SOFTWARE ENGINEERING LABORATORY (SEL) DATA BASE ORGANIZATION AND USER'S GUIDE REVISION 1

MARCH 1983

NASA
National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

SOFTWARE ENGINEERING LABORATORY (SEL) DATA BASE ORGANIZATION AND USER'S GUIDE REVISION 1

MARCH 1983

NASA

Goddard Space Flight Center

FOREWORD

The Software Engineering Laboratory (SEL) is an organization sponsored by the National Aeronautics and Space Administration Goddard Space Flight Center (NASA/GSFC) and created for the purpose of investigating the effectiveness of software engineering technologies when applied to the development of applications software. The SEL was created in 1977 and has three primary organizational members:

NASA/GSFC (Systems Development and Analysis Branch)
The University of Maryland (Computer Sciences Department)
Computer Sciences Corporation (PCASS Project)

The goals of the SEL are (1) to understand the software development process in the GSFC environment; (2) to measure the effect of various methodologies, tools, and models on this process; and (3) to identify and then to apply successful development practices. The activities, findings, and recommendations of the SEL are recorded in the Software Engineering Laboratory Series, a continuing series of reports that includes this document. A version of this document was also issued as Computer Sciences Corporation document CSC/SD-83/6012.

The primary contributors to this document include

Pei-Shen Lo (Computer Sciences Corporation)
David Wyckoff (Computer Sciences Corporation)

Other contributors include

Jerry Page (Computer Sciences Corporation)
Frank McGarry (Goddard Space Flight Center)

Single copies of this document can be obtained by writing to

Frank E. McGarry
Code 582.1
NASA/GSFC
Greenbelt, Md. 20771

ABSTRACT

This document provides a description of the structure of the Software Engineering Laboratory (SEL) data base. It defines each data base file in detail and provides information about how to access and use the data for programmers and other users. Several data base reporting programs are described also.

TABLE OF CONTENTS

TABLE OF CONTENTS (Cont'd)

TABLE OF CONTENTS (Cont'd)

SECTION 1 - INTRODUCTION

The Software Engineering Laboratory (SEL) was created to support efforts to measure and evaluate the effects of various methodologies, models, and tools on the software development process. The SEL is a combined effort involving Goddard Space Flight Center (GSFC), Computer Sciences Corporation (CSC), and the University of Maryland (UM).

One of the major functions of the SEL is the collection, analysis, and archiving of detailed data, describing all facets of the software development process within the Systems Development Section of GSFC. The projects providing the detailed data are software development efforts in support of GSFC flight dynamics ground support systems.

To facilitate the use of the information collected, a data base was designed that consists of approximately 330 indexed files on a DEC PDP-11/70 computer. In addition to several header or summary files, each project studied may require up to 11 files--one for each of the 7 types of forms collected and 4 general information files. Section 2 of this document describes the structure of the data base. The software packages that support the entry, maintenance, reporting, retrieving, and backup of this data base are described in Section 3.

SECTION 2 - DATA BASE ORGANIZATION

This section describes the structure and content of the SEL data base files. Many of the files are organized in response to the structure of the SEL forms. In general, the files are organized by project and by form type. Exceptions and additions are noted in the following subsections.

The following is a list of the data base files in the order in which they are described in Section 2.2 and in Appendix A ("proj" is the project name; "n" is the number of projects in the data base):

Descriptive File Name	Number of Files	File Name
Encoding Dictionary File	1	ENCODE.HDR
Estimated Statistics File	1	EST.HDR
File Name and Status File	1	STAT.HDR
Phase Dates File	1	HEADER.HDR
Subjective Evaluations Directory File	1	DIR.HDR
Subjective Evaluations File	1	SEF.HDR
Attitude Maintenance Change Report (ATM) File	n	proj1.ATM,proj2.ATM, ...,projn.ATM
Change Report Form (CRF) File	n	proj1.CRF,proj2.CRF, ...,projn.CRF
Component Status Report (CSR) File	n	proj1.CSR,proj2.CSR, ...,projn.CSR
Component Summary Form (CSF) File	n	proj1.CSF, proj2.CSF, ...,projn.CSF
General Project Summary (GPS) File	n	proj1.GPS,proj2.GPS, ...,projn.GPS
Resource Summary Form (RSF) File	n	proj1.RSF,proj2.RSF, ...,projn.RSF

Descriptive File Name	Number of Files	File Name
Run Analysis Form (RAF) File	n	proj1.RAF,proj2.RAF, ...,projn.RAF
Accounting Information (ACC) File	n	proj1.ACC,proj2.ACC, ...,projn.ACC
Comment (CMT) File	n	proj1.CMT,proj2.CMT, ...,projn.CMT
Component Information File (CIF)	n	proj1.CIF,proj2.CIF, ...,projn.CIF
Growth History (HIS) File	n	proj1.HIS,proj2.HIS, ...,projn.HIS
Source Analyzer Program (SAP) Output File	1	ALL.SAP
Transaction (backup) Files	7	TRANS.CIF, TRANS.CRF, TRANS.CSR, TRANS.CSF, TRANS.HIS, TRANS.RSF, TRANS.RAF

2.1 FILE ATTRIBUTES

All data base files, except the transaction files, are located on disk DB1, under user identification code (UIC) [204,1]. The name of a file is composed by attaching the project name and form type abbreviation to the disk and UIC designation. For example, to access change report data for project PROJ, the name would be DB1:[204,1]PROJ.CRF.

The transaction files are on disk DB0 to allow data base restoration in the case of a failure of disk DB1.

The larger projects have up to 2,000 forms and up to 10,000 records. A project this large would take up about 4000 500-byte blocks. The total data base takes up approximately 49,000 blocks.

2.2 FILE DESCRIPTIONS

On this data base, there are three file types or categories:

- Header or summary files
- Form data files
- Auxiliary files

Header or summary files contain directory and summary information (such as total lines of source code, project duration, and total effort) for each project.

Form data files correspond directly to a particular type of form; there is a separate file for each form type per project.

Auxiliary files contain support information, such as descriptive text (Comment Files), taken from the software engineering forms and component descriptions generated by SAP (Reference 1).

Except as noted, all files are indexed. Appendix A describes all file formats in detail, including every field in each record type.

2.2.1 HEADER (SUMMARY) FILES

The header or summary files contain directory and summary information for the entire data base. These files can be used to obtain top-level summary reports on all the data. The following six header files are described in this section:

- Encoding Dictionary
- Estimated Statistics
- File Name and Status
- Phase Dates
- Subjective Evaluations
- Subjective Evaluations Directory

2.2.1.1 Encoding Dictionary File (ENCODE.HDR)

The Encoding Dictionary File contains the numerical code type information used to represent the lengthier alpha-numeric or English text information. The codes are used to save space where certain titles, names, or other pieces of information are used repetitively throughout the data base. Twenty-four different types of codes are represented on the file. (Some types may require more than one record of data.) Typical pieces of data that are coded and placed on this dictionary are project name, programmer name, source language, types of changes, and types of error. Thus, since project names, for example, are used repetitively throughout the data base, the corresponding numerical codes are used instead of the full name. The codes are assigned by the data base administrator and do not have any particular significance as far as priority or importance are concerned. The Encoding Dictionary File contains the following fields:

- Code type
- Code
- Abbreviated name
- Full English description

Below are three sample records:[1]

Type	Code	Abbreviation	Description
4	1	UNITT	Unit test
4	2	SYSTEMT	System test
4	3	BNCHMRKT	Benchmark test

See Appendix A, Section A.1, for the file format.

[1]Throughout Section 2, each of the records is to be read across. For example, on page 2-7, the first or Project 1 record contains a code of 10, 638 components, 535 modules, and so on.

2.2.1.2 Estimated Statistics File (EST.HDR)

The Estimated Statistics File characterizes the size and resources (manpower, computer) of each project. The file contains a single record of information for each project on the data base. Each record contains a project name as well as information summarizing the characteristics of that project. This information is usually collected at the conclusion of a project by personnel close to the project. It summarizes the basic size and resource characteristics of each project. The project managers review the completed project and gather the following information for this file:

- Project name
- Number of components and modules
- Number of lines, executable statements, runs, and changes
- Number of pages of documentation
- Programmer, management, and services hours
- IBM S/360-95 and -75 hours (based on computer accounting information)
- Other computer hours

Below are three sample records--one for Project 1, one for Project 2, and one for Project 3--in the Estimated Statistics File. The programmer, management, and services hours are stored as integer type characters formed from the real number values times 10. The status flag[1] refers to the status of the data: 1 is unchecked data, 2 is hand-checked data, and 3 is data verified by application.

