEM FUNCTIONAL TEST DATASET: A computer file for each airplane configuration providing the procedural test steps, tools, safety, and precautionary notes required to functionally test a system, and determine pass-fail conditions.

ACRONYMS, ABBREVIATIONS, AND SYMBOLS

AECMA	Association Europeenne des Constructeurs de Materiel Aerospatial
ASCII	American Standard Code for Information Interchange
ATA	Airline Transportation Association
ATE	automated test equipment
DBT	design build team
FTE	Functional Test Engineering
FTP	File Transfer Protocol
IFTS	Integrated Functional Test System
OID	Operator Interface Device
SETL	Simplified English Test Language
SFTD	system functional test dataset

1. INTRODUCTION

1.1 Scope

The software tools described herein will support the development of the model 777 airplane system functional test datasets (SFTD) written in the Simplified English Test Language (SETL) for automated test equipment (ATE). These tools will assist Engineering in the generation, modification, verification, communication, and release of these datasets.

1.2 Purpose

The purpose of this document is to:

a. Provide baseline documentation for a number of software tools to support the generation, checkout, release, and maintenance of ATE datasets.

b. Scope the development effort for each of the tools identified above and establish priority.

c. Present a software development plan and identify the resources required.

d. Provide a manning plan and a list of items of concern.

1.3 Background

Currently, the following software tools are used to develop and maintain 747-400 datasets (1) in English (not SETL) for manual tests, and (2) in the "C" programming language for ATE tests:

a. Utility programs--includes syntax and format checker for the manual tests; step re-numbering utility; dataset release utility; configuration extraction utility; Microsoft Word dataset format macros.

b. Editor--Microsoft Word (or any editor which produces ASCII text).

c. Emulator--An Emerald Operator Interface Device (OID) connected to a Hewlett-Packard desktop controller with a special version of the ATE

software environment which enables Engineering to test the dataset running on ATE.

d. Network communications--Contact, a commercial program which provides VT100 terminal emulation for TELNET and file transfer protocol (FTP) sessions between personal computers and IFTS or a Hewlett-Packard computer.

1.4 Intended Users

The software tools described in this project will be used for developing the system functional test datasets in the New Airplane Division (model 777), Boeing Commercial Airplane Group, by:

a. Design Engineering.

b. Functional Test Engineering.

2. REQUIREMENTS DEFINITION

2.1 Assumptions and Groundrules

a. SETL is a subset of Simplified English, as defined by the Association Europeenne des Constructeurs de Materiel Aerospatial (AECMA) (Reference 1), with further syntax and format requirements as defined in *777 ATE Test Procedures Language Requirements* (Reference 2).

b. The business systems and procedures to be used will be those in place in Everett.

c. Computers in all phases must have network communication capability.

d. An IBM-compatible personal computer will be required initially for preparation of datasets and communication with IFTS.

e. The target Engineering platform will be a selected standard UNIX-POSIX workstation running "X Window" (Reference 3).

f. SETL datasets will be developed and released by Airline Transport Association (ATA) Design Build Teams (DBT) or working groups.

g. SETL architecture will be consistent for all ATA chapters.

2.2 General Requirements

a. Utility software--These include a syntax and format checker; a step re-numbering utility; an dataset release utility; a configuration addition, deletion, and extraction utility; and Microsoft Word dataset format macros.

b. Dataset editing software.

c. ATE emulator software.

d. Network communications software.

2.3 Implementation Requirements

2.3.1 Rationale for Phased Implementation

The implementation of the SETL dataset software tools is broken down into four phases. The tools will be

developed in "building-block" fashion, building toward the requirement to be able to edit and emulate the dataset simultaneously on a local workstation.

In the current 747-400 environment, when an error in the dataset is discovered during emulation, it takes twenty minutes, or more, to modify the dataset and restart the emulation process from the beginning. The process includes (1) exiting emulation, (2) entering the editor, (3) modifying the dataset, (4) re-compiling, (5) re-linking, (6) running, and (7) proceeding to the step where the error was discovered.

At the conclusion of this development effort, Engineering will be able to modify the dataset in one window and emulate the dataset in a companion, adjacent window.

Net productivity gains will result from the completion of each phase. The project could be frozen after any phase.

2.3.2 Description of Phases

Table 1, on page 16, provides a cross reference between the new software tools to be developed and the four phases.

2.3.2.1 Phase I

This phase requires some modification and addition to the software used in the current 747-400 dataset development environment in order to accommodate SETL.

With relatively moderate levels of effort and low risk, this approach yields an attractive solution to the immediate requirements for developing SETL datasets.

All datasets developed during this phase will be upward compatible with the software tools provided during the later phases. None of the datasets developed during Phase I will have to be revised during later phases.

Existing software to be utilized in Phase I needs documentation for the purposes of the model 777 program. In particular:

a. *Integrated Functional Test Data Management System (IFTDMS) Engineering Operator's Manual*, D935U002, written for Contact software used in the 747-400 program (Reference 4).

b. The existing emulator.

2.3.2.2 Phase II

This phase enables Engineering to communicate with IFTS, develop the SETL dataset, and emulate, all from a personal computer. This phase also introduces the X Window environment, which will eliminate the need for both the hard-wired, external OID and the local Hewlett-Packard computer.

Some modification to the emulator to accommodate X Window will be required to drive OID displays to the console screen.

Since there is very little implementation of X Window in the Boeing network environment, the risks associated with this phase are inherently greater than Phase I, but only moderately so.

2.3.2.3 Phase III

This phase uses database tools to provide a customized SETL dataset development environment, including step attribution, database revision utilities, database IFTS transfer, and dynamic syntax-format checking.

The ability to communicate between the workstation and IFTS will require some modification to IFTS.

There is a net productivity gain from increased workstation response. The introduction of the database is moderately complex. Therefore, the changes are limited. The database environment is required for Phase IV.

The Phase I utility programs will be ported from the personal computer environment to the workstation.

2.3.2.4 Phase IV

This phase provides faster development and emulation response by moving the emulator software from the remote Hewlett-Packard computer to the local workstation.

The remote Hewlett-Packard computer is no longer required.

This phase also provides dynamic, or "hot," links between the SETL dataset editor and the emulator

software. This provides faster development and debug cycle time.

2.3.3 Description of Required SETL Dataset Development Tools

2.3.3.1 Flat File Dataset Syntax and Format Checker (S1)

The syntax checker allows the engineer to perform syntax checks on the dataset prior to release. It is run on demand and at release as a batch job. This does a dataset format check, an AECMA check, and an IFTS compatibility check.

2.3.3.2 Flat File Dataset Revision Utilities (S2)

These tools provide the capability to generate a new configuration from an existing configuration in flat file environment. They also give the capability to remove configurations from test datasets in flat file environment. Automatic renumbering of steps also will be supported.

2.3.3.3 Special Documentation (S3)

User manuals will be prepared for the existing--

a. File transfer software (Contact).

b. Model 747-400 emulator software.

2.3.3.4 X Window Emulation (Emulator Resident on Remote Hewlett-Packard Computer) and Simulated OID (S4)

The emulator software will reside on, and run in, a remote Hewlett-Packard computer. X Window runs from an engineer's personal computer, and emulates the cart hardware with the operator providing the aircraft responses.

2.3.3.5 Custom Editor using Standard Database tools (S5)

Database editor tool allows the engineer to create or revise the dataset without entering the margin codes or step numbers.

2.3.3.6 Dynamic Syntax and Format Checker Integrated with Editor (S6)

This tool performs the same functions as S1 (see section 2.3.3.1), but in the database environment.

2.3.3.7 Dataset Revision Utilities Integrated with Editor (S7).

This tool performs the same functions as S2 (see section 2.3.3.2), but in the database environment.

2.3.3.8 Custom Database IFTS Transfer (S8)

Database transfer software is used to perform data transfer between the IFTS and Engineering workstation-personal computer environments.

2.3.3.9 X Window Emulation; Emulator Resident on Local Workstation (S9)

This emulator resides on an engineer's workstation-personal computer, and emulates the cart hardware with the operator providing aircraft responses.

2.3.3.10 Dynamic Links between the Editor and Emulator (S10)

Dynamic links between editor and emulator promotes changes made to the SETL dataset in the editing environment to the emulation environment, as well.

TABLE 1 SOFTWARE TOOLS TO BE DEVELOPED

TOOL NUMBER	PHASE I	PHASE II	PHASE III	PHASE IV
S1	D	U		
S2	D	U		
S3	E	D		
S4		D		
S5			D	U
S6			D	U
S7			D	U
S8			D	U
S9				D
S10				D

D--Tool designed in this phase.

E--Existing model 747-400 tool

U--Use existing tool designed in previous phase.

DESCRIPTION

S1: Flat File Dataset Syntax and Format Checker.
S2: Flat File Dataset Revision Utilities.
S3: Special Documentation.
S4: X Window Emulation; Emulator Resident on Remote H-P Computer.
S5: Custom Editor Using Standard Database Tools.
S6: Dynamic Syntax and Format Checker Integrated with Editor.
S7: Dataset Revision Utilities Integrated with Editor.
S8: Custom Database IFTS Transfer.
S9: X Window Emulation; Emulator Resident on Local Workstation.
S10: Dynamic Links Between Editor and Emulator.

3. DEVELOPMENT PLAN

This development plan will define tasks, deliverables, and management practices required to design, develop and produce SETL development software tools. The tools to be provided are identified in section 2.3.3, beginning on page 14.

The development plan provides the foundation on which all the activities are controlled to achieve the desired product.

Related documentation deliverables may be combined under a single document number if it is determined to add clarity and ease to the writing and/or utilization of the affected documentation.

3.1 Management Plans

Project software development will be managed in accordance with these plans:

a. "Software Development Plan," found herein.

b. *Software Configuration Management Plan*, SCMP--This plan will address configuration control processes needed in the development environments. Hardware, software, and processes shall be addressed. The configuration control of both the developed software and the development environment shall be included. Controlling software builds, etc., will be covered.

c. *Software Quality Assurance Plan*, SQAP--This plan will describe the methods by which the Software Quality Assurance function will audit the process to ensure proper implementation of the software requirements.

d. *Software Verification and Validation Plan*, SVP--This plan will layout overall requirements, including test facilities, test environments, and processes and test coverage goals used in finding errors in the developed software programs.

These plans will be based on *BCAG Avionics Computer Software Technical Standard* (Reference 5) modified to the scope of this project. The plans shall be common to the development of all pieces of software listed in section 2. The plans will be released as controlled documents.

3.2 Software Development Process

The process proposed for this effort is based upon experience of BCAG in acquisition of supplier software for embedded airborne software. However, the process has been tailored to the needs of software development of tools used in an engineering environment. The objective is to produce software of sufficient quality in the production of datasets delivered to the factory.

It is not anticipated that the there will be a requirement for regulatory agency certification of any software developed under this plan. However, the processes of this plan are based upon similar processes used in the development and acquisition of software for which certification is required (Reference 5). Should there be a reqirement in the future to provide visibility to a regulatory agency, the agency will be familiar with the processes and methods used in this plan.

A more detailed treatment of the software development plan begins on page 71.

3.3 Software Development Environment Plan

A seperate document, called the *Software Development Environment Plan*, will be produced. It will identify the hardware, software, methods, and processes needed to establish the environment in which the SETL dataset development software tools will be developed and, ultimately, used by Design Engineering.

3.4 System Integration Laboratory (SIL) Integration Support

During the SIL phase of the model 777 system functional testing, it is anticipated that problems may arise that are traceable to deficiencies in one or more of the developed software pieces. The development environments shall be maintained to support the correction of the deficiencies on a timely basis.

3.5 Work Breakdown Structure

This section of the development plan contains a list of task descriptions for the development and documentation of the SETL dataset software development tools. Because the development phases overlap, the work breakdown is presented functionally rather than sequentially.

3.5.1 Research and Document Requirements

The first phase of the development process shall be to research and document various requirements needed to continue into the design and implementation phases. These requirements will be either supplied by the user or specified by the development team and approved by the user. All necessary requirement specification tasks are listed regardless of whether they are supplied by the user or the development team and are as follows:

a. Research and prepare the *Project Software Functional Requirements Document*, PSFRD.

b. Research and prepare the *Software Functional Requirements Document*, SFRD(n), for each piece of software.

c. Research developer training requirements.

d. Research user training requirements.

e. Review requirements with user.

3.5.2 Design and Documentation

The development team will design and document the deliverable software. The design goal is to provide quality software tools for Engineering to develop and release SETL datasets. Following are the tasks identified for completion in this phase:

a. Train developer.

b. Analyze requirements.

c. Develop *Training and User Guide*, TUG.

d. Define and document *Software Development Environment Plan*, SDEP (see section 3.3).

e. Define and document *Software Guidelines*, SG.

f. Define and document *Desktop Procedures*, DP.

g. Prototype and report coding.

h. Conduct *Preliminary Design Review*, PDR.

i. Design and document software, including work done on existing Contact software and emulator (see section 2.3.2.1) .

j. Verify design.

k. Conduct *Critical Design Review*, CDR.

3.5.3 Implementation

The implementation phase is where the development team creates the environments required to develop software and control software configuration. Programmers will also be implementing the design within those environments. The tasks scheduled for completion in this phase are as follows:

a. Create development environment.

b. Create configuration management environment.

c. Code development tool software.

d. Verify software modules.

e. Integrate software.

3.5.4 Conduct and Document Testing

Testing is an integral part of the development process. This (1) includes verification testing in the office environment, and (2) validation testing in the laboratory with simulated test dataset activities using the tools. The tasks required are as follows:

a. Release *Software Verification and Test Procedure*, SVTP(n), for each piece of software.

b. Conduct *Test Readiness Review*.

c. Verify software meets requirements and design.

d. Release verification test results.

e. Validate software meets user needs.

f. Publish validation test results.

3.5.5 Release

In the release phase, the development team will request user approval of the released software product. The

users will base their approval on the results of verification and validation testing and on the completeness of the software documentation. The tasks required in this phase are as follows:

a. Release *Version Description Document*, VDD(n), for each piece of software.

b. Obtain release approval.

c. Release software to production.

d. Train user.

3.5.6 Management

The management phase of the project will begin from the start of the project through completion. Procedures and plans will be released incrementally during the start up phase of the project. Checks throughout the project will be made to verify that the procedures are being adhered to by the development team, and to gain visibility as to the project status. The procedures and plans required to be completed in this phase are as follows:

a. *Software Development Plan*, SDP.

b. *Software Configuration Management Plan*, SCMP.

c. *Software Quality Assurance Plan*, SQAP.

d. *Software Verification and Validation Plan*, SVP.

3.5.7 Work Breakdown Structure (WBS) Diagram

Figure 1 is a graphical representation of the work breakdown structure. It is a functional breakdown of work to be completed with little regard given to task sequence.

The alphanumeric identifiers are used as a cross-reference between the work breakdown structure and the network diagram that follows.

FIGURE 1 WORK BREAKDOWN STRUCTURE

3.6 Time and Cost Estimate

This subsection consists of man-hour estimate tables (tables 2 through 6, beginning on page 24) for each of the elements of the WBS (see page 22). They identify the flow time required to complete each of the tasks.

Staffing for S5 through S10 were based on a comparable complexity level with respect to S1, S2, and S4.

Staffing was added for the given reasons:

a. 12 man-months for project management the development of the software tools.

b. 10 man-months for development of project control plans and documents.

A total of 181 man-months are required for the development of the software tools.