[1]This is true for all sample records throughout Section 2.

Project Name	Project Code	Number of Components	Total Number of Modules	Number of New Modules
PROJ1	10	638	535	337
PROJ2	38	113	102	93
PROJ3	19	639	519	418

Number of Modified Modules	Number of Runs	Number of Changes	Pages of Document	Total Number of Lines
31	7500	1576	1793	75,393
0	1589	255	763	15,258
59	1000	2350	2458	85,369

Number of New Lines	Number of Modified Lines	Number of Total Exec Statements	Number of New Exec Statements	Number of Modified Exec Statements
49,316	4252	30,448	29,098	1179
14,873	0	4,482	4,413	0
76,883	5652	38,157	35,203	2161

Programmer Hours	Management Hours	Services Hours	S/360-95 Computer Hours
109,565	35,510	12,310	2090
31,638	13,022	11,942	628
116,586	27,119	27,444	3120

S/360-75 Computer Hours	Other Computer Hours	Status Flag	Active Flag	Project Category
1930	0	1	N	1
4	0	1	N	1
1852	0	1	N	1

See Appendix A, Section A.2, for the file format.

2.2.1.3 File Name and Status File (STAT.HDR)

The File Name and Status File is a type of summary directory for the entire data base. It contains one record for each indexed file in the data base. Each record contains a file name; creation, last backup, and last access dates (YYMMDD format); and number of records in the particular file. These data are updated automatically by the data entry program (Data Base Maintenance Software (DBAM) (Reference 2)) whenever a file is accessed.

Three sample File Name and Status File records are given below--one for Project 1, one for Project 2, and one for Project 3:

Project Name	Project Code	File Name	Creation Date	Last Backup Date	Last Update Date	Number of Records
PROJ1	10	DB1:[204,1]PROJ1.RSF	790312	820611	790312	91
PROJ2	38	DB1:[204,1]PROJ2.RSF	791026	820611	0	93
PROJ3	19	DB1:[204,1]PROJ3.RSF	790901	820611	0	162

See Appendix A, Section A.3, for the file format.

2.2.1.4 Phase Dates File (HEADER.HDR)

The Phase Dates File contains the start and end dates for all phases in the software development cycle. The file includes project name, code, and the dates (YYMMDD format) for the requirements, design, code and unit test, system test, acceptance test, cleanup, and maintenance phases for each project. These dates are obtained from the project manager at the conclusion of each project.

Below are three sample Phase Dates File records--one for Project 4, one for Project 2, and one for Project 3:

Project Name	Project Code	Development Computer	Target Computer	Alien Computer Use
PROJ4	2	0	0	0
PROJ2	10	0	0	0
PROJ3	19	0	0	0

Req. Start	Req. End	Design Start	Design End	Code and Test Start
761010	770213	770213	770604	770604
770101	770401	770401	770730	770730
780101	780501	780501	781014	781014

Code and Test End	System Test Start	System Test End	Acceptance Test Start	Acceptance Test End
771203	771203	780204	780204	780318
780114	780114	780218	780218	780415
790331	790331	790602	790602	791013

Cleanup Start	Cleanup End	Maintenance Start	Maintenance End	Status Flag
780318	780427	780429	780820	1
780415	780624	780624	781024	1
791013	791222	791222	800404	1

See Appendix A, Section A.4, for the file format.

2.2.1.5 Subjective Evaluations File (SEF.HDR)

The Subjective Evaluations File characterizes the development methods and environment of each project. New information is added to this file near the conclusion of each project. By reviewing code and documents and by observing the development process, project managers quantify the degree to which each of the qualities applies to the project. This is strictly a subjective management evaluation.

The data for each project are contained in seven variable-length records. Each record represents a main category of measures. The seven categories are

- Software Engineering (SE)--Includes practices and techniques (MT), tools (TS), and documentation (DC) measures
- Development Team Ability (AB)--Includes experience with application (AP), effectiveness of management (MG), and performance of team (PF) measures
- Difficulty of Project (DF)--Includes complexity of problem (CP), internal influences on project (IN), and external influences on project (EX) measures
- Process and Product Characteristics (PC)--Includes resources available (RA), software product (PR), and product/process performance (PP) measures
- Development Team Background (DB)--Includes team rank (RK), years of professional experience (YP), years of applicable experience (YA), and years of environment experience (YE) measures
- Models (MD)--Includes Walston-Felix model (WF), PRICE S3 model (PS), and COCOMO model (CO) measures
- Additional Details (AD)--Includes miscellaneous (MS) and code breakdown (SW) measures

See Appendix A, Section A.5, for the file format. Sample records are not presented here because of their extreme length. Reference 3 describes the data collected for this file.

2.2.1.6 Subjective Evaluations Directory File (DIR.HDR)

The Subjective Evaluations Directory File contains the alphanumeric code type information used to represent the lengthier English text information. The codes represent

certain titles, names, or other pieces of information describing measures used in the Subjective Evaluations File. Each record contains information for one specified measure in the Subjective Evaluations File. The Subjective Evaluations Directory File contains the following fields:

- Code for the measure
- Name of the measure
- Minimum value of the measure
- Maximum value of the measure
- Data record sequence number (1 through 7)
- Byte location in the data record
- Textual description of the measure

The following are three sample records:

Code	Name	Minimum Value	Maximum Value	Record Number	Byte Location	Description
AP01	EXPERT1	0	50	2	6	Expert 1
MT20	CCONFIG	0	50	1	44	Code (configuration control)
SW61	SCHANGEN	0	9000	7	472	Software Changes (new)

See Appendix A, Section A.6, for the file format.

2.2.2 FORM DATA FILES

These files correspond in number and in content to the information collected on the software engineering forms. There is one file for each form type per project. There are seven form data files, which are described in detail in the following subsections:

1. Attitude Maintenance Change Report (ATM) File
2. Change Report Form (CRF) File
3. Component Status Report (CSR) File
4. Component Summary Form (CSF) File

5. General Project Summary (GPS) File
6. Resource Summary Form (RSF) File
7. Run Analysis Form (RAF) File

2.2.2.1 Attitude Maintenance Change Report (ATM) File

The Attitude Maintenance Change Report File contains information on changes made to a program during the maintenance and operation phase of the project (after the project delivery date). The ATM form is filled out by maintenance personnel. Although this file contains essentially the same information provided on the Change Report Form, there are some slight differences. The following information can be found on the ATM File:

- Programmer
- Number of components changed
- Date on which change was determined
- Date on which change was started
- Type of change
- Primary error type
- Types of error detection activities
- Time spent implementing change

See Appendix A, Section A.7, for the file format.

2.2.2.2 Change Report Form (CRF) File

The Change Report Form File contains information on changes made by a programmer after the source has been added to the permanent library. The CRF is filled out by the programmer. Each form describes one error or change. One record on the CRF File represents one form and contains the following:

- Programmer
- Form date
- Number of components changed
- Number of components examined

- Date on which change was determined
- Date on which change started
- Amount of time/effort required for change
- Type of change
- Type of error (if error)
- When error entered system
- Activities used to isolate error
- Time required to isolate error
- Whether or not a workaround was used
- Whether or not change was related to a previous change

Below are three sample records on the CRF File. Hyphens indicate blanks.

Form Number	Project	Programmer	Form Date	Number of Components Changed
K00016	19	26543	790103	9
K00017	19	14336	781026	1
K00018	19	14336	781026	1

Number of Components Examined	More Than One Comp Affected	Date Change Was Determined	Date Change Was Started	Effort for Change
11	Y	790102	790102	2
1	-	781026	781026	2
1	-	781026	781026	1

Type of Change 1 2 3 4	Changed Components 1	2	3	4	5	Type of Error 1 2 3 4	When Error Entered System
1 4 - -	443	386	907	881	252	3 - - -	3
1 - - -	695	-	-	-	-	7 - - -	-
1 - - -	152	-	-	-	-	7 8 - -	4

Data Structure Error	Control Logic Error
X	-
-	-
-	X

The following fields describe activities used to isolate errors.