TABLE 2 S1, FLAT FILE SETL SYNTAX CHECKER

REQUIREMENTS	CURRENT PROGRAMS	CURRENT SLOC	USED FOR	RE-USEABLE	REV/ADD REQUIRED	REV/ADD SLOC	MAN MONTHS
SETL RULE COMPL	N/A	N/A	N/A	0%	100%	5.0K	25.0
IFTS COMPATIBILITY	DS_PARSE	6K	747-400 IFTS COMPATIBILITY	100%	20%	1.6K	6.0
DATASET FRMT COMPL	CHECK	2K	747-400 SYNTAX/FORMAT VOCABULARY	100%	20%	0.4K	1.0
					SUB TOTAL	7.2K	32.0
					ADD 20% FOR UNCERTAINTY		7.0
					TOTAL		39.0

TABLE 3 S2, FLAT FILE REVISION UTILITIES

REQUIREMENTS	CURRENT PROGRAMS	CURRENT SLOC	USED FOR	RE-USEABLE	REV/ADD REQUIRED	REV/ADD SLOC	MAN MONTHS
ADD, REMOVE, EXTRACT	N/A	N/A	N/A	0%	100%	0.6K	1.5
UTILITIES MENU	N/A	N/A	N/A	0%	100%	0.3K	0.5
AUTO RENUMBERING	SNRLC	0.9K	OLD FILE NOT MAINTAINED AND BUGS IN THE PRGM	10%	100%	1.5K	5.0
					SUB TOTAL	2.4K	7.0
					ADD 20% FOR UNCERTAINTY		2.0
					TOTAL		9.0

TABLE 4 S3, SPECIAL DOCUMENTATION

REQUIREMENTS	CURRENT PROGRAMS	SLOC	USED FOR	RE-USEABLE	REV/ADD REQUIRED	SLOC	MAN MONTHS
USERS GUIDE FOR FILE TRANSFER S/W	D936U002	N/A	747-400 FILE TRANSFER	100%	20%	N/A	1.0
USERS GUIDE FOR EMULATION S/W	N/A	N/A	N/A	0%	100%	N/A	3.0
					TOTAL MM		4.0

TABLE 5 S4, X WINDOW EMULATION FROM PERSONAL COMPUTER; EMULATION RESIDENT ON H-P COMPUTER

REQUIREMENTS	CURRENT PROGRAMS	CURRENT SLOC	USED FOR	RE-USEABLE	REV/ADD REQUIRED	REV/ADD SLOC	MAN MONTHS
BIT MAP OF OID SCREEN	N/A	N/A	N/A	0%	100%	3.0K	13.0
GENERATE CHILD WIND FROM EMULATOR	N/A	N/A	N/A	0%	100%	2.0K	8.0
SEND MSGS TO SIM OID	OIS DRIVERS	6K	HARD WIRED OID	100%	15%	1.0K	4.0
					SUB TOTAL	6.0K	25.0
					ADD 20% FOR UNCERTAINTY		5.0
					TOTAL		30.0

NOTE: BUDGET REQUIRED FOR INITIAL PURCHASE OF X WINDOWS FOR HP COMPUTERS AND X TERMINAL EMULATORS FOR PC

TABLE 6 EFFORT VERSUS SOFTWARE PIECE, S(N)

Sn	PROTOTYPE	SRD	SDD	TEST PROCEDURE	CODE	TEST CONDUCT	USER GUIDE	VMI	MAN DAYS	MAN MONTHS
S1	20	114	194	125	125	142	14	7	741	39.0
S2	6	26	45	29	29	32	3	2	171	8.0
S3							76		76	4.0
S4	15	88	148	97	97	108	11	6	570	30.0
S5	4	23	39	26	26	29	3	2	152	8.0
S6	4	23	39	26	26	29	3	2	152	8.0
S7	4	23	39	26	26	29	3	1	152	8.0
S8	3	15	24	16	16	18	2	1	95	5.0
S9	15	88	148	97	97	108	11	6	570	30.0
S10	9	52	89	58	58	65	7	4	342	18.0
								SUBTOTAL:	3021	159.0
								PROJECT MANAGEMENT:	228	12.0
								PROJECT CONTROL PLANS & DOCUMENTS:	190	10.0
								TOTAL:	3439	181.0

3.7 Network Diagram

This subsection identifies the sequence of activities to complete the project and also the critical path to completion. The diagram (see figures 2 through 5) is generated using Timeline software using the WBS (see page 22) and the man-hour estimates (see page 23).