For Program Validation 1 2 3 4 5	For Detecting Symptoms 1 2 3 4 5	Tried in Finding Cause 1 2 3 4 5	For Finding Cause 1 2 3 4 5
1 4 - - -	1 4 - - -	5 6 B - -	5 6 B - -
1 5 - - -	1 - - - -	- - - - -	- - - - -
1 - - - -	5 - - - -	- - - - -	- - - - -

Time To Isolate Error	Workaround Used	Related to Previous Change	Previous Form Number	Previous Form Date
1	X	Y	00002	780831
1	-	N	-	-
1	-	N	-	-

Reason Comment Flag	Description Comment Flag	General Comment Flag	Status Flag
Y	Y	Y	1
Y	Y	N	1
Y	Y	N	1

See Appendix A, Section A.8, for the file format.

2.2.2.3 Component Status Report (CSR) File

The Component Status Report File contains data on the amount of time spent by a programmer on different activities and components (modules) in the development process. The time spent on components is divided into design, code, and test stages. One record on the CSR File represents one line on

the CSR form; a form may spread over several records. A record (line) contains the following:

- Programmer
- Form date
- Component
- Hours spent in each phase
- Other activity (name)
- Other activity (hours spent)

The CSR form is filled out by the programmer once a week.

Below are three sample records in the CSR File. The hours shown represent real numbers even though they are shown as integers. The correct real number value is obtained by dividing the given number by 10. Hyphens represent blanks.

Form Number	Sequence Number	Project	Programmer	Form Date
B03146	1	36	22137	791012
B03146	2	36	22137	791012
B03146	3	36	22137	791012

Component	Design Create Hours	Design Read Hours	Design Review Hours	Code Hours
451	100	50	0	0
50	0	0	0	0
-	-	-	-	-

Code Read Hours	Code Review Hours	Unit Test Hours	Integration Test Hours	Review Test Hours
0	0	0	0	0
0	0	0	0	0
-	-	-	-	-

Other Activity Name	Other Activity Hours	Status Flag	Phase Flag
-	-	1	D
-	-	1	D
TRAVEL	5	1	D

See Appendix A, Section A.9, for the file format.

2.2.2.4 Component Summary Form (CSF) File

The Component Summary Form File contains a general description of a component. This form is filled out by the programmer twice: when a component is first defined, it is filled out with estimates of the effort and size of the component; when the component is completed, it is filled out with the actual values. There should be two forms for each component at the completion of a project. One record in the CSF File represents one form. Each record contains the following information:

- Programmer
- Form date
- Form stage
- Component
- Precision of specification
- Complexity
- Type of software
- Type of statements
- Number of statements
- Relation to other software
- Type of addition (if addition)
- Number of components called, shared, and descendent
- Languages used
- Form of specification
- Constraints (yes/no)
- Number of design, code, and test runs

- Design, code, and test computer time used
- Time/effort spent in design, code, and test
- Design, code, and test end dates

Below are three sample records on the CSF File. Hyphens represent blanks.

Form Number	Project	Programmer Filling Out Form	Programmer Implementing Component	Form Date
101878	36	2	2	800606
101879	36	2	2	800617
101880	36	3	3	800709

Form Stage	Component	Precision of Specification	Complexity	Type of Software 1 2 3
N	459	3	E	3 - -
N	456	1	E	5 - -
C	462	3	M	1 - -

Percent of Assignment Statements	Percent of Control Statements	Percent of Other Statements	Lines Without Comments	Lines With Comments
20	50	30	50	100
0	0	100	7	25
60	10	10	55	70

Number of Machine Bytes	Independent of Other Software	Relation to Other Software	Type of Addition 1 2 3 4	Number of Components Called
-	N	1	1 4 - -	2
-	Y	-	- - - -	0
400	Y	-	- - - -	0

Number Calling This Comp	Number of Shared Components	Number of Components Descending	Primary Language	Percent Primary Language
1	0	3	1	100
0	1	0	1	100
1	0	3	1	100

Secondary Language	Percent Secondary Language	Functional Design 1	Functional Design 2	Procedural Design 1	Procedural Design 2	English Design 1	English Design 2
-	-	1	-	-	-	-	-
-	-	4	-	-	-	-	-
-	-	1	-	-		-	-

Formal Design 1	Formal Design 2	Other Design 1	Other Design 2	Memory Constraint Yes	Memory Constraint Ok	Exec Time Constraint Yes	Exec Time Constraint Ok	Other Constraint Yes	Other Constraint Ok
-	-	-	-	X	X	-	-	-	-
-	-	-	-	-	-	-	-	-	-
-	-	-	-	-	-	X	-	-	-

Design Runs	Code Runs	Test Runs	Design Computer Time	Code Computer Time
0	2	4	0	10
0	2	3	0	5
0	2	6	0	10

Test Computer Time	Design Effort	Code Effort	Test Effort	Estimated Design End Date
100	80	70	120	800711
20	40	30	70	800502
120	60	60	140	800703

Estimated Code End Date	Estimated Test End Date	Description Comment Flag	Components Called 1	2	3	4	5
800711	800905	Y	94	92	-	-	-
800711	801231	Y	-	-	-	-	-
800703	800711	Y	-	-	-	-	-

Calling Components 1	2	3	4	5	Shared Components 1	2	3	4	5
83	-	-	-	-	-	-	-	-	-
-	-	-	-	-	451	-	-	-	-
84	-	-	-	-	-	-	-	-	-

Components Affected by Reorganization 1	2	3	4	5	Form of Design Other Name
-	-	-	-	-	DUMMY-OTHER-NAME-ABC
-	-	-	-	-	-
-	-	-	-	-	-

Constraint Other Name	Useful Items Comment	Additional Comment Flag	Status Flag
DUMMY-NAME-ABCDEFGHI	Y	N	1
-	Y	Y	1
-	N	N	1

See Appendix A, Section A.10, for the file format.

2.2.2.5 General Project Summary (GPS) File

The General Project Summary File contains a summary of resources, times, program sizes, costs, and several other aspects of a project. The GPS form is filled out by the project manager or project leader at the beginning and end of the project and at the end of major phases. The following information is contained on the GPS File:

- Project description
- Resources used
- Scheduling
- Cost of project
- Size of project
- Computer access
- Techniques employed
- Formalisms used
- Automated tools used
- Type of project organization
- Standards used
- Milestones reached
- Documentation issued

- Problems encountered
- Quality assurance employed

See Appendix A, Section A.11, for the file format.

2.2.2.6 Resource Summary Form (RSF) File

The Resource Summary Form File contains information on programmer time, computer time and runs, and other service charges. The resources are recorded by the project manager for each week for up to 11 weeks on a single form. Each record in the file contains information from one line of the RSF File, either manpower, computer, or services data. A record contains the following information:

- Resource type indicator (manpower, computer, or services)
- Resource code
- Form date
- Percentage management
- Beginning date of data
- Hours each week (up to 11 weeks)
- Number of computer runs each week (up to 11 weeks)

Below are three sample RSF File records. The resource hours are integers representing real number values times 10. Hyphens indicate blanks. The number sign (#) indicates the week (1 through 11).

Form Number	Sequence Number	Project	Resource Type	Resource
C00144	1	36	M	18024
C00144	2	36	M	22137
C00144	3	36	C	1

Form Date	Percent Management	Beginning Date Of Data
791214	100	791005
791214	10	791005
791214	-	791005

Runs #1	Resource Hours #1	Runs #2	Resource Hours #2	Runs #3	Resource Hours #3
0	100	0	100	0	100
-	-	0	240	0	400
0	240	-	-	5	10

Runs #4	Resource Hours #4	Runs #5	Resource Hours #5	Runs #6	Resource Hours #6
0	100	0	100	0	320
0	100	0	100	0	100
0	0	4	80	0	0

Runs #7	Resource Hours #7	Runs #8	Resource Hours #8	Runs #9	Resource Hours #9
0	100	-	-	0	100
0	385	0	240	0	400
0	0	0	0	0	0

Runs #10	Resource Hours #10	Runs #11	Resource Hours #11	Status Flag	Phase Flag
0	400	-	-	1	D
-	-	-	-	1	D
0	0	0	0	1	D

See Appendix A, Section A.12, for the file format.

2.2.2.7 Run Analysis Form (RAF) File

The Run Analysis Form File contains information about computer runs made by a programmer on a project. The RAF,

filled out by the programmer, has data from up to nine separate runs. One record represents one line (run) on the RAF. The following information is contained on the RAF File:

- Programmer
- Run date
- Computer model used
- Interactive run indicator
- Purpose of run (unit test, maintenance)
- Number and type of components
- Whether or not first run
- Whether or not run met objectives
- Run results

Below are three sample records on the RAF File. Hyphens indicate blanks.