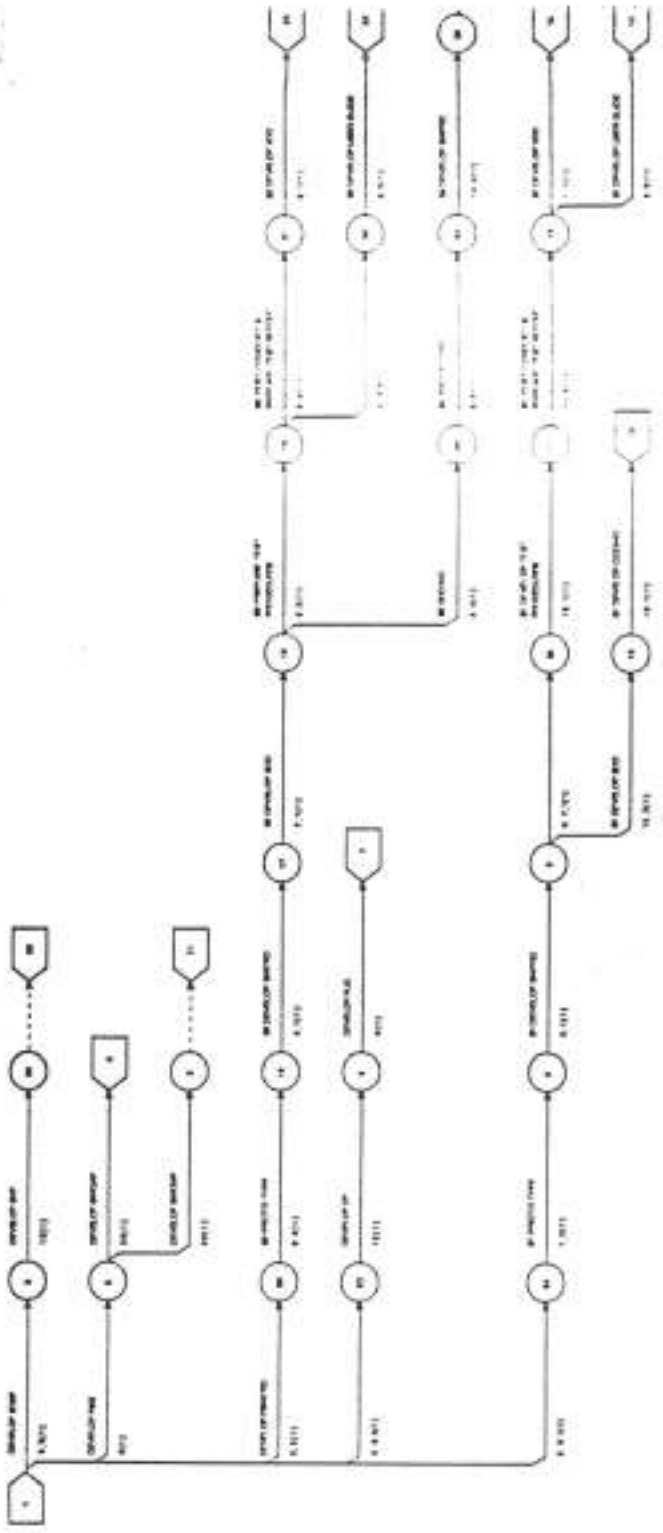


FIGURE 2 NETWORK DIAGRAM, PAGE 1

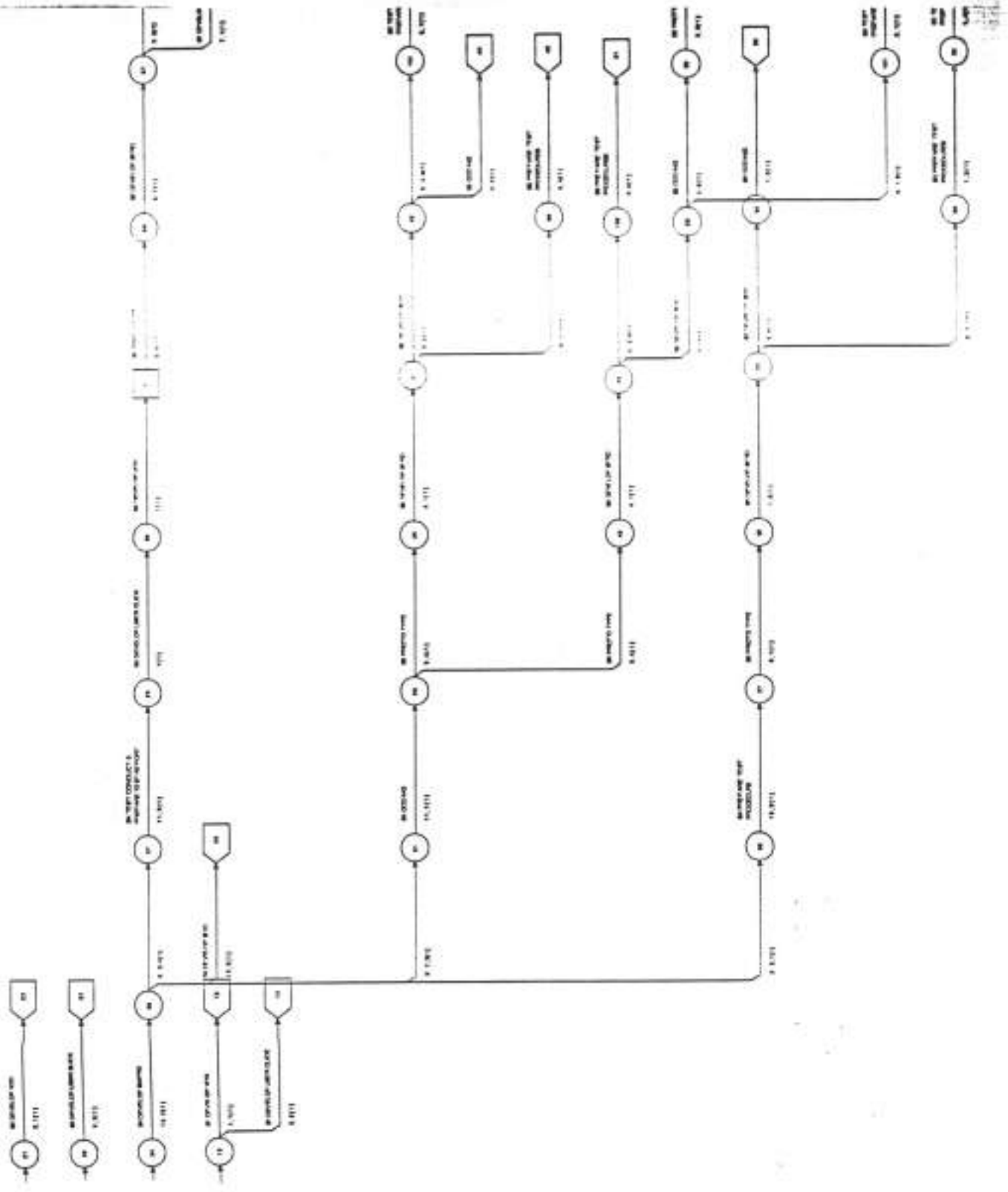


FIGURE 3 NETWORK DIAGRAM, PAGE 2

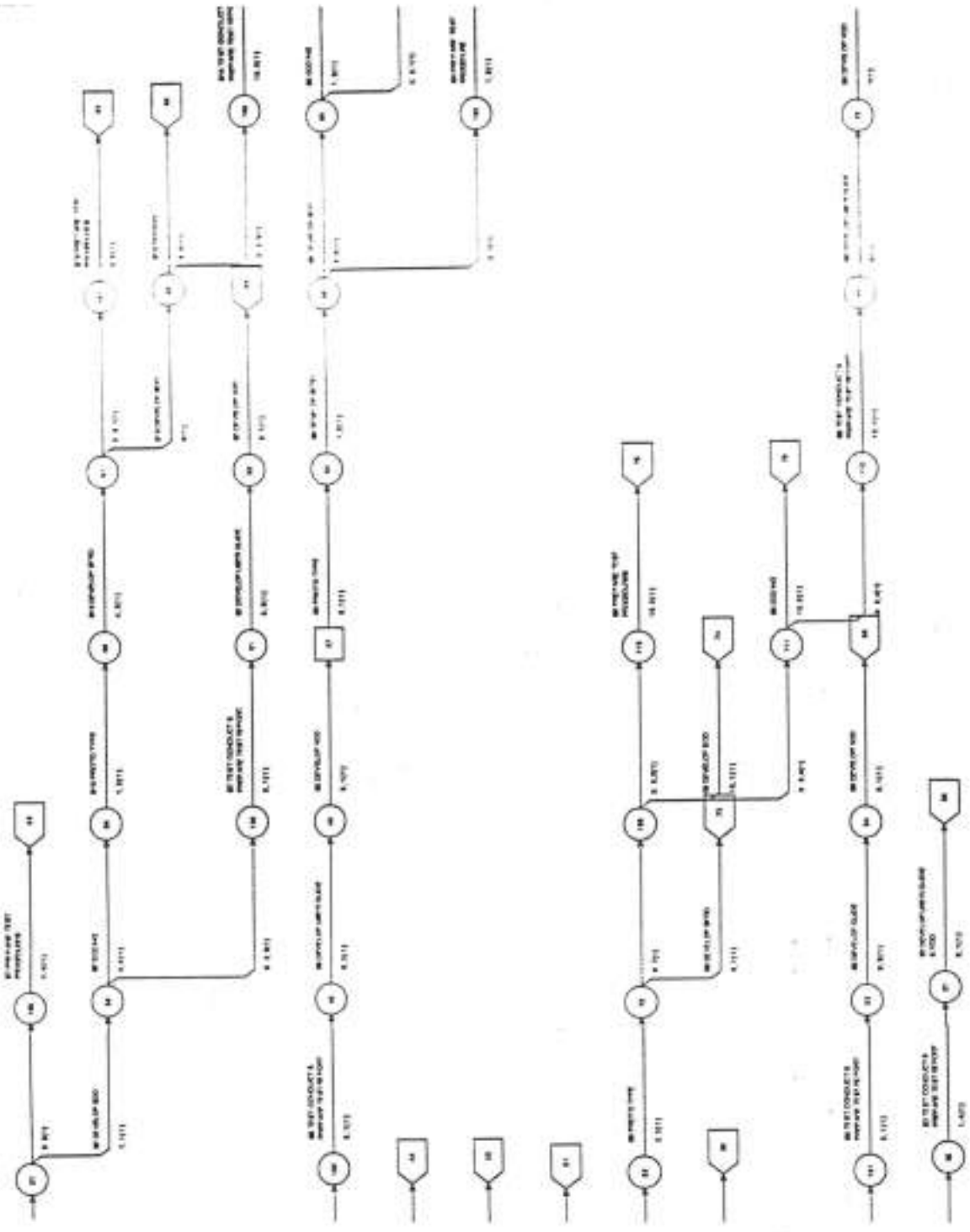


FIGURE 4 NETWORK DIAGRAM, PAGE 3

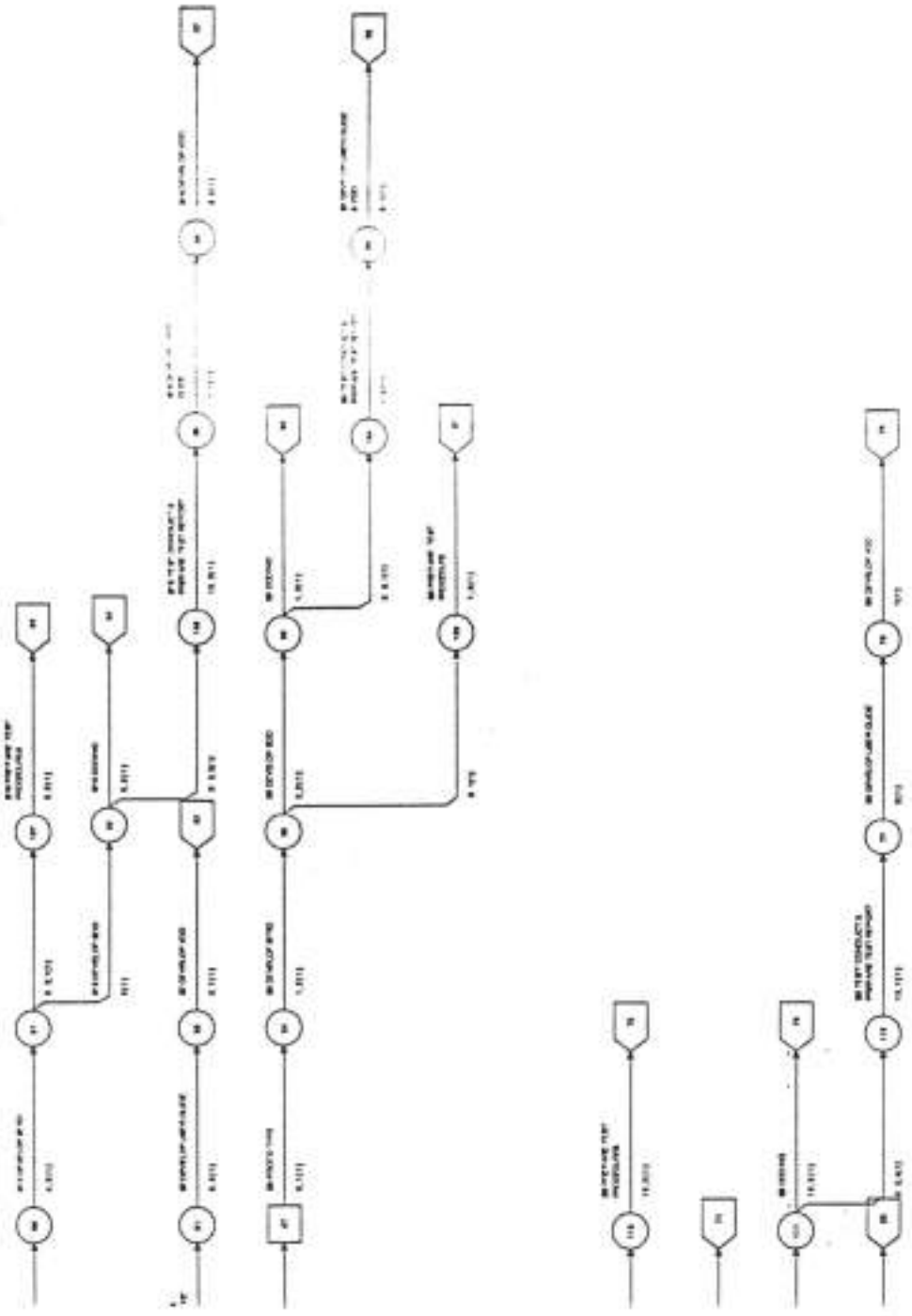


FIGURE 5 NETWORK DIAGRAM, PAGE 4

3.8 Resource Loading

This subsection consists of a diagram (see figure 6) that spreads the available or anticipated resources over the time available to complete this project.

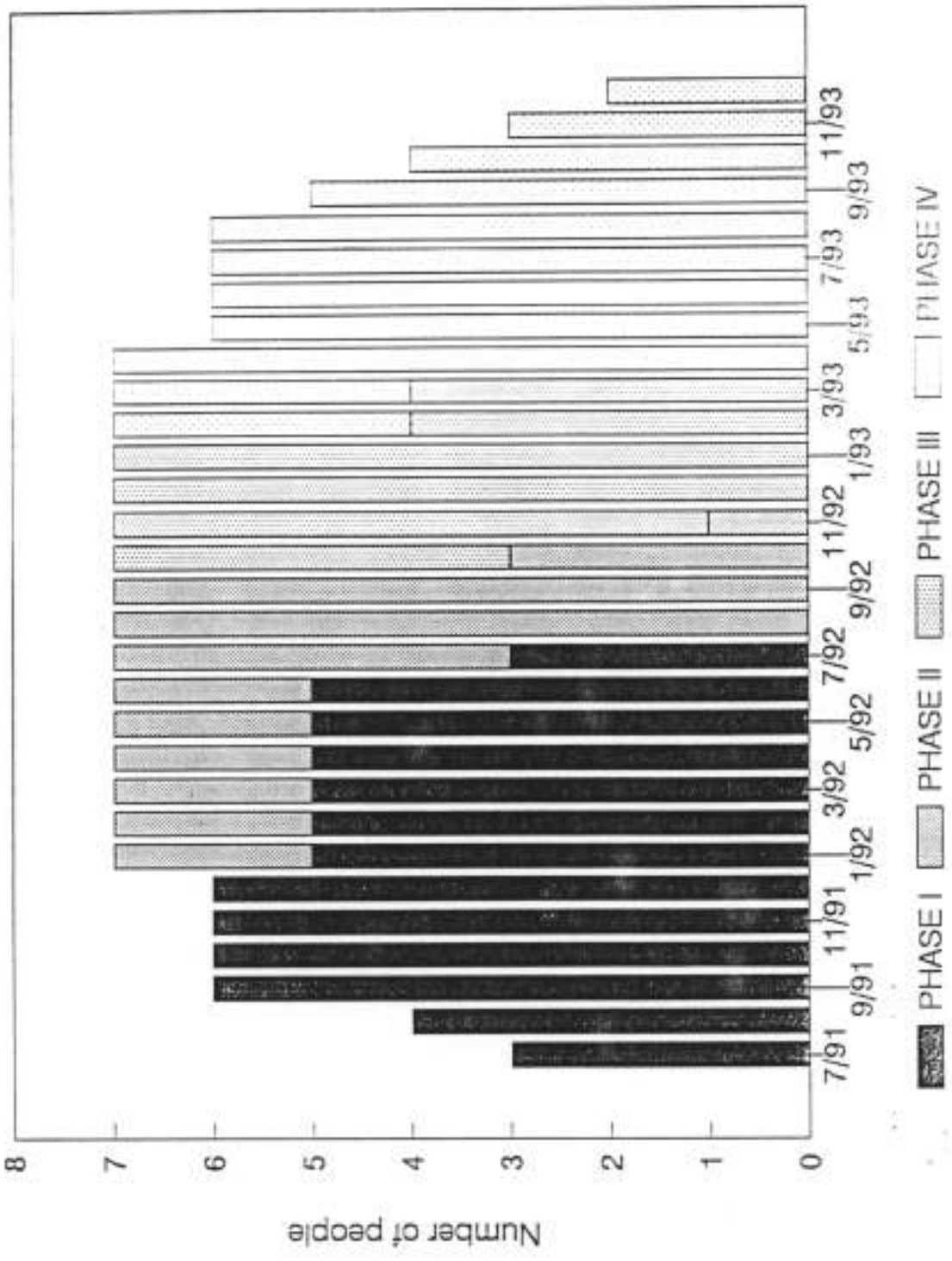


FIGURE 6 RESOURCE LOADING

3.9 Schedule

This subsection contains the schedules (see figures 7 through 17) based on the analysis of the data contained in sections 8, 9, and 10. This section provides the basis for subsequent project status and control.

Schedules will be maintained outside of this document.