Form Number	Sequence Number	Project	Programmer	Run Date
J01946	1	42	22137	791025
J01946	2	42	22137	791025
J01946	3	42	22137	791026

Computer	Interactive Run Indicator	Run Purpose 1 2 3 4	Number Of Components
6	X	4 7 - -	2
6	-	7 - - -	1
3	-	7 - - -	1

Components 1	2	3	4	5	First Run Indicator	Run Met Objectives Indicator
280	4	-	-	-	X	Y
4	-	-	-	-	-	-
10	-	-	-	-	-	-

Run Result 1 2 3 4	Comment Indicator	Status Flag
1 4 - -	N	1
4 - - -	N	1
4 - - -	N	1

See Appendix A, Section A.13, for the file format.

2.2.3 AUXILIARY FILES

This subsection describes the remaining six file types, which are identified as auxiliary files:

1. Accounting Information (ACC) File
2. Comment (CMT) File
3. Component Information File (CIF)
4. Growth History (HIS) File
5. Source Analyzer Program (SAP) Output File
6. Transaction Files

2.2.3.1 Accounting Information (ACC) File

The Accounting Information File contains accounting information for jobs run on the IBM S/360-95 and -75. Each record contains information relating to a specific 4-hour block of time (i.e., 1 day's activities on a computer are represented by six records). A record contains the following information:

- Date
- Start time of 4-hour period
- CPU and I/O time for the IBM S/360-95 and -75
- Number of runs for the IBM S/360-95 and -75
- Number of remote job entry (RJE) jobs
- Number of card reader jobs

This information is obtained from an accounting history tape on the IBM S/360, which is generated from an online accounting system that monitors all activity on the particular machine.

See Appendix A, Section A.14, for the file format.

2.2.3.2 Comment (CMT) File

The Comment File contains all comments from the Change Report Form, the Component Summary Form, and the Run Analysis Form for a given project. (The component status report and resource summary forms do not have comment fields.) Each record on the CMT File contains a comment and the number of the originating form. This file is automatically updated by DBAM whenever one of the form types with comments is processed. This information is stored separately, since it was felt that most users of the form data files would generally not want the comment information. Therefore, the form data files were made smaller by deleting this text information.

Below are three sample records on the CMT File:

Form Number	Sequence Number	Comment Type	Record Number	Project
101878	1	D	1	36
101879	1	D	1	36
101880	1	U	1	36

Continuation Indicator	Text	Status Flag
N	FILL PREREAD ARRAYS	1
N	DRIVER FOR READING TELEMETRY RECORDS	1
N	CHECKS BUFFER SIZE	1

See Appendix A, Section A.15, for the file format.

2.2.3.3 Component Information File (CIF)

The Component Information File was developed to characterize each component. This file contains several source code statistics for each component. Some of the items are general library information, such as how many changes were made to a component. The rest are statistics extracted from the

FORTRAN source code of the component by SAP. Each CIF record contains the following information:

- Component name and code
- PANVALET level number (number of source changes)
- Module and subsystem function
- Whether component is new, old, or modified
- Number of executable statements
- Number of lines with comments
- Number of comment lines
- Number of unique operators
- Number of unique operands
- Total number of operators
- Total number of operands
- Number of input and output variables from module
- Number of decisions
- Number of FUNCTION references
- Number of I/O statements
- Number of assignment statements
- Number of CALL statements
- Number of FORMAT statements

Note that operands and operators are software measures described by Halstead in Reference 4.

There is a unique correspondence between the component name and component code listed above that serves as a dictionary for all component codes used in other data base files for a particular project.

Below are three sample records on the CIF. Hyphens indicate blanks.

Project	Component Name	Component Code	PANVALET Level Number	Module Function
19	ACBIAS	275	4	-
19	ACBIASM	350	-	-
19	ACBIASUN	351	-	-

Subsystem Function	Origin	Executable Statements	Source Lines	Comments
-	-	90	254	102
-	1	31	104	44
-	1	17	89	43

Operators	Operands	Total Operators	Total Operands	Input and Output Variables
24	64	421	315	29
9	25	158	155	9
9	19	70	67	10

Decisions	FUNCTION References	I/O Statements	Assignment Statements	CALL Statements
21	29	1	50	18
2	0	1	27	0
2	0	1	13	0

FORMAT Statements	Status Flag
2	1
2	1
2	1

See Appendix A, Section A.16, for the file format.

2.2.3.4 Growth History (HIS) File

The Growth History File contains information about the changing number of modules and lines of code for each project. Each record contains a date and the total number of source code lines, modules, and changes up to that date. This information comes from weekly listings of the PANVALET library directory for projects using the IBM computers and from weekly file directory listings for projects using the DEC computers.

Below are three sample records from the HIS File.

Project	Date	Source Lines to Date	Modules to Date	Changes to Date	Status Flag
10	770923	12414	143	12	1
10	770930	12414	143	12	1
10	771007	15973	172	56	1

See Appendix A, Section A.17, for the file format.

2.2.3.5 SAP Output File

The SAP Output File is a single intermediate sequential file containing several source code statistics produced by SAP. Each record in this file contains information on individual components, such as the number of executable statements and the number of assignment statements. The record format is similar to that of the CIF but not identical. Some rearrangement is made before DBAM moves the data into the appropriate CIF.

Below are three sample records from the SAP Output File.

Project Name	Module Name	Parameters Passed In	Comment Lines	Executable Statements
PROJ2	ACDUMFL1	28	105	93
PROJ2	TPTPCHEK	3	63	12
PROJ5	DAINRT	6	32	1

I/O Statements	Source Lines	Operators	Operands	Total Operators
1	252	12	105	316
1	82	7	8	27
0	39	0	0	0

Total Operands	Number Of IF and .IF Statements	Decisions	Input and Output Var. to Module	COMMON Variables
280	1	4	57	6
12	5	6	3	0
0	0	0	6	0

DO and DOWHILE Statements	FUNCTION References	Structured Statements	Parameters Passed Out	Assignment Statements
3	0	0	73	64
0	0	8	5	0
0	0	0	0	0

CALL Statements	FORMAT Statements
22	2
5	1
0	0

See Appendix A, Section A.18, for the file format.

2.2.3.6 Transaction Files

Transaction Files are sequential backup disk files that contain a record of all additions, deletions, and changes made to the data base since the last DBAM backup. (A DBAM tape backup run resets the number of transaction records to zero.) There are seven transaction files in the data base: one for each form type (CRF, CSF, CSR, RAF, and RSF), one for the CIFs, and one for the HIS Files. DBAM automatically adds to the Transaction Files whenever data in the data base are added, changed, or deleted.

See Appendix A, Section A.19, for the file format.

2.3 GENERAL NOTES ON THE DATA BASE DATA

All data on the data base are stored in character format. All fields displayed as numbers are right justified and

blank filled except for dates, which are zero filled with a format of YYMMDD. In many cases, an all-blank integer field (as opposed to a zero) indicates missing data.

Component codes are associated with component names on the CIFs, whereas all other coded fields are defined on the Encoding Dictionary.

All forms processed are given a unique six-character string (a letter followed by five digits)--for example, B00138.[1] The letter represents the form type as follows:

A or J	Run Analysis	RAF
B	Component Status	CSR
C	Resource Summary	RSF
D or K	Change Report	CRF
E or I	Component Summary	CSF

A phase flag (R, D, or M) indicates whether the form came from the requirements, development, or maintenance teams. (Development in this case refers to the time between the design start date and the cleanup end date as defined on the Phase Dates File.)

All but four file types have a status flag. (The Encoding Dictionary, the File Name and Status File, the Subjective Evaluations Directory File, and the SAP Output File do not have status flags.) New records are entered with a status of 1 (for "unchecked"). After hand validation, the status will be reset to 2. After data are verified by application, the status will be reset to 3.

---

[1]The format of the RAF, CRF, and CSR forms has evolved. Each revision of a form was assigned a new prefix to the form number. Thus, in some cases a file may contain form records with form numbers prefixed by one of two possible letters.