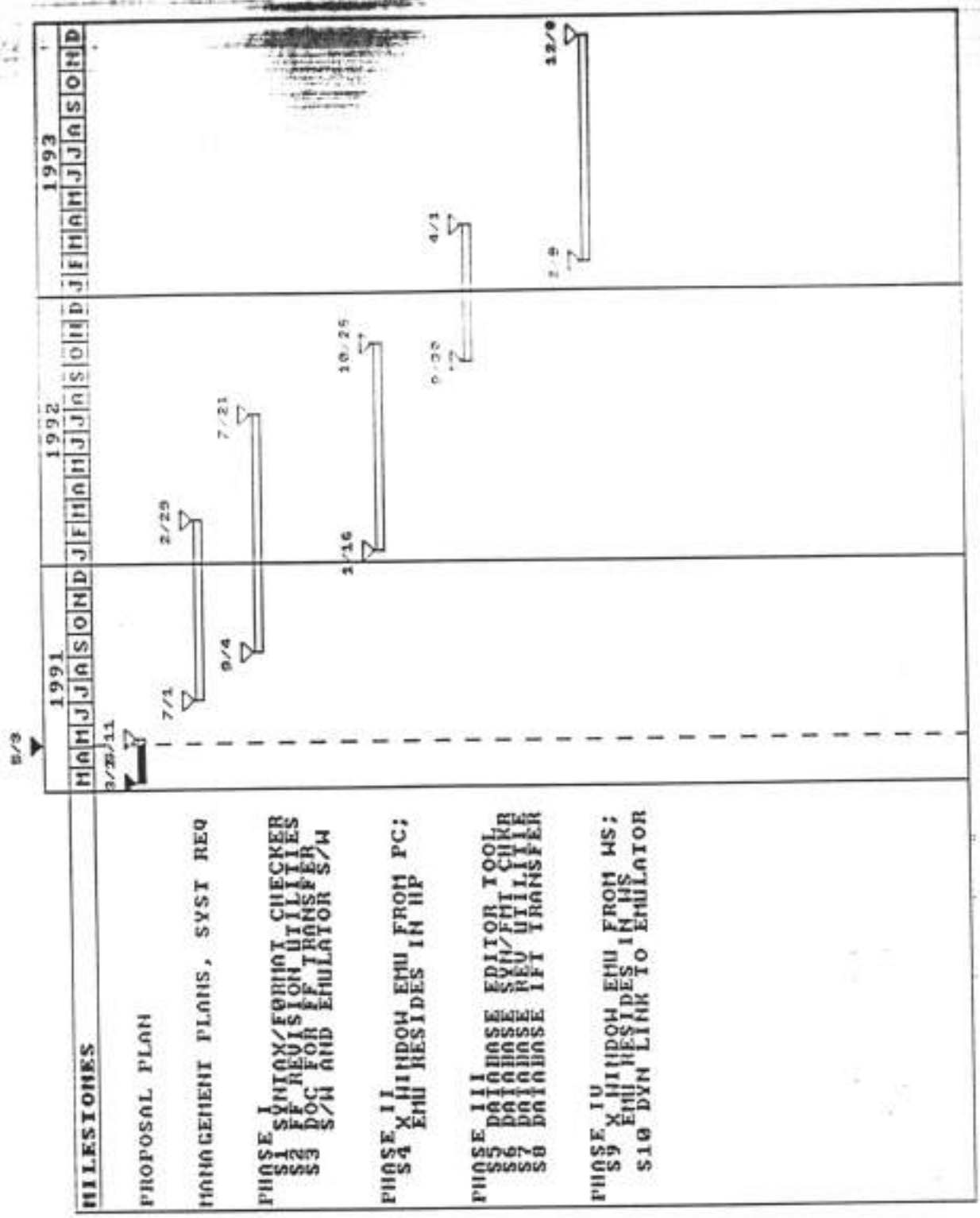


FIGURE 7 SCHEDULE, TIER I

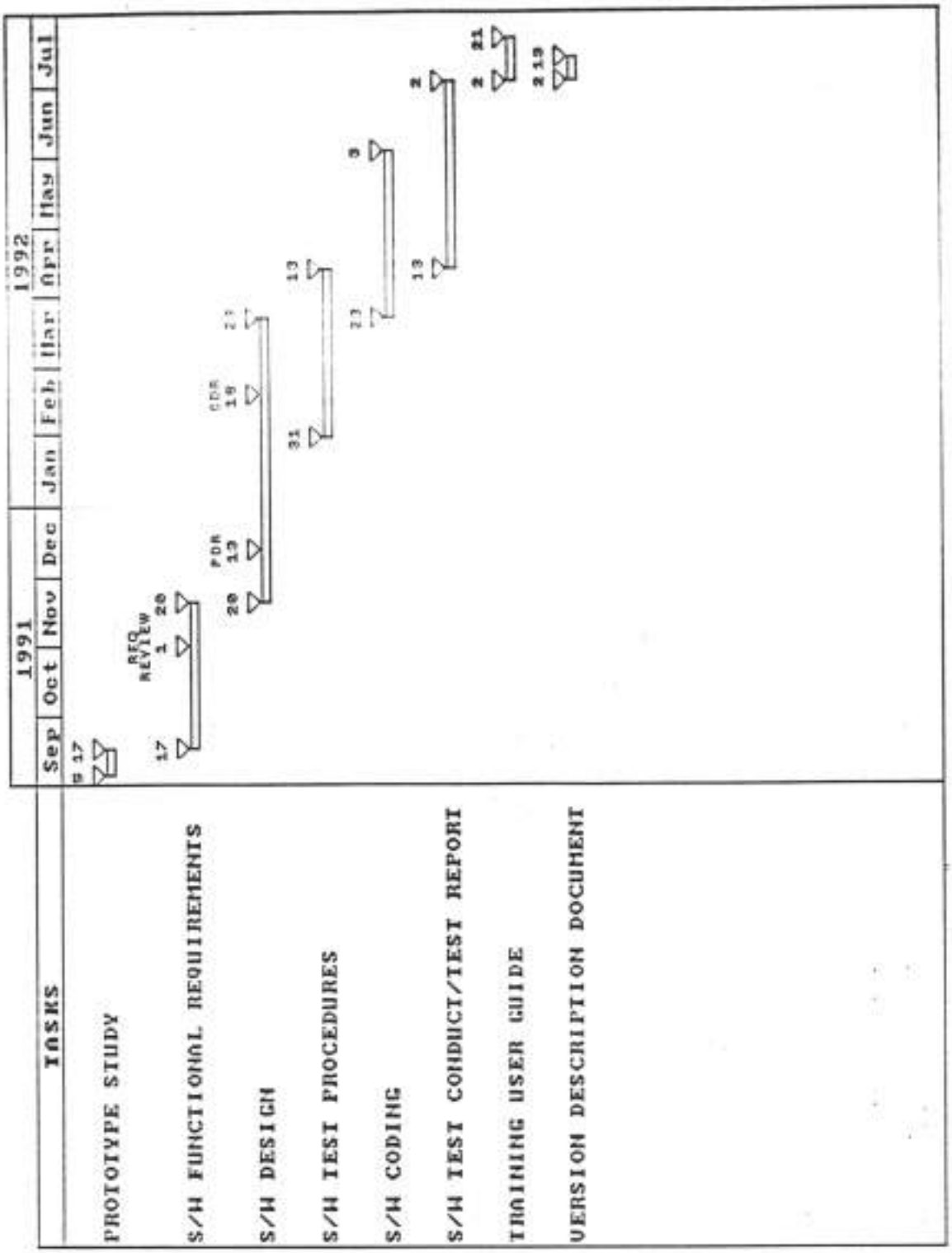


FIGURE 8 SCHEDULE, S1--FILE SYNTAX-FORMAT CHECKER

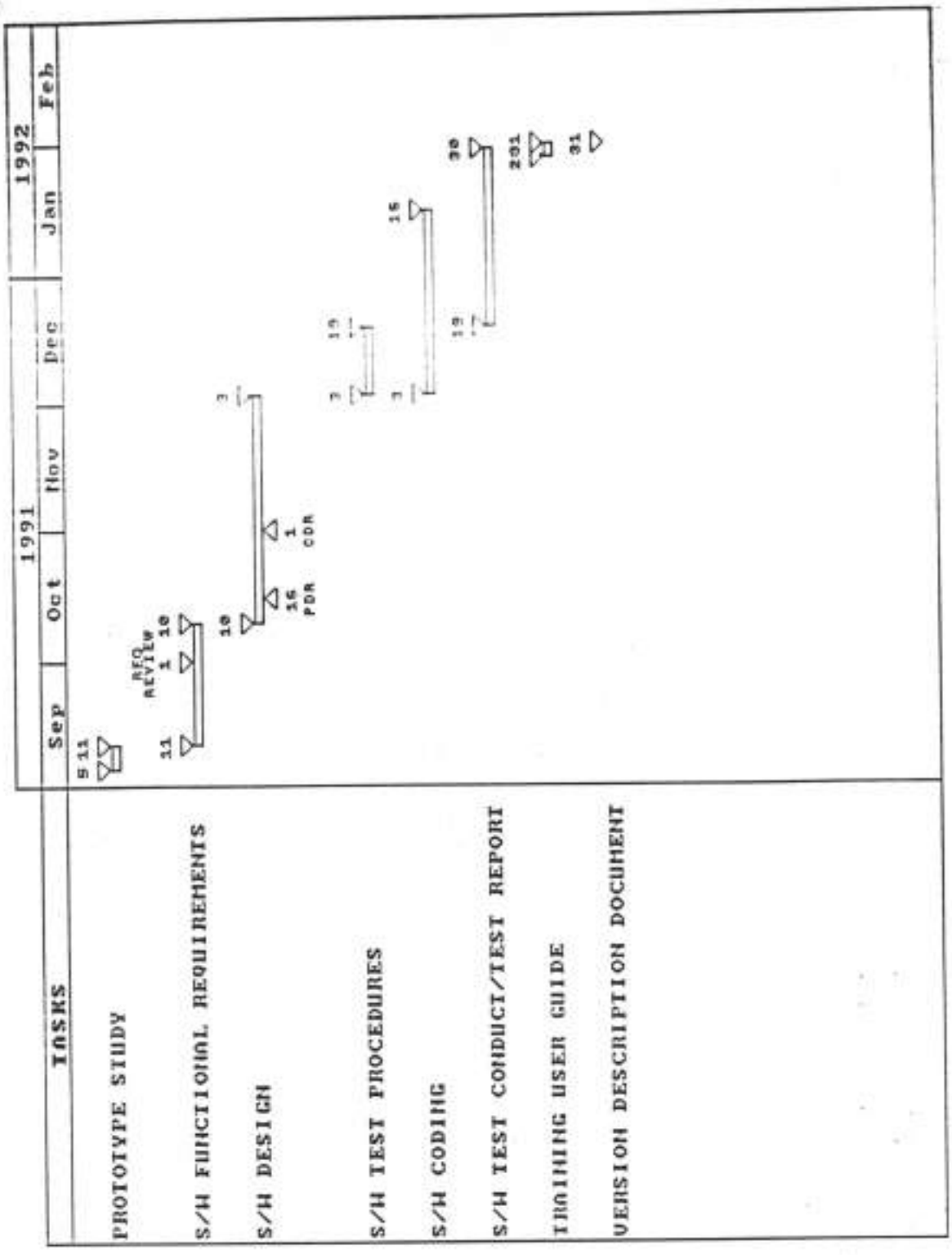


FIGURE 9 SCHEDULE, S2--FILE REVISION UTILITIES

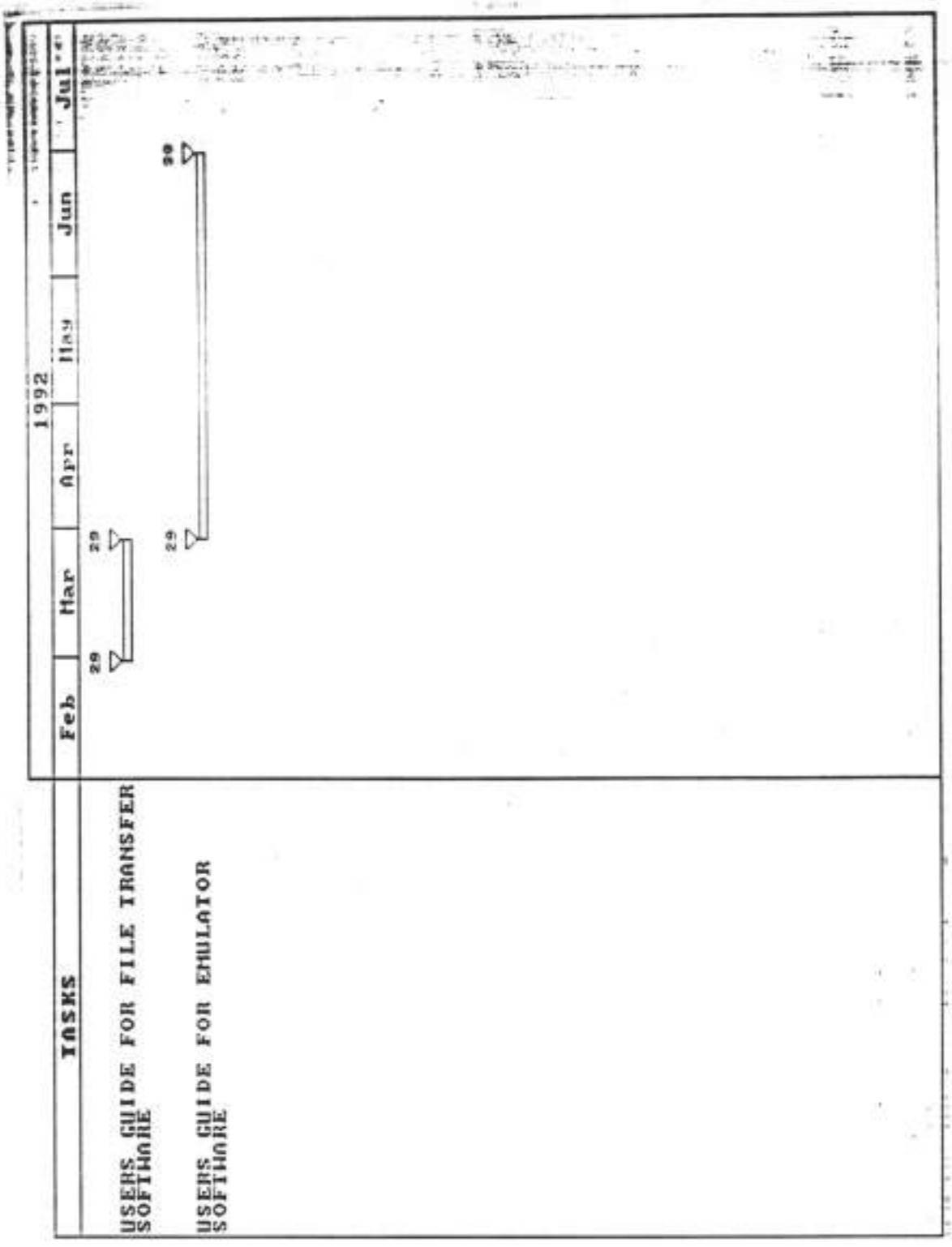


FIGURE 10 SCHEDULE, S3--SPECIAL DOCUMENTATION

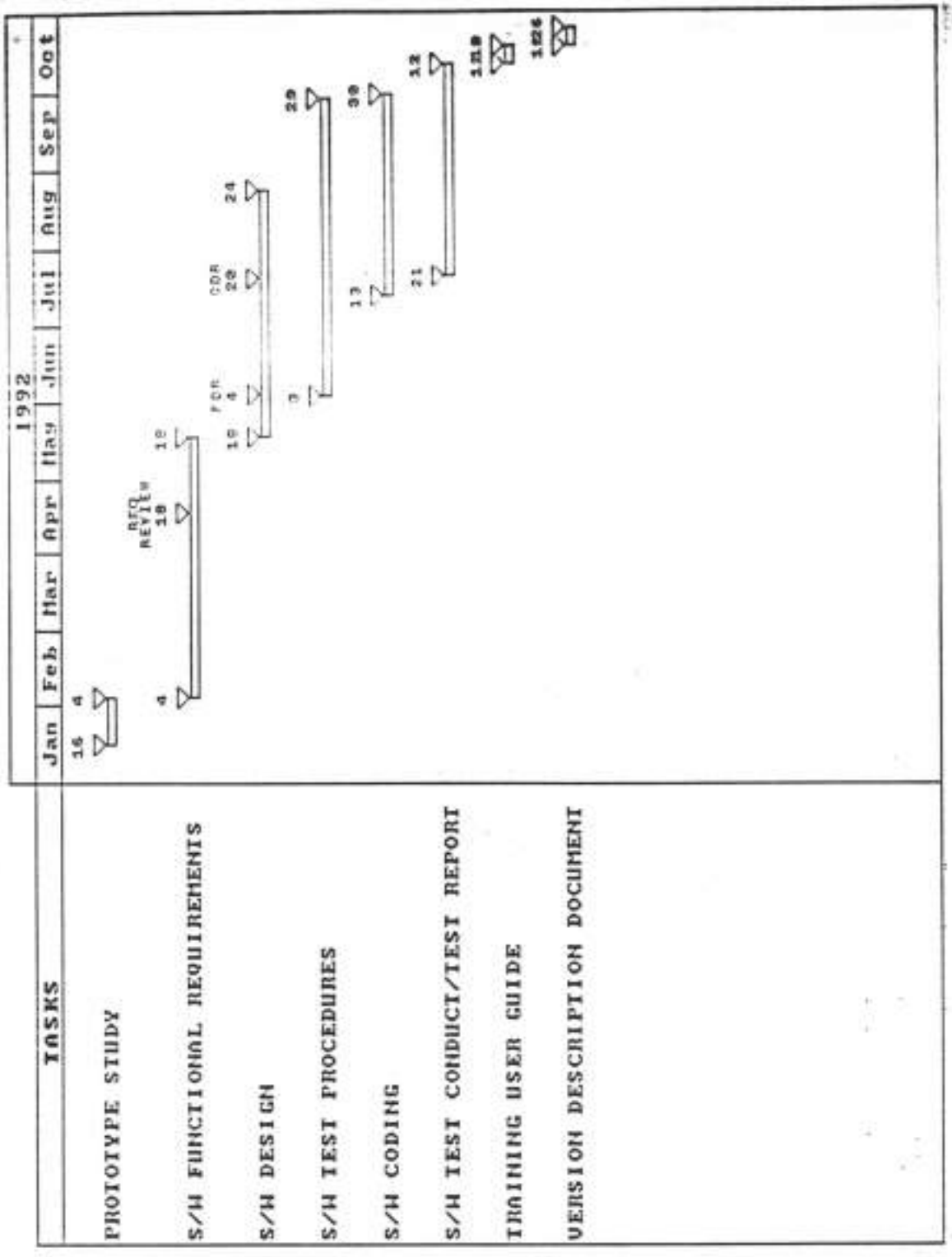


FIGURE 11 SCHEDULE, S4--X WINDOW EMULATION WITH EMULATOR RESIDENT ON H-P COMPUTER

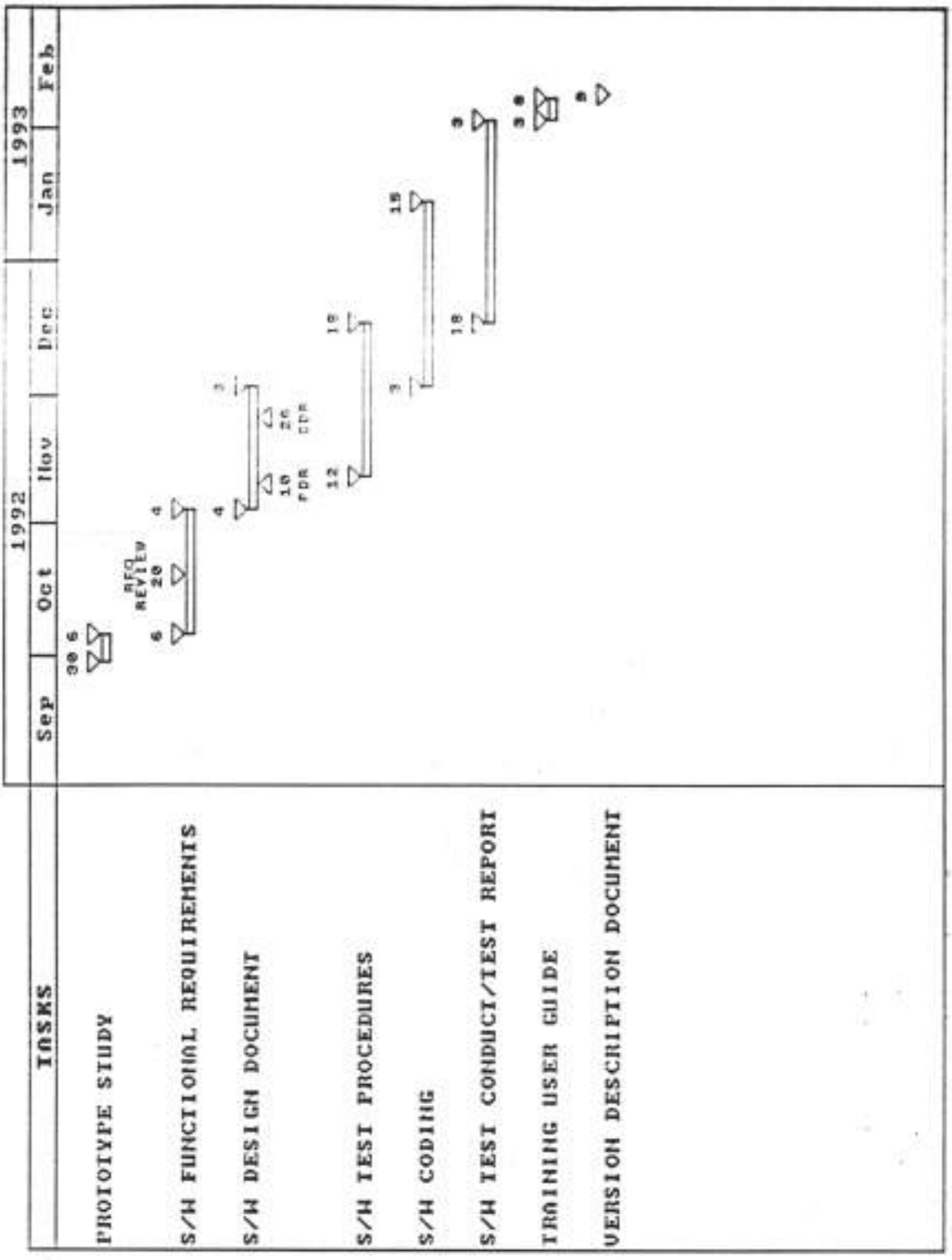


FIGURE 12 SCHEDULE, S5–CUSTOM EDITOR USING STANDARD DATABASE TOOLS

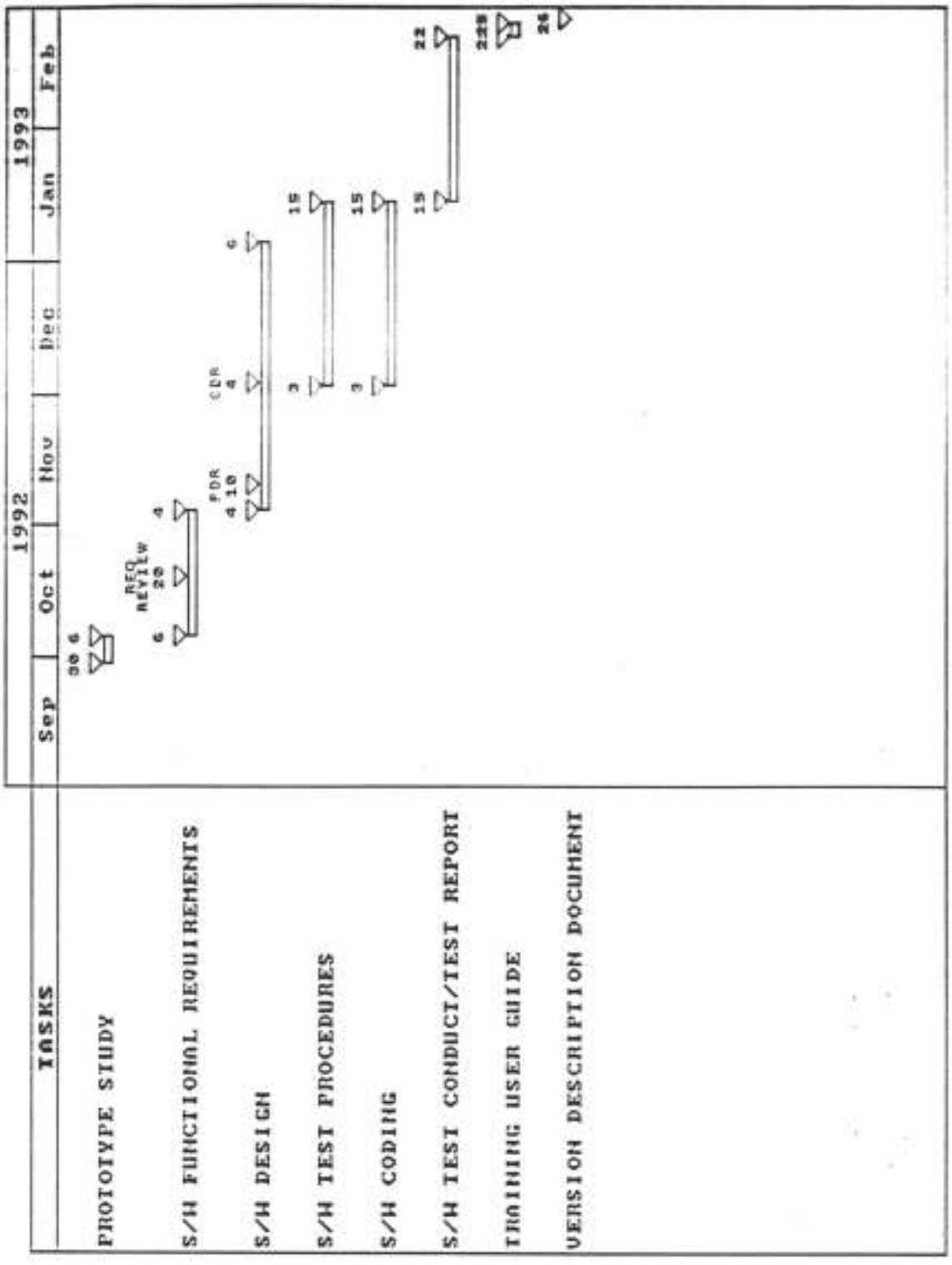