SECTION 3 - DATA BASE USER'S GUIDE

This section contains information on data base access and use. It is assumed that the user understands the basic operation and capabilities of the DEC PDP-11/70 (References 5 and 6). This section also describes the capabilities of DBAM (Reference 2), a general indexed file access program (DATATRIEVE) (Reference 7), and several other basic profile reporting programs. The support software that is described includes the following:

1. DBAM--Data Base Maintenance Software, used to access and validate data base data
2. SEL data base header files listing procedures--DATATRIEVE command procedures to list the contents of the SEL data base header files
3. NF--Form-counting report program that counts the number of forms by programmer for a given project
4. RPSTSCTR--Record-counting report program that counts the number of records on each data base file
5. WK--Hour- and form-counting report program that counts forms and programmer hours by programmer by week for a given project for any form type
6. PF--Basic profile report program that sums responses from files of any form type
7. RU--Resource utilization report program that summarizes manpower and computer resources
8. CS--Detailed component status report program that reports CSR File data by programmer by project
9. REP4, REP5--CIF reporting programs that list components, their software type, and Halstead measures

All these programs reside on DB1:[204,5] (except DATATRIEVE, which is already installed). Helpful user information also exists on the .HLP files on DB1:[204,5].

3.1 DATA BASE MAINTENANCE SOFTWARE

The DBAM system has five basic functions:

1. CREATE--Create new files for given project.
2. ARCHIVE--Back up all data base files on tape.
3. RESTORE--Restore all or specific files of the data base from the backup tape.
4. COMPRESS--Compress data base files to reduce space used and increase access efficiency.
5. UPDATE--Add, change, or delete data base records. All new data are validated to prevent the entry of incorrect data. UPDATE is the primary function used in the general data entry process.

To run this program, the user must log on under [204,3] (no password) and enter the following (the indirect command file):

```
@SELDBS
```

For complete information on how to run DBAM, see file DB1:[204,5]SELDBS.HLP or Reference 2.

3.2 SEL DATA BASE HEADER FILES LISTING PROCEDURES

The following five DATATRIEVE command procedures are used to list the contents of the five SEL data base header files:

1. DBRPTDIR--Lists the contents of the Subjective Evaluations Directory file and produces a formatted report in SEFDIR.RPT under the user's user identification code (UIC)

2. DBRPTENC--Lists the contents of the Encoding Dictionary and produces a formatted report in ENC.RPT under the user's UIC

3. DBRPTEST--Lists the contents of the Estimated Statistics File and produces two formatted reports in EST1.RPT and EST2.RPT under the user's UIC

4. DBRPTHDR--Lists the contents of the Phase Dates File and produces a formatted report in HDR.RPT under the user's UIC

5. DBRPTSTS--Lists the contents of the File Name and Status File and produces a formatted report in STAT.RPT under the user's UIC

DATATRIEVE is a DEC-supplied, file-access program allowing formatted listings to be made of the record contents of any Record Management System (RMS) indexed file. DATATRIEVE should be used to verify exactly what data exist in the data base. To execute these procedures, the user enters DTR. A prompt of "DTR>" is displayed to indicate that DATATRIEVE is running. The user can then enter the indirect file name for the desired listings: @[204,4]DBRPTDIR.DTR, @[204,4]DBRPTENC.DTR, @[204,4]DBRPTEST.DTR, @[204,4]DBRPTHDR.DTR, or @[204,4]DBRPTSTS.DTR. For more information on how to use DATATRIEVE, see Reference 7.

3.3 FORM COUNTER (NF)

This form-counting program produces a one-page report of the number of each type of form on the data base for each programmer for a particular project. Indirect files are allowed in response to the prompt for a project name.

3.4 RECORD COUNTER (RPSTSCTR)

This record-counting program produces a single-page report of the number of all records in all file types for all projects. Note that for some file types, the number of records

equals the number of forms and that for other file types they are not equal.

3.5 HOUR AND FORM COUNTER BY WEEK (WK)

This program produces a one- to two-page report of the number of forms or the number of hours or runs by programmer by week for a specific project. Indirect files are allowed.

3.6 GENERALIZED RESPONSE ACCUMULATOR (PF)

This is a basic profile program that currently reports on four file types: the CIF, the CRF File, the CSF File, and the RAF File. This program reports the counts of the responses of each field broken down by another field count. Indirect files are allowed.

3.7 RESOURCE UTILIZATION REPORT (RU)

The resource utilization report program produces a three-page report of manpower and computer resource data of a given project. There are two sections to the report. The first is a summary of programmer, manager, and services hours broken down by the five middle phases on the Phase Dates File. The second section shows run, change, and line counts.

This program obtains the resource data first from the RSF File and then from the CSR File.

3.8 DETAILED COMPONENT STATUS REPORT (CS)

This program produces a report of the data on a specific project's CSR File. The report prints separate sections for each programmer on the project. Each section has two parts: the activity section, which is a summary of OTHER hours, and the component section, which lists the hours spent on each component.

3.9 COMPONENT INFORMATION FILE REPORTS (REP4, REP5)

Two similar report programs produce detailed reports of the CIF. The first, REP5, produces a list of components and their associated Halstead parameters computed from the basic data on each CIF record. (For more information on Halstead's measures, see Reference 4.) The second report, REP4, produces a similar list of components and associated data by type of software and sorted by number of executable statements.

The type of software categories used in REP4 are listed below:

Code	Type
A	I/O (input/output)
B	Control/driver
BA	Control/driver with I/O
C	Control/computational
CA	Control/computational with I/O
D	Algorithmic/data transfer
DA	Algorithmic/data transfer with I/O
E	Block data

3.10 POTENTIAL PROBLEMS

This subsection contains some notes on situations that may prevent further processing.

1. If an unfamiliar abnormal end (ABEND) of execution occurs while running a program, the complete error message should be recorded and brought to the attention of programming personnel or the data base administrator. An ABEND may lock files, which means that those files are inaccessible and the program may not be run again until the files are unlocked.

2. To unlock a locked file, the user must either log on with the UIC of the owner of the file or use the main console (privileged UIC) and enter "PIP file,ext/UN".

3. If the VT100 keyboard locks for any reason (nothing can be entered), the SET-UP key should be pressed twice to unlock it.

4. If a user program continues to run beyond its desired use, it can be terminated or stopped by entering "ABO TTn" (ABORT), where n is the terminal number. If it is an installed program such as DTR or FOR, it can be terminated by entering "ABO nam", where nam is the three-letter name of the program. If new output continues to be displayed on the screen, CONTROL C should be entered before trying to terminate.

APPENDIX A - DATA BASE FILE FORMATS

This appendix describes, in detail, field definitions for all files in the data base.

Section	Page	Record Length	Name or Extension	File Description
A.1	A-2	60	ENCODE.HDR	Encoding Dictionary
A.2	A-3	120	EST.HDR	Estimated Statistics
A.3	A-5	52	STAT.HDR	File Name and Status
A.4	A-6	112	HEADER.HDR	Phase Dates
A.5	A-8	Variable	SEF.HDR	Subjective Evaluations
A.6	A-48	100	DIR.HDR	Subjective Evaluations Directory
A.7	A-49	72	ATM	Attitude Maintenance Change Report
A.8	A-51	101	CRF	Change Report Form
A.9	A-55	79	CSR	Component Status Report
A.10	A-56	250	CSF	Component Summary Form
A.11	A-61	0	GPS	General Project Summary
A.12	A-62	115	RSF	Resource Summary Form
A.13	A-63	53	RAF	Run Analysis Form
A.14	A-65	67	ACC	Accounting Information
A.15	A-67	104	CMT	Comment
A.16	A-68	80	CIF	Component Information
A.17	A-70	29	HIS	Growth History
A.18	A-71	78	ALL.SAP	Source Analyzer Program output (for all projects)
A.19	A-72	-	-	Transaction (different record length for each file)

The seven Transaction Files are located on DB0:[204,1]. Component codes are defined in the CIF.

A.1 ENCODING DICTIONARY (ENC) FILE

Item	Location	Format	Description
1	1-3	I3	Code type Numeric code identifying the category
2	4-8	A5	Code Alphanumeric code identifying a particular value
3	9-16	A8	Abbreviation (e.g., JCLERROR)
4	17-60	44A1	Verbal description of code

Primary key: Code type and code (bytes 1 through 8)

Secondary key: Code type and abbreviation (bytes 1 through 3 and 9 through 16, split key)

A.2 ESTIMATED STATISTICS (EST) FILE

Item	Location	Format	Description
1	1-8	8A1	Project name (e.g., MAGBIAS)
2	9-10	I2	Project code from ENCODE.HDR
3	11-14	I4	Number of components
4	15-18	I4	Total number of modules
5	19-22	I4	Number of new modules
6	23-26	I4	Number of modified modules
7	27-32	I6	Number of computer runs
8	33-38	I6	Number of source code changes
9	39-44	I6	Number of pages of documentation
10	45-50	I6	Total number of lines of code
11	51-56	I6	Number of new lines of code
12	57-62	I6	Number of modified lines of code
13	63-68	I6	Total number of executable statements
14	69-74	I6	Number of new executable statements
15	75-80	I6	Number of modified executable statements
16	81-86	F6.1	Programmer work hours (in tenths)
17	87-92	F6.1	Management work hours (in tenths)
18	93-98	F6.1	Other (services) work hours (in tenths)
19	99-104	F6.1	IBM S/360-95 computer hours (in tenths)
20	105-110	F6.1	IBM S/360-75 computer hours (in tenths)
21	111-116	F6.1	Other computer hours (in tenths)

Item	Location	Format	Description
22	117	I1	Status flag: = 1, unchecked = 2, hand checked = 3, verified by application
23	118	A1	Active flag: = Y, active = N, inactive = blank, no response
24	119	I1	Project category: = 1, attitude oriented = 2, orbit oriented = 3, scientific oriented = 4, data base oriented = 5, tool = 6, real time = 7, other = blank, no response
25	120	A1	Spare

Primary key: Project code (bytes 9 through 10)

Secondary key: Project name (bytes 1 through 8)

A.3 FILE NAME AND STATUS (STS) FILE

Item	Location	Format	Description
1	1-2	I2	Project code from ENCODE.HDR
2	3-4	I2	File code from ENCODE.HDR
3	5-29	25A1	File name (fully qualified) (e.g., DB1:[204.1]SMM.RSF)
4	30-35	I6	Creation date of file (YYMMDD)
5	36-41	I6	Last backup date of file (YYMMDD)
6	42-47	I6	Last update date of file (YYMMDD)
7	48-52	I5	Number of records in file

Primary key: Project code and file code (bytes 1 through 4)

Secondary key: Project code (bytes 1 and 2)

Tertiary key: File code (bytes 3 and 4)

Quaternary key: File name (bytes 5 through 29)

A.4 PHASE DATES FILE (HDR)

Item	Location	Format	Description
1	1-8	8A1	Project name (e.g., MAGBIAS)
2	9-10	I2	Project code from ENCODE.HDR
3	11-12	I2	Development computer from ENCODE.HDR: = 1, IBM S/360 = 2, DEC PDP-11/70 = blank, no response
4	13-14	I2	Target computer from ENCODE.HDR: = 1, IBM S/360 = 2, DEC PDP-11/70 = blank, no response
5	15	I1	Extent of alien computer use
6			PHASE DATES
7	16-21	I6	Requirements start (YYMMDD)
8	22-27	I6	Requirements end (YYMMDD)
9	28-33	I6	Design start (YYMMDD)
10	34-39	I6	Design end (YYMMDD)
11	40-45	I6	Code and test start (YYMMDD)
12	46-51	I6	Code and test end (YYMMDD)
13	52-57	I6	System test start (YYMMDD)
14	58-63	I6	System test end (YYMMDD)
15	64-69	I6	Acceptance test start (YYMMDD)
16	70-75	I6	Acceptance test end (YYMMDD)
17	76-81	I6	Cleanup start (YYMMDD)
18	82-87	I6	Cleanup end (YYMMDD)
19	88-93	I6	Maintenance start (YYMMDD)
20	94-99	I6	Maintenance end (YYMMDD)
21	100-111	A12	Spares

Item	Location	Format	Description
22	112	I1	Status flag: = 1, unchecked = 2, hand checked = 3, verified by application

Primary key: Project code (bytes 9 and 10)

Secondary key: Project name (bytes 1 through 8)

A.5 SUBJECTIVE EVALUATIONS FILE (SEF)

Each project has seven records of varying length as described below.

A.5.1 SEF RECORD 1

Item	Location	Format	Description
1	1-2	I2	Project code from ENCODE.HDR
2	3	I1	Record sequence number
3	4	I1	Status flag for the Practices and Techniques (MT) measure: = 1, unchecked = 2, hand checked = 3, verified by application
4	5	I1	Evaluation code for the MT measure
			ORGANIZATION
5	6-7	F2.1	Chief programmer
6	8-9	F2.1	Not defined
			DESIGN
7	10-11	F2.1	Walkthroughs
8	12-13	F2.1	Formal reviews
9	14-15	F2.1	Formalisms
10	16-17	F2.1	Tree charts
11	18-19	F2.1	Program Design Language (PDL)
12	20-21	F2.1	Hierarchical Input Processing Output (HIPO)
13	22-23	F2.1	Top-down
14	24-25	F2.1	Iterative enhancement
15	26-27	F2.1	N-squared charts
16	28-29	F2.1	Not defined
17	30-31	F2.1	Not defined
18	32-33	F2.1	Not defined

Item	Location	Format	Description
			CODE
19	34-35	F2.1	Stubs
20	36-37	F2.1	Top-down
21	38-39	F2.1	Structured
22	40-41	F2.1	Walkthroughs
23	42-43	F2.1	Reading
24	44-45	F2.1	Configuration control
25	46-47	F2.1	Not defined
26	48-49	F2.1	Not defined
27	50-51	F2.1	Not defined
			TEST
28	52-53	F2.1	Formalism
29	54-55	F2.1	Followthrough
30	56-57	F2.1	Batch
31	58-59	F2.1	IV & V presence
32	60-61	F2.1	IV & V use
33	62-63	F2.1	Not defined
34	64-65	F2.1	Not defined
			SUMS
35	66-68	F3.1	Items 7 through 14
36	69-71	F3.1	Items 19 through 24
37	72-74	F3.1	Items 28 through 32
38	75-78	F4.1	Items 35 through 37 and item 5
39	79	I1	Status flag for the Tools (TS) measure: = 1, unchecked = 2, hand checked = 3, verified by application
40	80	I1	Evaluation code for the TS measure
41	81-82	F2.1	Formal training in methodology
42	83-84	F2.1	Informal training
43	85-86	F2.1	Methodology reinforcement

Item	Location	Format	Description
44	87-88	F2.1	Requirements language (MEDL-R)
45	89-90	F2.1	Design language (PDL)
46	91-92	F2.1	Precompiler (SFORT)
47	93-94	F2.1	Software aids (e.g., XREF, MAP, LIST)
48	95-96	F2.1	Librarian
49	97-98	F2.1	Data generators
50	99-100	F2.1	Terminals (TSO)
51	101-102	F2.1	Remote Job Processing (RJP)
52	103-104	F2.1	Configuration Analysis Tool (CAT)
53	105-106	F2.1	Not defined
54	107-108	F2.1	Not defined
55	109-110	F2.1	Not defined
56	111-113	F3.1	Sum items 41 through 52
57	114	I1	Status flag for the Documentation (DC) measure: = 1, unchecked = 2, hand checked = 3, verified by application
58	115	I1	Evaluation code for the DC measure
59	116-117	F2.1	SEL forms
60	118-119	F2.1	Design document
61	120-121	F2.1	Design decisions
62	122-123	F2.1	Semiformal quality assurance
63	124-125	F2.1	Activity notebooks
64	126-127	F2.1	Unit development folders
65	128-129	F2.1	Test plans
66	130-131	F2.1	User's guide/system description
67	132-133	F2.1	Formal treatment of user's guide/system description
68	134-135	F2.1	Weekly/monthly progress reports

Item	Location	Format	Description
69	136-137	F2.1	Not defined
70	138-139	F2.1	Not defined
71	140-141	F2.1	Not defined
72	142-143	F2.1	Not defined
73	144-145	F2.1	Not defined
			SUMS
74	146-148	F3.1	Items 59 through 68
75	149-152	F4.1	Item 38, item 56*500/600, and item 74
76	153-162	A10	Spares

Primary key: Project code and record sequence number (bytes 1 through 3)

A.5.2 SEF RECORD 2

Item	Location	Format	Description
1	1-2	I2	Project code from ENCODE.HDR
2	3	I1	Record sequence number
3	4	I1	Status flag for the Experience with Application (AP) measure: = 1, unchecked = 2, hand checked = 3, verified by application
4	5	I1	Evaluation code for the AP measure
5	6-7	F2.1	Expert 1
6	8-9	F2.1	Expert 2
7	10-11	F2.1	Expert 3
8	12-13	F2.1	Expert 4
9	14-15	F2.1	Expert 5
10	16-17	F2.1	Project manager
11	18-19	F2.1	Project leader
12	20-21	F2.1	Programmers
13	22-23	F2.1	Analysts
14	24-25	F2.1	Participation in requirements definition
15	26-27	F2.1	Participation in design
16	28-29	F2.1	Team interactions before project
17	30-31	F2.1	Not defined
18	32-33	F2.1	Not defined
19	34-35	F2.1	Not defined
			SUMS
20	36-38	F3.1	Items 5 through 9
21	39-41	F3.1	Items 10 through 12
22	42-44	F3.1	Items 14 through 16
23	45-47	F3.1	Items 5 through 16