FIGURE 13 SCHEDULE, S6--DYNAMIC SYNTAX-FORMAT CHECKER INTEGRATED WITH EDITOR

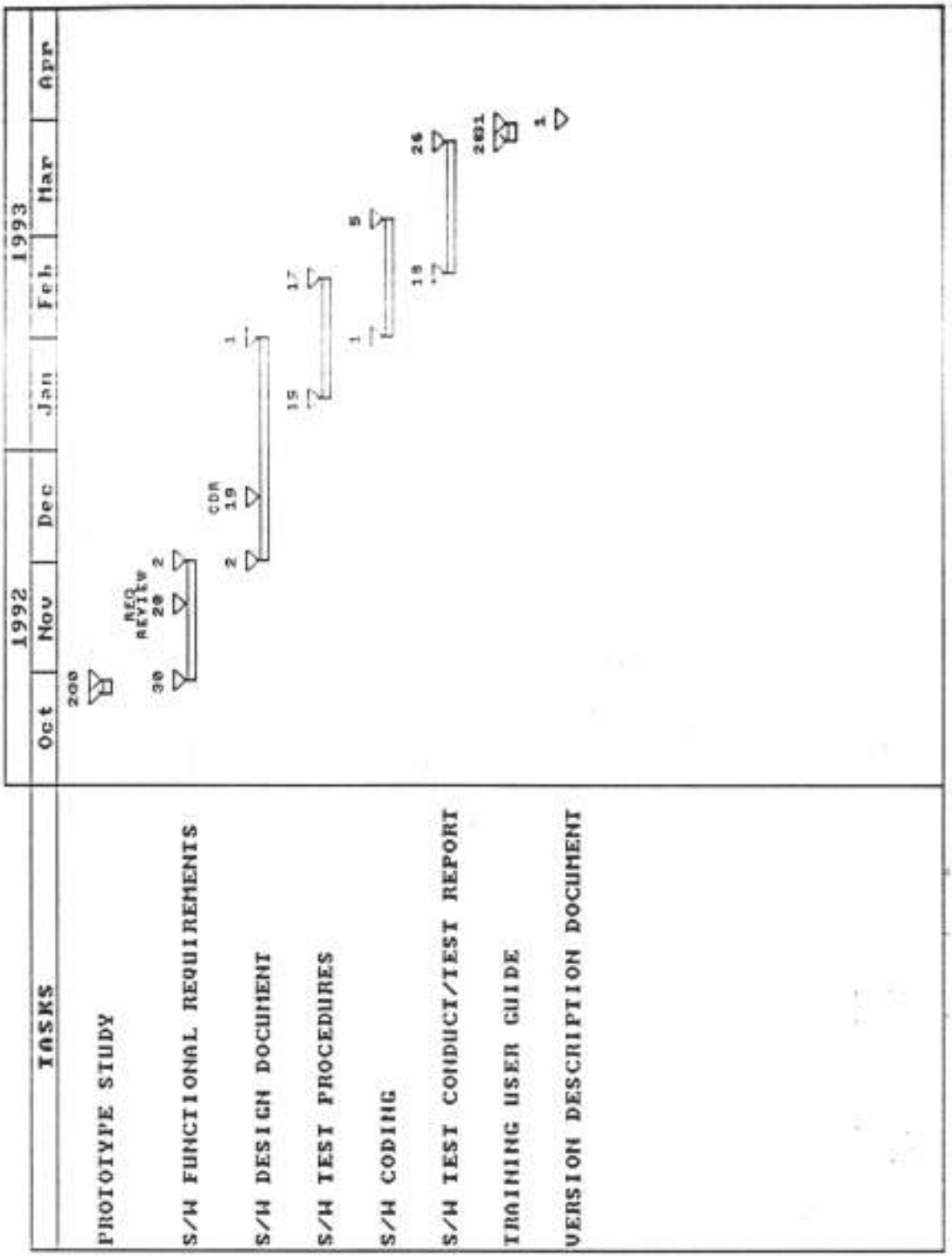


FIGURE 14 SCHEDULE, S7--DATASET REVISION UTILITIES INTEGRATED WITH EDITOR

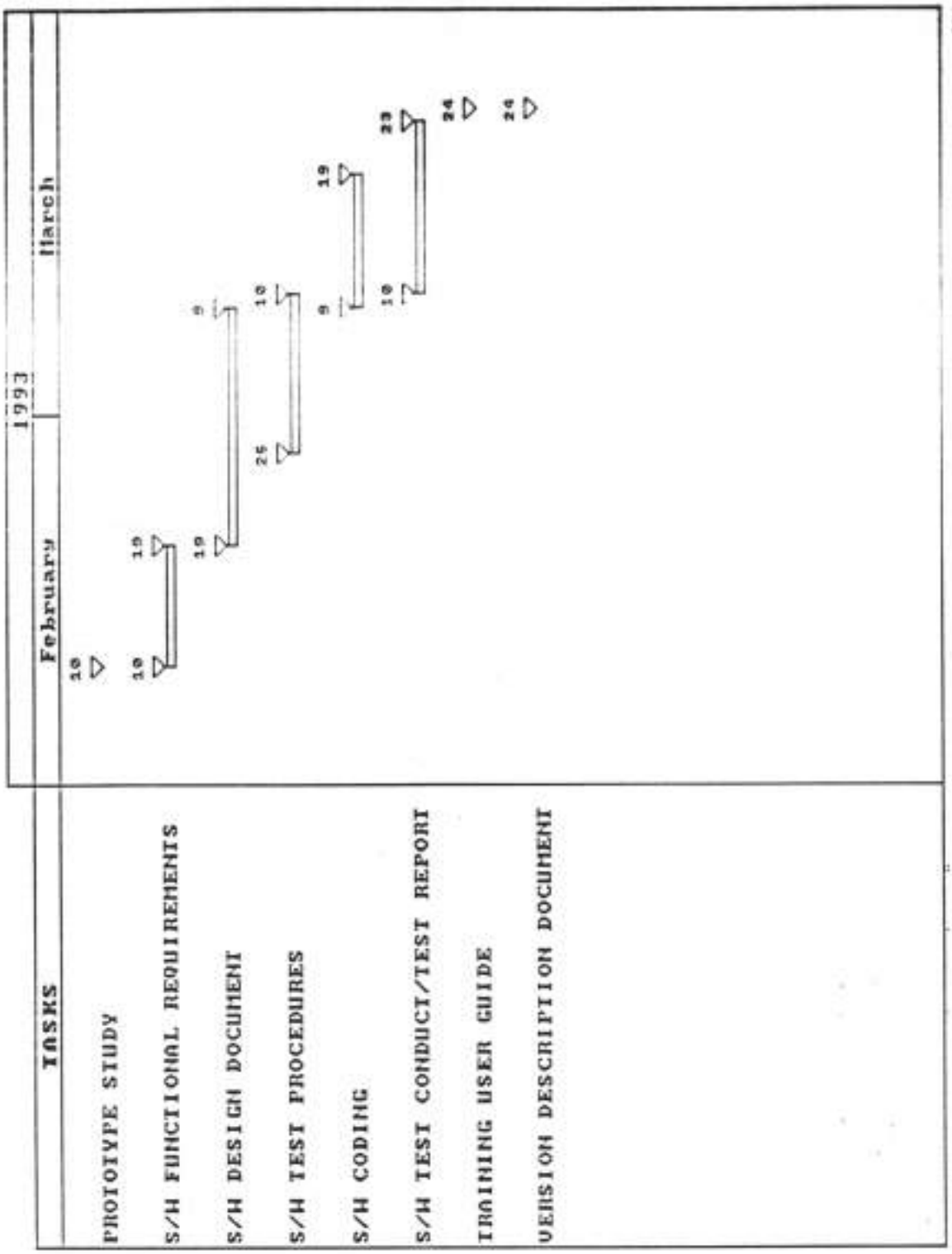


FIGURE 15 SCHEDULE, S8--DATABASE TRANSFER TO ENGINEERING WORKSTATION

FIGURE 16 SCHEDULE, S9--X WINDOW EMULATION WITH EMULATOR RESIDENT ON LOCAL WORKSTATION

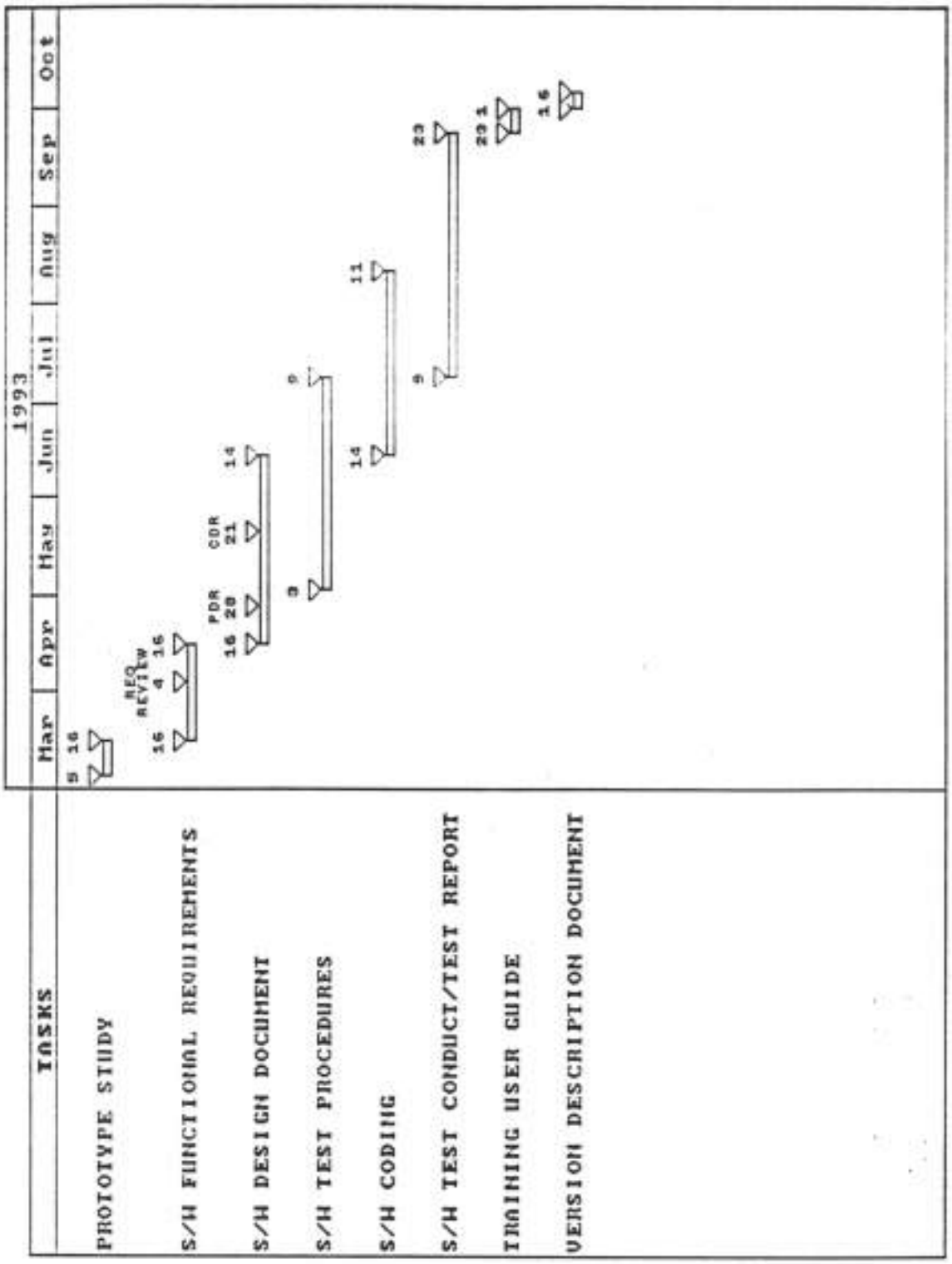


FIGURE 17 SCHEDULE, S10--DYNAMIC LINKS BETWEEN EDITOR AND EMULATOR

3.10 Deliverable Items Summary

This summary provides a single listing of all deliverable items found throughout this document.

3.10.1 Software (Source and Executable)

3.10.1.1 Phase I

a. Flat File Dataset Syntax-Format Checker (S1).

b. Flat File Dataset Revision Utilities (S2).

c. Users Guides (S3)--

(1) File transfer software--*Integrated Functional Test Data Management System (IFTDMS) Engineering Operator's Manual*, D935U002. This existing 747-400 document for Contact software needs modification for use with the 777 program.

(2) Emulator software--The existing emulator needs documentation for the purposes of the 777 program.

3.10.1.2 Phase II

X Window emulation software resident on a remote Hewlett-Packard computer--simulated OID on a Design Engineering personal computer using X Window (S4).

3.10.1.3 Phase III

a. Custom Editor using Standard Database Tools (S5).

b. Dynamic Syntax-Format Check Routine Integrated with the Editor (S6).

c. Dataset Revision Utilities Integrated with the Editor (S7).

d. Custom Database IFTS Transfer (S8).

3.10.1.4 Phase IV

a. X Window Emulation Software for the Emulator Resident on the Workstation--simulated operator interface device on Engineering workstation using X Window (S9).

b. Dynamic ("Hot") Links between the Editor and simulated OID (S10).

3.10.2 Documentation

Related documentation deliverable items may be combined under a single document number when clarity and ease of writing and/or use is determined to be improved.

a. "Software Development Plan," SDP, found herein.

b. *Software Configuration Management Plan*, SCMP.

c. *Software Quality Assurance Plan*, SQAP.

d. *Software Verification and Validation Plan*, SVP.

e. *Project Software Functional Requirements Document*, PSFRD.

f. Software control document(s)--

(1) *Software Development Environment Plan*, SDEP.

(2) *Software Guidelines*, SG.

(3) *Project Desktop Procedures*, DP.

g. *Software Functional Requirements Document*, SFRD(n), for each S(n).

h. *Software Design Documents*, SDD, for each S(n).

i. *Software Verification Test Procedures*, SVTP, for each S(n).

j. *Training and User Guide*, TUG, for each S(n).

k. *Version Description Document*, VDD(n), for each S(n).

4. SYSTEM ARCHITECTURE

4.1 Phase I

Phase I is developed as a baseline to support the System Integrated Laboratory (SIL) testing, and the first airplane. Reference figure 18.

a. Design Engineering Environment--

(1) A personal computer using Contact software will be used to download the dataset from IFTS. Revisions to the dataset are made using a text editor. Flat file syntax and format check software is used to perform syntax and format checks on the revised dataset.

(2) File transfer protocol (FTP) on a personal computer will be used to transmit the dataset to a local Hewlett-Packard computer hard-wired to external Operator Interface Device for emulation. Emulator software is resident on the Hewlett-Packard computer.

b. Dataset Storage, Release and Acquisition--Datasets are stored in IFTS and transmitted via Contact software.

c. Functional Test Engineering Environment--same as Design Engineering.

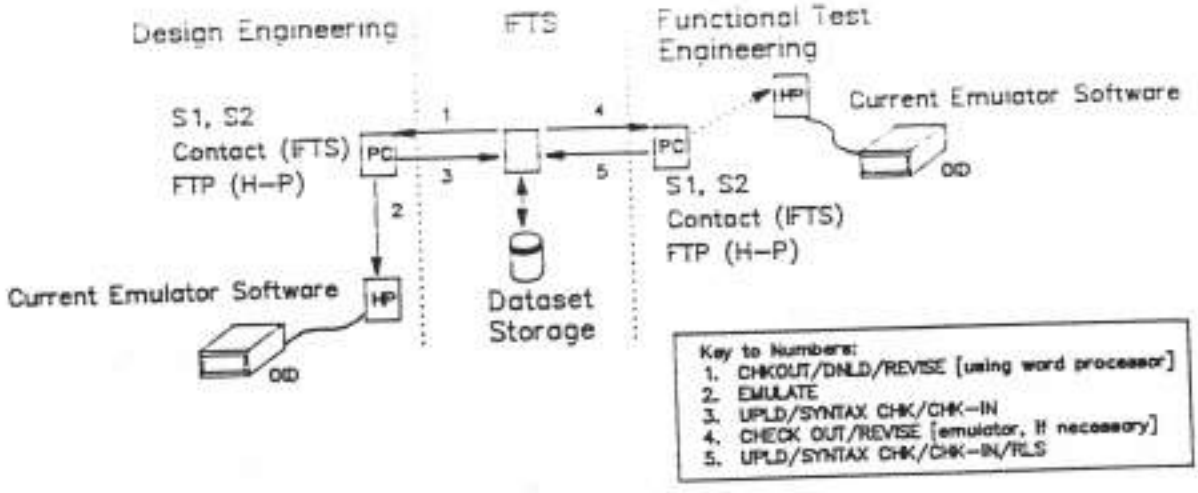


FIGURE 18 PHASE I ARCHITECTURE

4.2 Phase II

The external operator interface device is deleted, and the X Window environment (Reference 3)is introduced for emulation and development on Engineering personal computers. Reference figure 19.

a. Design Engineering Environment--

(1) A personal computer using Contact software will be used to download the dataset from IFTS. Revisions to the dataset are made using a text editor. Flat file syntax check software is used to perform syntax-format checks on the revised dataset.

(2) FTP on a personal computer will be used to transmit the dataset to a remote Hewlett-Packard computer for emulation. Emulator software is resident on the remote Hewlett-Packard computer. The X Window environment on the personal computer will be used for a simulated operator interface device.

b. Dataset Storage, Release and Acquisition--Datasets are stored in IFTS and transmitted via Contact software.

c. Functional Test Engineering Environment--same as Design Engineering.

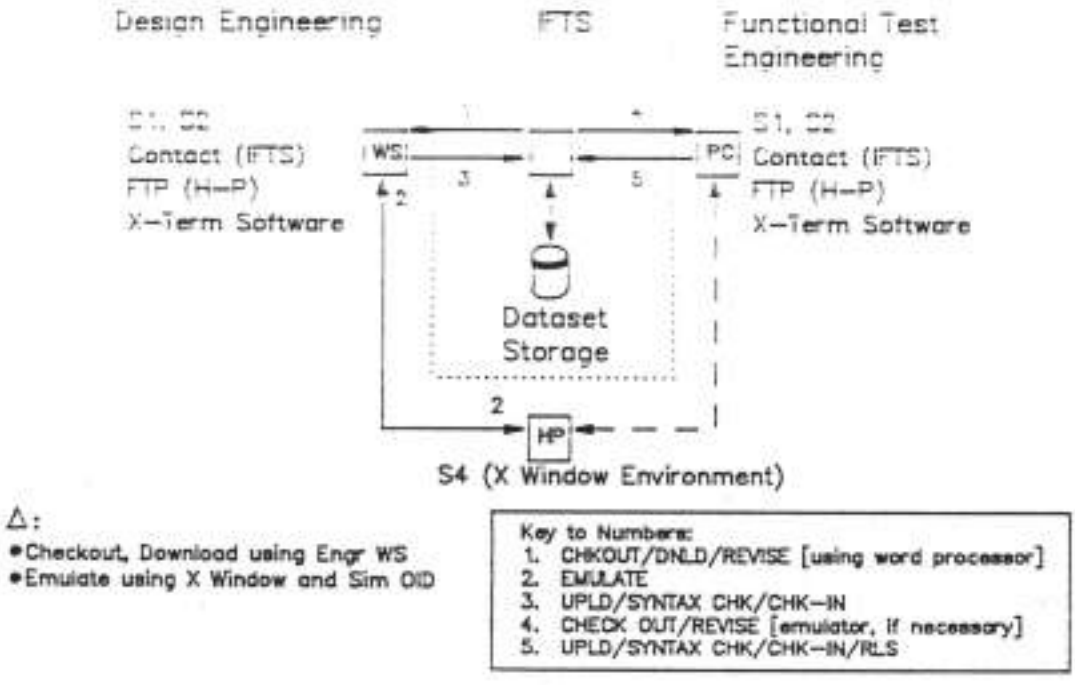


FIGURE 19 PHASE II ARCHITECTURE

4.3 Phase III

The datasets are developed, maintained, and transferred in database format. Also, in Phase III, syntax and format checks are performed dynamically. Reference figure 20.

a. Design Engineering Environment--

(1) A workstation using database file transfer software will be used to download the dataset from IFTS. Revisions to the dataset are made using a database application. Syntax and format checks of datasets are performed dynamically while editing.

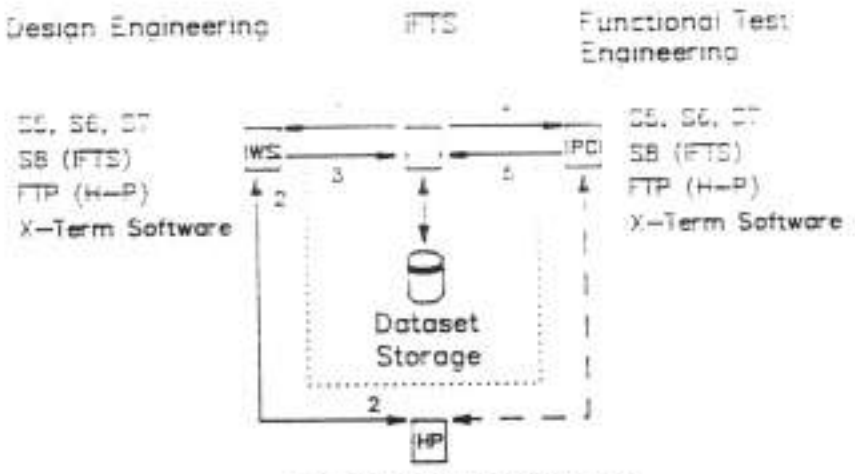
(2) FTP on a workstation will be used to transmit the dataset to a remote Hewlett-Packard computer for emulation. Emulator software is resident on the remote computer. The X Window environment on a workstation will be used for a simulated Operator Interface Device.

b. Dataset Storage, Release and Acquisition--Datasets are stored in IFTS and transmitted via a database file transfer software.

c. Functional Test Engineering Environment--

(1) A personal computer using database file transfer software will be used to download the dataset from IFTS. Revisions to the dataset are made using a database application. Syntax and format checks of datasets are performed dynamically while editing.

(2) FTP on a personal computer will be used to transmit the dataset to a remote Hewlett-Packard computer for emulation. The emulator software is resident on the remote computer. X Window will be used for a simulated operator interface device.



S4 (X Window Environment)

Δ:
- Download test imported into WS database
- Database environment provides data entry, syntax checking, step attribution

Key to Numbers:
1. CHKOUT/DNLD/REVISE [using word processor]
2. EMULATE
3. UPLD/SYNTAX CHK/CHK-IN
4. CHECK OUT/REVISE [emulator, if necessary]
5. UPLD/SYNTAX CHK/CHK-IN/RLS

FIGURE 20 PHASE III ARCHITECTURE

4.4 Phase IV

The emulator is ported to the local workstation or personal computer, the remote Hewlett-Packard computer is eliminated, and dynamic links between the editor and the emulator are provided. Reference figure 21.

a. Design Engineering Environment--

(1) A workstation using database file transfer software will be used to download the dataset from IFTS. Revisions to the dataset are made using a database application. Syntax and format checks are performed dynamically while editing.

(2) The emulator software is resident on the local workstation. The X Window environment will be used for the simulated OID.

b. Dataset Storage, Release and Acquisition--Datasets are stored in IFTS and transmitted via database file transfer software.

c. Functional Test Engineering Environment--

(1) A personal computer using database file transfer software will be used to download the dataset from IFTS. Revisions to the dataset are made using a database application. Syntax and format checks performed dynamically while editing.

(2) Emulator software is resident on local personal computer. The X Window environment will be used for the simulated OID.

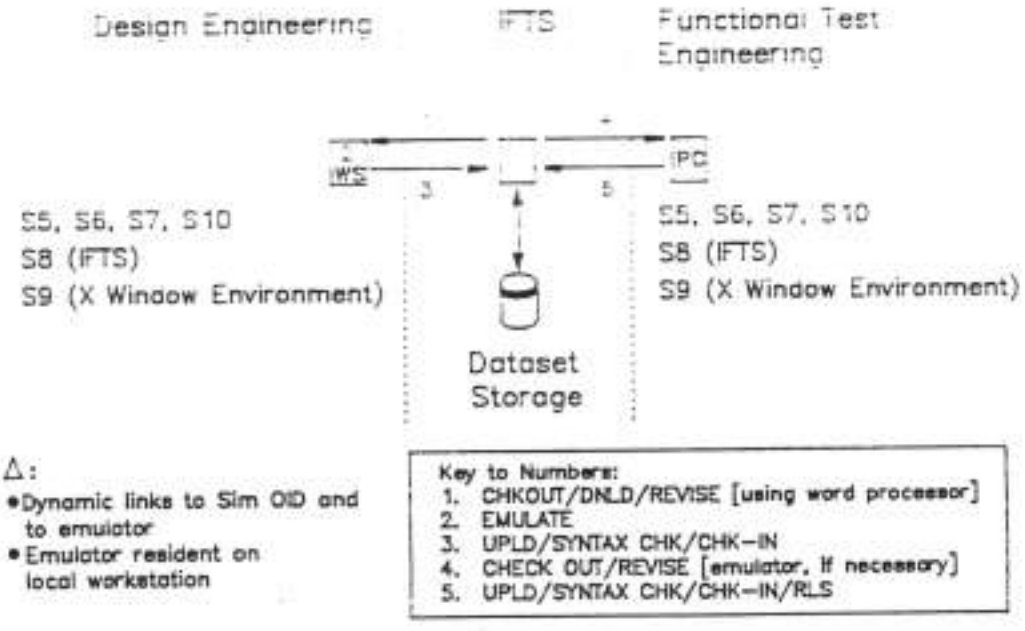


FIGURE 21 PHASE IV ARCHITECTURE

5. PROCESS FLOW DESCRIPTION

5.1 Phase I

Figure 22 illustrates the Phase I process flow.

a. Design Engineering checks out the dataset from IFTS and downloads it to a personal computer using Contact software, revises, and performs syntax checking of the datasets.

b. Design Engineering downloads the revised dataset to a local Hewlett-Packard computer and verifies the dataset by emulation on the Hewlett-Packard computer using a hard-wired external OID.

c. Design Engineering transmits the revised dataset from a personal computer to IFTS using Contact software, checks it in, and queues Functional Test Engineering.

d. Functional Test Engineering checks out the revised dataset from IFTS, downloads it to a personal computer using Contact software, emulates the dataset when required, prepares the dataset for release, and checks the dataset for syntax errors.

e. Functional Test Engineering then uses Contact software to upload and release the dataset to IFTS.