Item	Location	Format	Description
24	48	I1	Status flag for the Effectiveness of Management (MG) measure: = 1, unchecked = 2, hand checked = 3, verified by application
25	49	I1	Evaluation code for the MG measure
			PRELIMINARY DESIGN
26	50-51	F2.1	Project manager
27	52-53	F2.1	Project leader
28	54-55	F2.1	Analysis manager
29	56-57	F2.1	Analysis leader
30	58-59	F2.1	Development manager
31	60-61	F2.1	Development leader
			DETAILED DESIGN
32	62-63	F2.1	Project manager
33	64-65	F2.1	Project leader
34	66-67	F2.1	Analysis manager
35	68-69	F2.1	Analysis leader
36	70-71	F2.1	Development manager
37	72-73	F2.1	Development leader
			IMPLEMENTATION
38	74-75	F2.1	Project manager
39	76-77	F2.1	Project leader
40	78-79	F2.1	Analysis manager
41	80-81	F2.1	Analysis leader
42	82-83	F2.1	Development manager
43	84-85	F2.1	Development leader
			SYSTEM TESTING
44	86-87	F2.1	Project manager
45	88-89	F2.1	Project leader
46	90-91	F2.1	Analysis manager
47	92-93	F2.1	Analysis leader

Item	Location	Format	Description
48	94-95	F2.1	Development manager
49	96-97	F2.1	Development leader
			ACCEPTANCE TESTING
50	98-99	F2.1	Project manager
51	100-101	F2.1	Project leader
52	102-103	F2.1	Analysis manager
53	104-105	F2.1	Analysis leader
54	106-107	F2.1	Development manager
55	108-109	F2.1	Development leader
			STABILITY
56	110-111	F2.1	Project manager
57	112-113	F2.1	Project leader
58	114-115	F2.1	Analysis manager
59	116-117	F2.1	Analysis leader
60	118-119	F2.1	Other changes
			SUMS
61	120-122	F3.1	Items 26 through 31
62	123-125	F3.1	Items 32 through 37
63	126-128	F3.1	Items 38 through 43
64	129-131	F3.1	Items 44 through 49
65	132-134	F3.1	Items 50 through 55
66	135-137	F3.1	Items 56 through 60
67	138-140	F3.1	Items 26, 32, 38, 44, 50
68	141-143	F3.1	Items 27, 33, 39, 45, 51
69	144-146	F3.1	Items 28, 34, 40, 46, 52
70	147-149	F3.1	Items 29, 35, 41, 47, 53
71	150-152	F3.1	Items 30, 36, 42, 48, 54
72	153-155	F3.1	Items 31, 37, 43, 49, 55
73	156-159	F4.1	Items 26 through 60
74	160	I1	Status flag for the Performance of Team (PF) measure: = 1, unchecked = 2, hand checked = 3, verified by application

Item	Location	Format	Description
75	161	I1	Evaluation code for the PF measure
76	162-164	F3.2	Design - programmers
			DESIGN - TECHNICAL STAFF
77	165-167	F3.2	Programmers and project managers
78	168-170	F3.2	Programmers, project managers, and analysis managers
79	171-173	F3.2	Programmers and development managers
			DESIGN - DEVELOPMENT MANAGEMENT
80	174-176	F3.2	Project
81	177-179	F3.2	Project and analysis
82	180-182	F3.2	Development
			DESIGN - INTERFACE MANAGEMENT
83	183-185	F3.2	Analysis
84	186-188	F3.2	Development
85	189-191	F3.2	Design - not defined
86	192-194	F3.2	Implementation - programmers
			IMPLEMENTATION - TECHNICAL STAFF
87	195-197	F3.2	Programmers and project managers
88	198-200	F3.2	Programmers, project managers and analysis managers
89	201-203	F3.2	Programmers and development managers
			IMPLEMENTATION - DEVELOPMENT MANAGEMENT
90	204-206	F3.2	Project
91	207-209	F3.2	Project and analysis
92	210-212	F3.2	Development

Item	Location	Format	Description
			IMPLEMENTATION - INTERFACE MANAGEMENT
93	213-215	F3.2	Analysis
94	216-218	F3.2	Development
95	219-221	F3.2	Implementation - not defined
96	222-224	F3.2	Test - programmers
			TEST - TECHNICAL STAFF
97	225-227	F3.2	Programmers and project managers
98	228-230	F3.2	Programmers, project managers, and analysis managers
99	231-233	F3.2	Programmers and development managers
			TEST - DEVELOPMENT MANAGEMENT
100	234-236	F3.2	Project
101	237-239	F3.2	Project and analysis
102	240-242	F3.2	Development
			TEST - INTERFACE MANAGEMENT
103	243-245	F3.2	Analysis
104	246-248	F3.2	Development
105	249-251	F3.2	Test - not defined
106	252-254	F3.2	Overall - programmers
			OVERALL - TECHNICAL STAFF
107	255-257	F3.2	Programmers and project managers
108	258-260	F3.2	Programmers, project managers, and analysis managers
109	261-263	F3.2	Programmers and development managers
			OVERALL - DEVELOPMENT MANAGEMENT
110	264-266	F3.2	Project
111	267-269	F3.2	Project and analysis

Item	Location	Format	Description
112	270-272	F3.2	Development
			OVERALL - INTERFACE MANAGEMENT
113	273-275	F3.2	Analysis
114	276-278	F3.2	Development
115	279-281	F3.2	Overall - not defined
			SUMS
116	282-285	F4.1	Items 23, 61, 62, and item 76*600/300
117	286-289	F4.1	Items 23, 61, 62, and item 77*600/309
118	290-293	F4.1	Items 23, 61, 62, and item 78*600/314
119	294-297	F4.1	Item 23, item 63*2, and item 86*600/300
120	298-301	F4.1	Item 23, item 63*2, and item 87*600/309
121	302-305	F4.1	Item 23, item 63*2, and item 88*600/314
122	306-309	F4.1	Items 23, 64, 65, and item 96*600/300
123	310-313	F4.1	Items 23, 64, 65, and item 97*600/309
124	314-317	F4.1	Items 23, 64, 65, and item 98*600/314
125	318-321	F4.1	Item 23, item 73*600/1750, and item 106*600/300
126	322-325	F4.1	Item 23, item 73*600/1750, and item 107*600/309
127	326-329	F4.1	Item 23, item 73*600/1750, and item 108*600/314
128	330-339	A10	Spares

Primary key: Project code and record sequence number (bytes 1 through 3)

A.5.3 SEF RECORD 3

Item	Location	Format	Description
1	1-2	I2	Project code from ENCODE.HDR
2	3	I1	Record sequence number
3	4	I1	Status flag for the Complexity of Problem (CP) measure: = 1, unchecked = 2, hand checked = 3, verified by application
4	5	I1	Evaluation code for the CP measure
			CONSTRAINT
5	6-7	F2.1	Memory
6	8-9	F2.1	Timing
7	10-11	F2.1	Amount of data in step
8	12-13	F2.1	Data base size
9	14-15	F2.1	Number of data sets
			COMMUNICATIONS
10	16-17	F2.1	Number of programs
11	18-19	F2.1	Number of subsystems
12	20-21	F2.1	Number of data sets
13	22-23	F2.1	Use of old code
14	24-25	F2.1	New algorithms
15	26-27	F2.1	Schedule
16	28-29	F2.1	Not defined
17	30-31	F2.1	Not defined
18	32-33	F2.1	Not defined
19	34-35	F2.1	Not defined
			SUMS
20	36-38	F3.1	Items 5 and 6
21	39-41	F3.1	Items 7 through 9
22	42-44	F3.1	Items 10 through 12
23	45-47	F3.1	Items 13 through 15