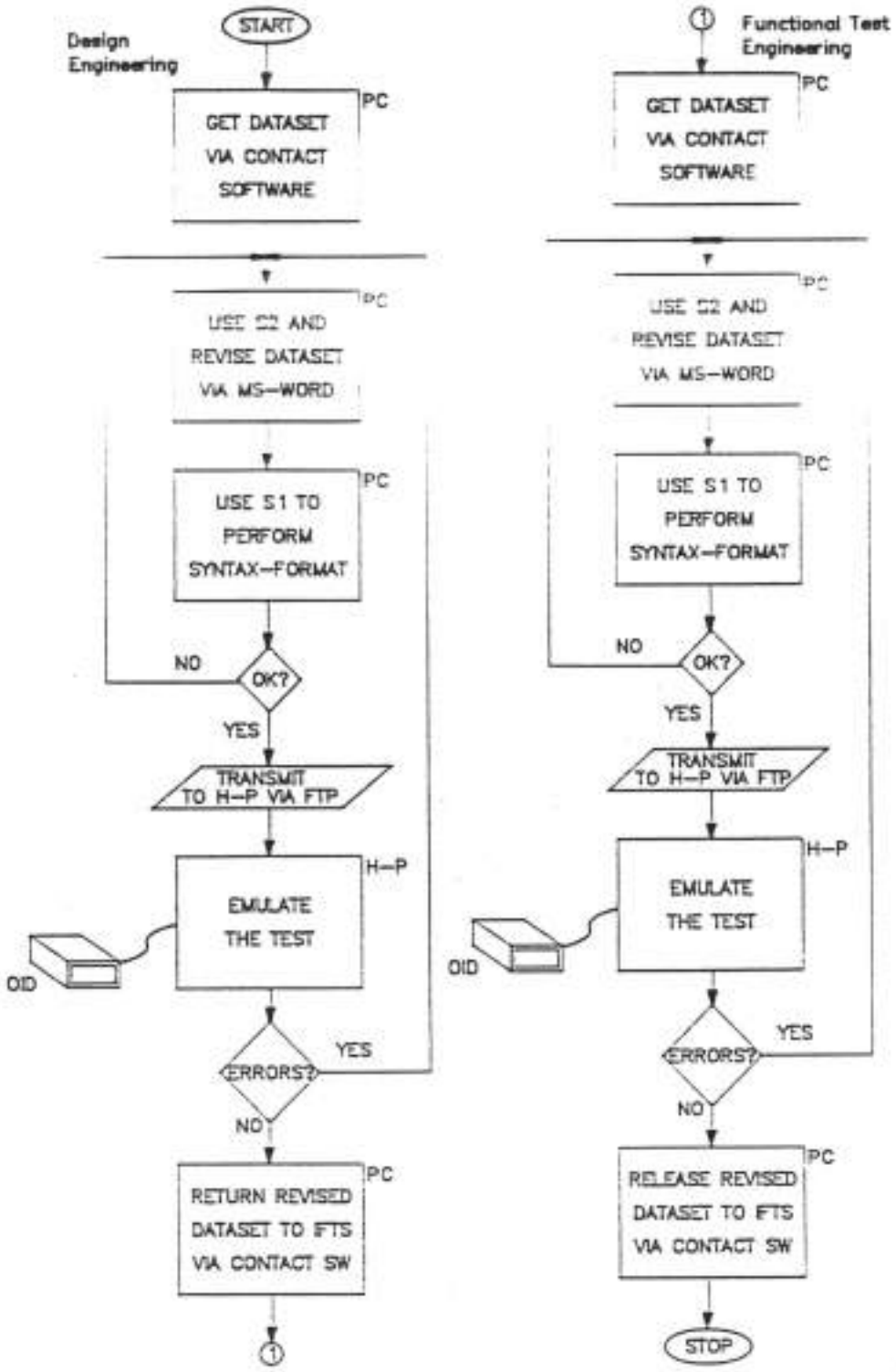


FIGURE 22 PHASE I DESCRIPTION OF PROCESS FLOW

5.2 Phase II

Figure 23 illustrates the Phase II process flow.

a. Design Engineering checks out the dataset from IFTS and downloads it to a personal computer using Contact software, revises, and performs syntax checking of the datasets.

b. Design Engineering downloads the revised dataset to a remote Hewlett-Packard computer and verifies the dataset by emulating it on the Hewlett-Packard computer from the personal computer.

c. Design Engineering transmits the revised dataset from a personal computer to IFTS using file transfer software, and queues Functional Test Engineering.

d. Functional Test Engineering checks out the revised dataset from IFTS, downloads it to a personal computer using Contact software, emulates the dataset when required, prepares the dataset for release, and checks the dataset for syntax errors.

e. Functional Test Engineering then uses Contact software to upload and release the dataset to IFTS.

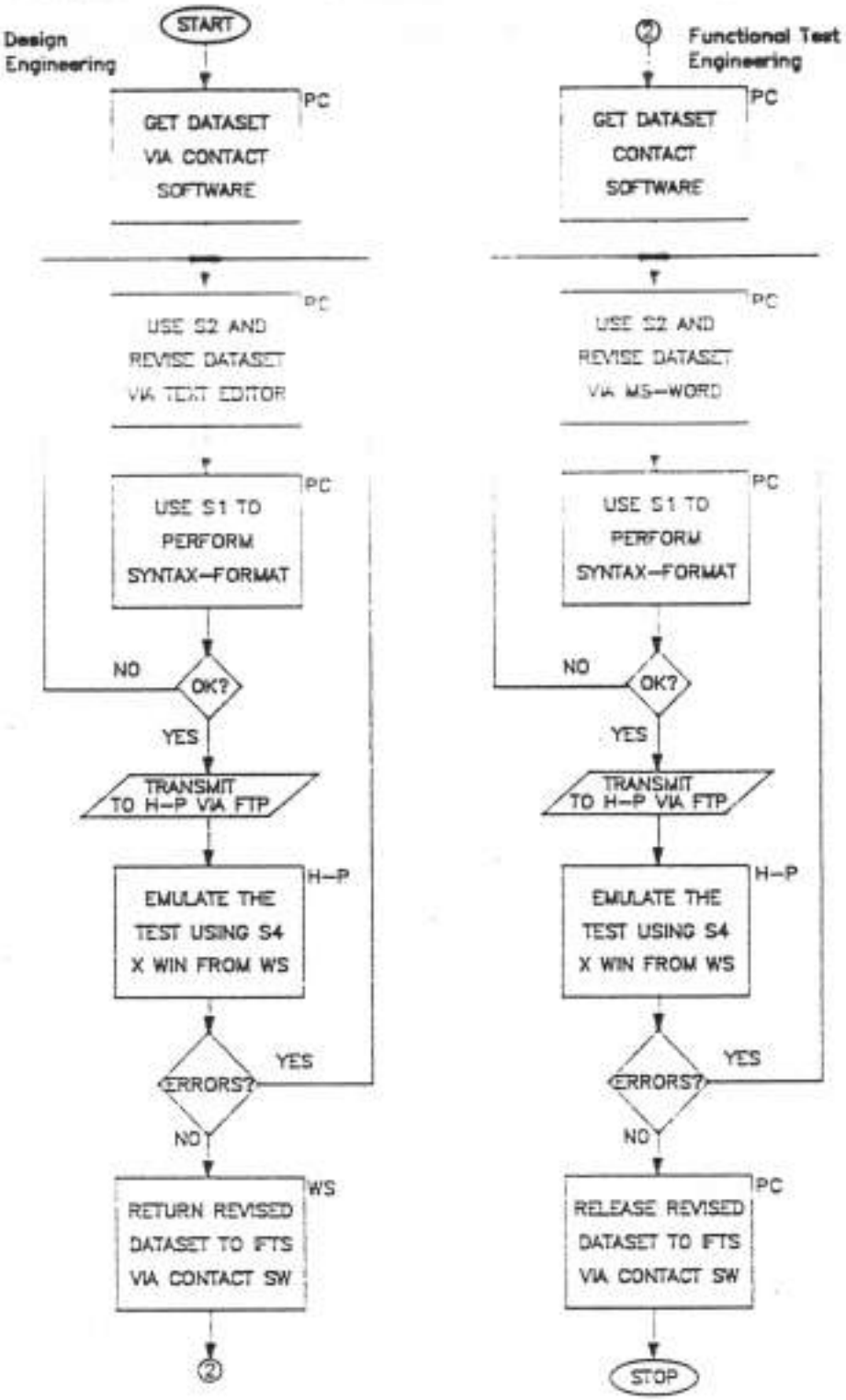


FIGURE 23 PHASE II DESCRIPTION OF PROCESS FLOW

5.3 Phase III

Figure 24 illustrates the Phase III process flow.

a. Design Engineering checks out the dataset from IFTS and downloads to a workstation using database file transfer software and revises the dataset.

b. Design Engineering downloads the revised dataset to a remote Hewlett-Packard computer and verifies the dataset by emulating on the Hewlett-Packard computer from the workstation.

c. Design Engineering transmits the revised dataset from the workstation to IFTS using database file transfer software, and queues Functional Test Engineering.

d. Functional Test Engineering checks out the revised dataset from IFTS, downloads it to a personal computer using database file transfer software, emulates the dataset when required, prepares the dataset for release, and checks the dataset for syntax errors.

e. Functional Test Engineering then uses database file transfer software to upload and release the dataset to IFTS.

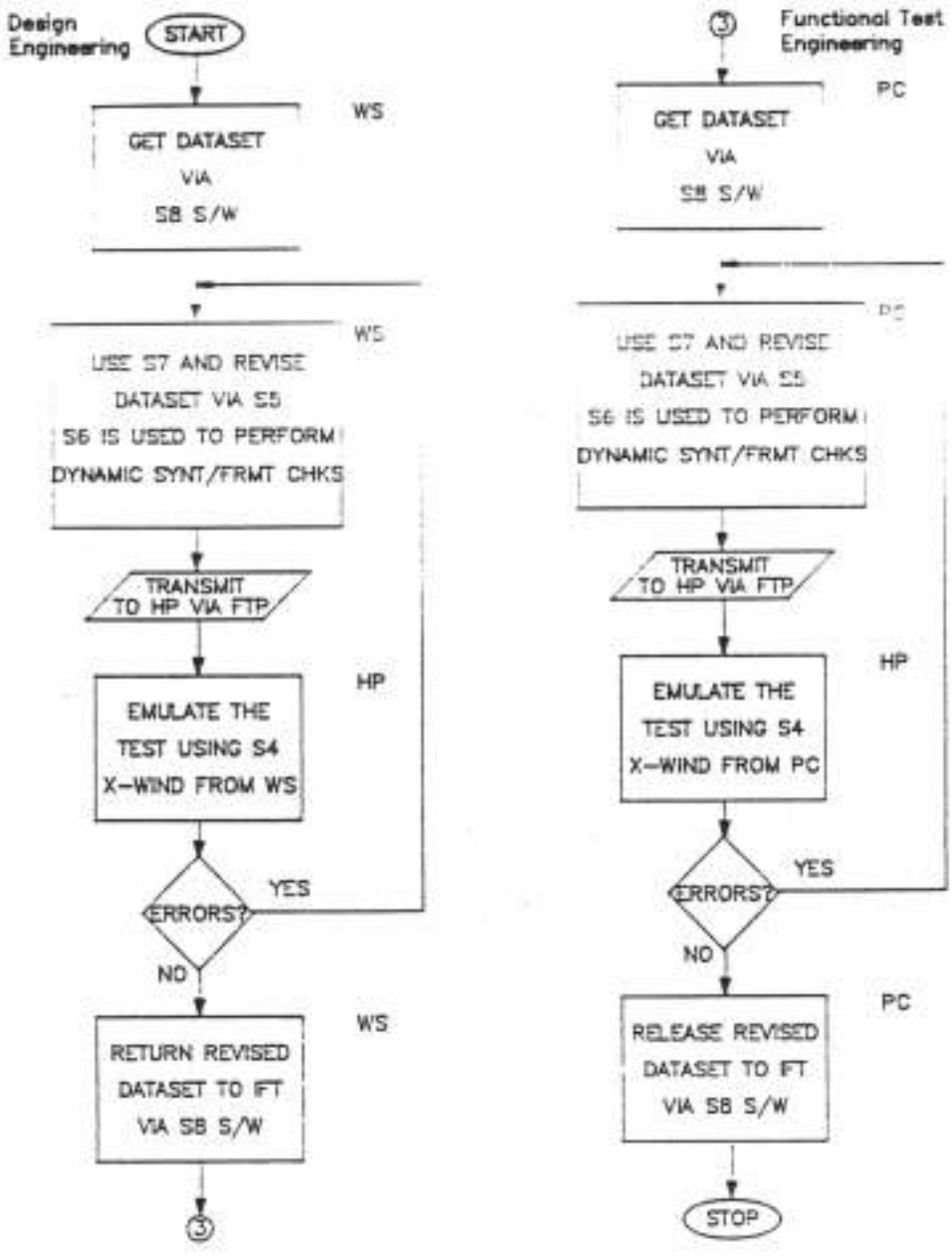


FIGURE 24 PHASE III DESCRIPTION OF PROCESS FLOW

5.4 Phase IV

Figure 25 illustrates the Phase IV process flow.

a. Design Engineering checks out the dataset from IFTS and downloads to workstation using database file transfer software and revises the dataset.

b. Design Engineering verifies the revised dataset by emulating in X Window environment on the workstation.

c. Design Engineering transmits the revised dataset from the workstation to IFTS using database file transfer software, and queues Functional Test Engineering.

d. Functional Test Engineering checks out the revised dataset from IFTS, downloads it to a personal computer using database file transfer software, emulates the dataset when required, prepares the dataset for release, and checks the dataset for syntax errors.

e. Functional Test Engineering then uses database file transfer software to upload and release the dataset to IFTS.

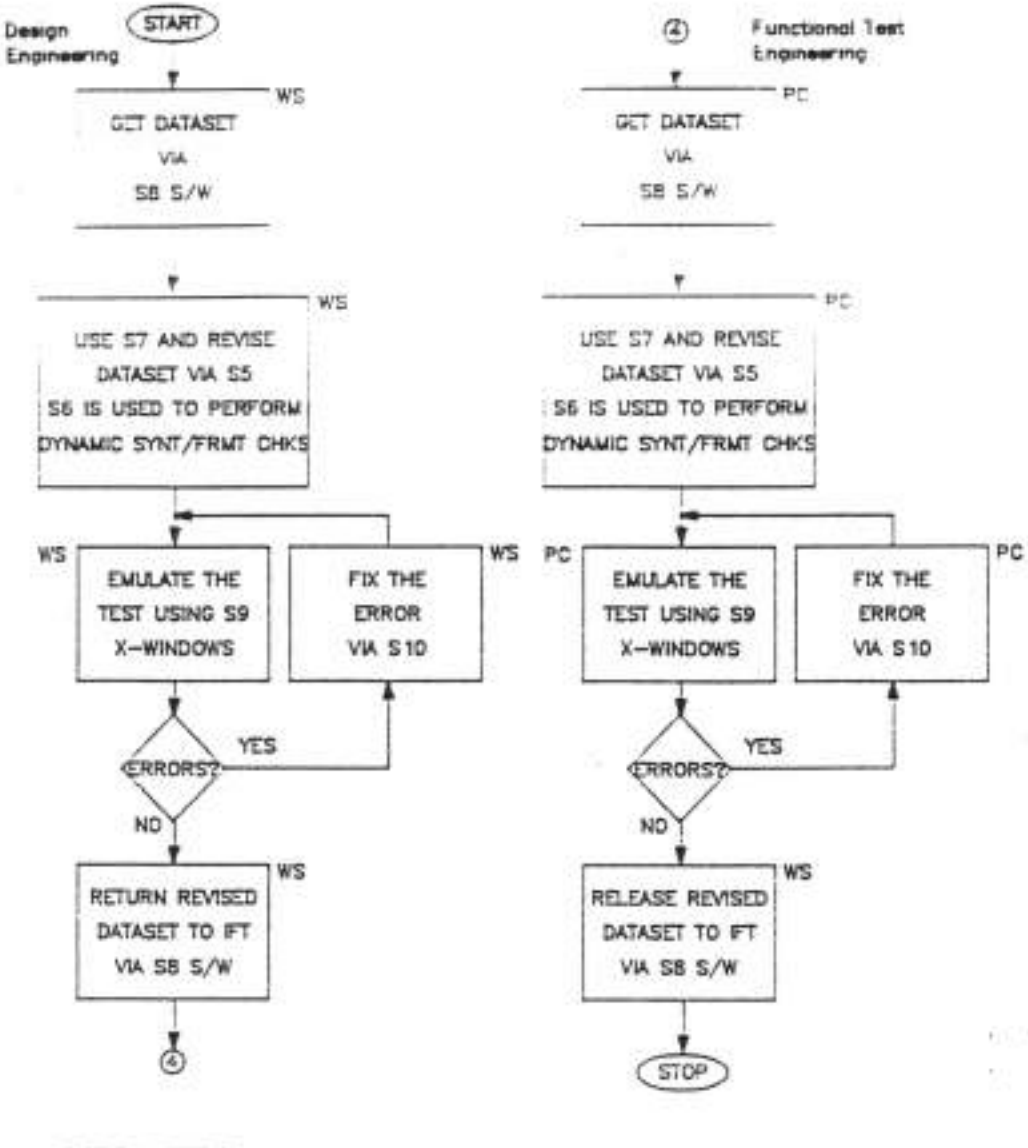


FIGURE 25 PHASE IV DESCRIPTION OF PROCESS FLOW

6. ADVANTAGES AND RISKS

Table 7, "Advantages and Risk factors of Software Tools Matrix," provides the summary of advantages and risk factors of the Software tools.

TABLE 7 ADVANTAGE AND RISK FACTORS OF SOFTWARE TOOLS MATRIX

Phases	Changes	Advantages	Risk
I	Flat File Checker--S1	Basic Need.	Low
	Flat File Revision Utility--S2	Basic Need.	Low
	Special Documentation--S3	Design Engineer edits the dataset from local workstation.	Low
II	X Window Emulation on Remote Hewlett-Packard Computer--S4	Emulate from local personal computer using X Window Local Hewlett-Packard Computers are no longer needed External OID is no longer needed.	Moderate
III	Database Editor (Needed For Phase IV)--S5	Step attribution capability.	Moderate
	Database Checker--S6	Dynamic syntax and format checking.	Low
	Database Revision Utility--S7	Database revision capability.	Low
	Database IFIS Transfer--S8	Faster transfer of the dataset.	Moderate
IV	X Window Emulation On Local Workstation or Personal Computer--S9	Faster emulation (online emulation) Hewlett-Packard Computers are no longer needed Network not used during emulation.	Moderate
	Dynamic Links Between Editor, Emulator, and Simulated OID--S10	Faster Editing.	Low

7. TOOL MATRIX

Table 9, page 69, "SETL Software Tool Environment Matrix by Tool," and table 9, page 70, "SETL Software Tool Environment Matrix by Category," provide cross references between all the required software tools (including existing software tools) and the four phases.

TABLE 8. SETL SOFTWARE ENVIRONMENT MATRIX BY TOOL

SETL Software Tools	Phase I	Phase II	Phase III	Phase IV
Editor	Flat File with MS-Word	(P)	S5	(P)
Flat File Syntax-Format Checker	S1	(P)		
Flat File Revision Utility	S2	(P)		
Flat File IFTS Interface	* Contact Software	* Contact Software		
X Window Emulation Resident on Remote H-P Computer	* Existing 747-400 Emulator	S4	(P)	
Dynamic Syntax-Format Checker with Editor			S6	(P)
Dataset Revision Utility with Custom Editor			S7	(P)
Custom Database-IFTS Interface			S8	(P)
X Window Emulation Resident On Local Workstation				(P)
Dynamic Link Between Editor, Emulator, And Simulated OID				(P)

E--Existing Software P--Same as Previous Phase

S(n)--Software Tool Item Number

TABLE 9 SETL SOFTWARE TOOL ENVIRONMENT MATRIX BY CATEGORY

Phases	Test Revision Environment	File Transfer	Emulation	CID
I	Flat File	Personal Computer Contact Software	Local H F Computer	Hard-Wired
II	Flat File	Personal Computer Contact Software	Remote H F Computer	Simulated
III	Database	Workstation File Transfer Software	Remote H F Computer	Simulated
IV	Database	Workstation File Transfer Software	Local Workstation	Simulated

8. DETAILED SOFTWARE DEVELOPMENT PLAN

8.1 Project Software

8.1.1 Methods

The engineering development methods shall be as defined herein. As the development process evolves and is refined, this plan shall be revised and approved as needed to document those refinements.

Software developers shall use planned releases of each of the software deliverables. The iterative process of coding, releasing, verifying, and validating software is a function of size and complexity. This method will support testing early in the project. This approach will also ease the building of software releases and incorporation of *Problem Reports* (PR).

Functional Test Engineering will oversee preparation of detailed documents describing the processes, controls, and tools that will be applied to tasks such as requirements analysis, coding, debugging, and integration, as applicable. Also identified is the basis for defining and assigning project tasks, resource loading, performing risk management, dealing with software complexity and future software maintenance.

A series of releases of the code shall be planned if the software is above a certain size or complexity. The first release shall contain enough software to represent the most important "core" functions. Subsequent releases shall be built-up around the core, with sub-functions added. The objectives of a series of releases are:

a. To reduce the complexity of any one release to manageable proportions,

b. To speed delivery of the first release so that testing in the target computer and system testing may start at an earlier point. Fundamental deficiencies in these environments may be remedied earlier in the development cycle rather than later.

c. To provide an opportunity to manage the fixing of PRs of differing importance and need. The fixing of PRs of low priority may be postponed to a later planned release, while those of high priority may be fixed in the next or early release.

d. To allow incremental development of related software deliverables (for example, software test procedures) such that the development effort is manageable.

e. The inspection process shall be a submission of the subject item to a peer group for review. The review process is controlled and the defects are formally logged. The defects are analyzed and rectified as appropriate. The inspection process, if properly constituted, can be most productive. Some preparation will be needed to train personnel in the inspection techniques. The inspection process shall be applied to all deliverables.

8.1.2 "Problem Report," PR, System

The problem report system, PR, shall serve as a common system for all problems recorded against this project, throughout all Phases, regardless of whether the root problem affects hardware, software, system requirements, or the like. The PR system shall use database technology and shall be supported by application report writers. Once a piece of software is released for test, changes to that software shall only be made against a PR. Likewise, once a document has been released within the development environment, changes to software documentation shall only be made against a PR. The PR system shall be under the control of a PR Review Board. Details of the PR system may be found in SCMP.

8.1.3 "Software Guidelines," SG

Software Guidelines, SG, shall be developed and shall be approved by the time of the SPDR. These guidelines shall be applicable to the type of software to be developed and within the scope of this project. The guidelines shall be based on existing BCAG standards, but modified accordingly. The objective is to support a unified basis for software development and timely delivery of maintainable quality software.

8.1.4 "Desktop Procedures," DP

Desktop Procedures, DPs, shall be developed. The objective of the DPs will be to describe and standardize steps designed to perform specific day-to-day development activities. An example of one type of activity is the location and naming conventions of daily storage and retrieval of new and revised software. The preliminary version of these procedures

shall be approved and released two weeks before SPDR (see section 8.1.7). The basis of the DPs shall be the project management plans and the SG. It is planned that the DPs shall be updated as a result of project experience.

8.1.5 Training of Development Personnel

Project personnel will be trained in the use of the DP and shall become familiar with the development environment documentation.

8.1.6 "Project Software Functional Requirements," PSFRD

The PSFRD shall describe the functional requirements for each Phase of the project. The document shall be delivered in two increments. The first will provide a preliminary version two weeks prior to the SPDR (see section 8.1.7). The second will provide the completed version two weeks prior to the SCDR. The PSFRD shall contain sufficient information to provide a basis for specification of each individual software piece.

8.1.7 "Software Preliminary Design Review," SPDR

The SPDR shall be held before the start of development of the first piece of software. The objective of this SPDR shall be to ensure viability of the overall software approach.

8.1.8 "Software Prototype Study Report," SPSR

A feasibility study using prototypes shall be undertaken to demonstrate certain software techniques (e.g., window environments, file transfers, database technology) considered to be a project risk. The preliminary results of this study shall be provided two weeks prior to SPDR. Another release shall be provided two weeks before SCDR. The study shall continue throughout the life of the project such that useful data is available for consideration prior to the PDR(n) and the CDR(n) of each individual software piece. The study objectives will be to determine if the techniques represent an unacceptable risk to the project and to give design direction.

8.1.9 "Software Critical Design Review," SCDR

The SCDR shall be held before the start of code implementation. The objective of the SCDR shall be to ensure viability of the overall software design approach.

8.2 Software Elements S(1), S(2), . . . S(n)

The development process for each software piece, S(n). (see section 3.1), will generate the following:

8.2.1 "Software Functional Requirements Document," SFRD(n), for Software S(n)

An SFRD(n), describing the functional requirements to be performed by the particular piece of software, shall be the initial specification document. Approval of the SFRD(n) is essential to the productive development of the software. The SFRD(n) data shall be the basis of the software design and of software verification. The SFRD(n) shall be released and controlled under the processes described in the Software Configuration Management Plan.

The SFRD(n) shall contain a matrix showing how the document complies with the requirements of the PSFRD.

The SFRD(n) shall be subject to review by inspection process prior to approval.

8.2.2 "Software Design Document," SDD(n), for Software S(n)

A SDD(n), describing the design of the particular piece of software, shall be developed from the SFRD(n). The SDD(n) shall be approved and shall be the basis of code implementation and of software verification. The SDD(n) shall be released and controlled under the processes described in the Software Configuration Management Plan.

The SDD(n) shall contain a traceability matrix, showing how all the SFRD(n) functions are covered by the SDD(n), or otherwise accounted for. The matrix shall also account for all SDD(n) requirements.

The SDD(n) shall be subject to review by inspection process prior to approval.

A preliminary version of the SDD(n) shall be available to support the PDR(n) for the software piece in question. The completed version of the SWDD(n) shall be available to support the CDR(n) for the software piece in question.

8.2.3 Software PDR(n) for Software S(n)

A Software PDR(n) shall be held for each piece of software, S(n), before the design of that software

enters the detailed design effort. The review shall determine if the preliminary design will yield the desired product. Deficiencies and remedies shall be noted.

8.2.4 Software CDR(n) for Software S(n)

A Software CDR(n) shall be held for each piece of software, S(n), before commencement of the implementation in code of the detailed design. The Review shall determine if the completed design will yield the desired product. Deficiencies shall be noted and remedies applied.

8.2.5 "Software Verification Test Procedures," SVTP(n), for Software S(n)

The SVTP(n) shall be started in accordance with the Software Verification Plan. The Procedures shall be ready to support the developed code. The SVTP(n) development shall be an on-going effort to keep pace with the emerging software. Those portions of the SVTP(n) needed to support the software-to-software integration shall be developed first, followed by that portion needed to support hardware-to-software integration, then system integration.

The Procedures shall contain commentary explaining the purpose for each test and its expected results.

Intermediate versions and the composite version of the SVTP(n) shall be released and controlled under the processes described in the SCMP. The SVTP(n) shall contain a traceability matrix, showing how all the SFRD(n) functions and SDD(n) requirements are covered by the composite SVTP(n), or otherwise accounted for.

8.2.6 Code Implementation for Software S(n)

The SDD(n) shall be implemented by coding in the chosen computer language for that piece of software. The coding shall be in accordance with the agreed PSG and the specific computer language standards noted therein.

Each software shall be subject to inspection after debug. After inspection defects have been remedied, the module shall be released and controlled under the processes described in the Software Configuration Management Plan.

8.2.7 "Software-to-Software Integration Test," SSIT, of Software S(n)

The software modules shall be verified individually and in combination, one-to-another, according to the SVP. The verification environment shall be as specified in the SVP. The environment may be a simulated or emulated target computer residing in a host computer. The results of the verification shall be recorded in the SETL Tool Development Software Verification Report, SVTR(n), for that particular piece of software. All defects found shall be recorded in the PR system and handled accordingly.

8.2.8 "Hardware-to-Software Integration Test," HSIT, of Software S(n)

The integrated software shall be verified on the actual computer. The verification environment shall be as specified in the SVP. The results of the verification shall be recorded in the SVTR(n). All defects found shall be recorded in the PR system and handled accordingly.

8.2.9 System Integration of Software S(n)

The integrated hardware and software shall be verified on the actual system. The verification environment shall be as specified in the SVP. The results of the verification shall be recorded in the SVTR(n). All defects found shall be recorded in the PR system and handled accordingly.

8.2.10 "Training and User Guide," TUG(n), for Software S(n)

A TUG shall be developed, approved, and released for each piece of software. The document shall describe how the software is to be used in an operational environment. The document shall contain sufficient material to support training of end-user personnel.

8.2.11 "Version Description Document," VDD(n), of Software S(n)

A VDD is a configuration index of all the source modules by version and the processes used to make the executable for that particular piece of software. The name and version of the specification, design, verification and validation, and all support documents are listed for that version of the released software. The VDD(n) shall be approved, released, and controlled under the processes described in the Software Configuration Management Plan.

The VDD(n) shall be subject to review by inspection process prior to approval.

8.3 Management Reviews and Status Reporting

8.3.1 Reviews

Shall be conducted on a periodic basis and at key milestones, as described in these managements plans (see section 3.2) and schedules, to provide visibility to development progress, schedule progress and cost performance. Scheduling of reviews shall support the opportunity for corrective action.

8.3.2 Status

The status reporting process will take place on a mutually agreeable regular basis to provide visibility to management on project status. The report content shall be standardized by mutual agreement with the customer. The schedule included in this plan shall be updated as part of the status report. The status report shall include trend data and visibility of progress to plan. The report shall include PR status. The reports shall include forecast of project effort and expected completion dates.

REFERENCES

1. *AECMA Simplified English: A Guide for the Presentation of Aircraft Maintenance Documentation in the International Aerospace Maintenance Language*, AECMA Document PSC-85-16598, Change 5. Paris, France: Association Europeenne des Constructeurs de Materiel Aerospatial. 1 December 1989.

2. *777 ATE Test Procedures Language Requirements*, D532W001, Rev. New. Boeing Commercial Airplane Group. Seattle: The Boeing Company. (to be released, Spring 1991)

3. X Window Ver. 11. Hewlett-Packard Company, Corvallis, Oregon.

4. *Integrated Functional Test Data Management System (IFTDMS) Engineering Operator's Manual*, D935U002. Boeing Commercial Airplane Group. Seattle: The Boeing Company. February 1990.

5. *BCAG Avionics Computer Software Technical Standard*, D6-35071-1, Rev. D. Boeing Commercial Airplane Group. Seattle: The Boeing Company. July 1990.

BIBLIOGRAPHY

Functional Test Dataset Release System Design Document for the 747-400, D280U105-200. Boeing Commercial Airplane Group. Seattle: The Boeing Company. May 1990.

Integrated Functional Test System Analysis Document, D6-55801. Boeing Commercial Airplane Group. Seattle: The Boeing Company. February 1991.

"Embedded Software Standards," Rev. F. BSWS-1000. *Boeing Software Standards*. D-6000. Seattle: The Boeing Company. 10 October 1989.

"Embedded Software Guidelines for Cost Management," Rev. E. BSWS-1007. *Boeing Software Standards*. D-6000. Seattle: The Boeing Company. 16 October 1990.

"Acceptance/Functional Test Procedure Drawings," Rev. F. BDS-1260-2. *Boeing Drafting Standards*. BCA Series. Seattle: The Boeing Company. 21 December 1989.

"Drawing Dataset Preparation," Rev. C. BDS-1510-1. *Boeing Drafting Standards*. BCA Series. Seattle: The Boeing Company. 17 December 1990.

"Book-Form Drawings," Rev. D., BDS-1120-3. Boeing Drafting Standards. BCA Series. Seattle: The Boeing Company. 24 September 1989.

"Preparation of Documents," Rev. A. Boeing Drafting Standards. BCA Series. Seattle: The Boeing Company. 27 April 1990.