Item	Location	Format	Description
24	48-50	F3.1	Items 5 through 15
25	51	I1	Status flag for the Internal Influences on Project (IN) measure: = 1, unchecked = 2, hand checked = 3, verified by application
26	52	I1	Evaluation code for the IN measure
			OVERTIME
27	53-54	F2.1	Weekends
28	55-56	F2.1	Nights
29	57-58	F2.1	Early phases
			STAFFING PROBLEMS
30	59-60	F2.1	Design
31	61-62	F2.1	Turnover
32	63-64	F2.1	Early departure (acceptance testing)
33	65-66	F2.1	Extra help needed
			PROJECT MANAGER
34	67-68	F2.1	At start
35	69-70	F2.1	Turnover
36	71-72	F2.1	At end
37	73-74	F2.1	Team attitude
38	75-76	F2.1	Project leader turnover
39	77-78	F2.1	Number of project managers/leaders
40	79-80	F2.1	Not defined
41	81-82	F2.1	Not defined
			SUMS
42	83-85	F3.1	Items 27 through 29
43	86-88	F3.1	Items 30 through 33
44	89-91	F3.1	Items 34 through 36, 38, and 39
45	92-94	F3.1	Items 27 through 39

Item	Location	Format	Description
46	95	I1	Status flag for the External Influences on Project (EX) measure: = 1, unchecked = 2, hand checked = 3, verified by application
47	96	I1	Evaluation code for the EX measure
			REQUIREMENTS
48	97-98	F2.1	Changes
49	99-100	F2.1	Completeness
			SUPPORT
50	101-102	F2.1	Analysis
51	103-104	F2.1	Mission project
52	105-106	F2.1	Development manager
53	107-108	F2.1	Development leader
			OUTSIDE DEVELOPMENT
54	109-110	F2.1	Number of subsystems
55	111-112	F2.1	Frontend processors
56	113-114	F2.1	Ontime delivery
			SIMULATOR
57	115-116	F2.1	Availability
58	117-118	F2.1	Correctness
59	119-120	F2.1	Data support
			ANALYSIS LEADER
60	121-122	F2.1	At start
61	123-124	F2.1	Turnover
62	125-126	F2.1	At end
63	127-128	F2.1	Number of analysis leaders/ managers
			SUPPORT
64	129-130	F2.1	Software
65	131-132	F2.1	Hardware
66	133-134	F2.1	Not defined
67	135-136	F2.1	Not defined

Item	Location	Format	Description
			SUMS
68	137-139	F3.1	Items 48 and 49
69	140-142	F3.1	Items 50 through 53
70	143-145	F3.1	Items 54 through 56
71	146-148	F3.1	Items 57 through 59
72	149-151	F3.1	Items 60 through 63
73	152-154	F3.1	Items 64 and 65
74	155-157	F3.1	Items 48 through 65
75	158-161	F4.1	Item 24*650/550, item 45, and item 74*650/900
76	162-171	A10	Spares

Primary key: Project code and record sequence number (bytes 1 through 3)

A.5.4 SEF RECORD 4

Item	Location	Format	Description
1	1-2	I2	Project code from ENCODE.HDR
2	3	I1	Record sequence number
3	4	I1	Status flag for the Resources Available (RA) measure: = 1, unchecked = 2, hand checked = 3, verified by application
4	5	I1	Evaluation code for the RA measure
			DEVELOPMENT PROCESS
5	6-7	F2.1	Formal training
6	8-9	F2.1	Informal training
7	10-11	F2.1	Documentation
			SUPPORT SOFTWARE
8	12-13	F2.1	Instruction
9	14-15	F2.1	Maintenance
10	16-17	F2.1	Simulator
			COMPUTER SUPPORT
11	18-19	F2.1	Model 75
12	20-21	F2.1	Model 95
13	22-23	F2.1	Other model
14	24-25	F2.1	RJP
15	26-27	F2.1	TSO
16	28-29	F2.1	OPS
17	30-31	F2.1	Space
18	32-33	F2.1	Graphic device
19	34-35	F2.1	Not defined
			PERSONNEL
20	36-37	F2.1	Librarian
21	38-39	F2.1	Dedicated expert
22	40-41	F2.1	IV & V team

Item	Location	Format	Description
23	42-43	F2.1	Not defined
24	44-45	F2.1	Not defined
			SUMS
25	46-48	F3.1	Items 5 through 7
26	49-51	F3.1	Items 8 through 10
27	52-54	F3.1	Items 11 through 18
28	55-57	F3.1	Items 20 through 22
29	58-60	F3.1	Items 25 through 28
30	61	I1	Status flag for the Software Product (PR) measure: = 1, unchecked = 2, hand checked = 3, verified by application
31	62	I1	Evaluation code for the PR measure
32	63-64	F2.1	Cost of project
33	65-66	F2.1	Timeliness of completion
34	67-68	F2.1	Confidence in product
			SIZE
35	69-70	F2.1	New software
36	71-72	F2.1	Extensively modified software
37	73-74	F2.1	Slightly modified software
38	75-76	F2.1	Old software
39	77-78	F2.1	Readable
40	79-80	F2.1	Reliable documentation
			COMPLETENESS
41	81-82	F2.1	Design
42	83-84	F2.1	Code
43	85-86	F2.1	Testing
			MEET REQUIREMENTS
44	87-88	F2.1	Processing
45	89-90	F2.1	Memory
46	91-92	F2.1	Not defined

Item	Location	Format	Description
47	93-94	F2.1	Not defined
48	95-96	F2.1	Not defined
49	97-98	F2.1	Not defined
50	99-100	F2.1	Not defined
51	101-102	F2.1	Not defined
			SUMS
52	103-105	F3.1	Items 35 through 38
53	106-108	F3.1	Items 41 through 43
54	109-111	F3.1	Items 44 and 45
55	112-114	F3.1	Items 32 through 45
56	115	I1	Status flag for the Product/Process Performance (PP) measure: = 1, unchecked = 2, hand checked = 3, verified by application
57	116	I1	Evaluation code for the PP measure
			PRODUCT
58	117-118	F2.1	Reliability
59	119-120	F2.1	Performance
60	121-122	F2.1	Operational considerations
61	123-124	F2.1	Ease of testing
62	125-126	F2.1	Not defined
63	127-128	F2.1	Not defined
			PROCESS
64	129-130	F2.1	Visibility
65	131-132	F2.1	Planning and followthrough
66	133-134	F2.1	Stable schedule
67	135-136	F2.1	Stable with perturbations
68	137-138	F2.1	Timeliness of records
69	139-140	F2.1	Not defined
70	141-142	F2.1	Not defined
71	143-144	F2.1	Not defined

Item	Location	Format	Description
72	145-146	F2.1	Not defined
			SUMS
73	147-149	F3.1	Items 58 through 61
74	150-152	F3.1	Items 64 through 68
75	153-155	F3.1	Items 73 and 74
76	156-165	A10	Spares

Primary key: Project code and record sequence number (bytes 1 through 3)

A.5.5 SEF RECORD 5

Item	Location	Format	Description
1	1-2	I2	Project code from ENCODE.HDR
2	3	I1	Record sequence number
3	4	I1	Status flag for the Team Rank (RK) measure: = 1, unchecked = 2, hand checked = 3, verified by application
4	5	I1	Evaluation code for the RK measure
5	6-8	F3.1	Design - programmers
			DESIGN - TECHNICAL STAFF
6	9-11	F3.1	Programmers and project managers
7	12-14	F3.1	Programmers, project managers, and analysis managers
8	15-17	F3.1	Programmers and development managers
			DESIGN - DEVELOPMENT MANAGEMENT
9	18-20	F3.1	Project
10	21-23	F3.1	Project and analysis
11	24-26	F3.1	Development
			DESIGN - INTERFACE MANAGEMENT
12	27-29	F3.1	Analysis
13	30-32	F3.1	Development
14	33-35	F3.1	Design - not defined
15	36-38	F3.1	Implementation - programmers
			IMPLEMENTATION - TECHNICAL STAFF
16	39-41	F3.1	Programmers and project managers
17	42-44	F3.1	Programmers, project managers, and analysis managers

Item	Location	Format	Description
18	45-47	F3.1	Programmers and development managers
			IMPLEMENTATION - DEVELOPMENT MANAGEMENT
19	48-50	F3.1	Project
20	51-53	F3.1	Project and analysis
21	54-56	F3.1	Development
			IMPLEMENTATION - INTERFACE MANAGEMENT
22	57-59	F3.1	Analysis
23	60-62	F3.1	Development
24	63-65	F3.1	Implementation - not defined
25	66-68	F3.1	Test - programmers
			TEST - TECHNICAL STAFF
26	69-71	F3.1	Programmers and project managers
27	72-74	F3.1	Programmers, project managers, and analysis managers
28	75-77	F3.1	Programmers and development managers
			TEST - DEVELOPMENT MANAGEMENT
29	78-80	F3.1	Project
30	81-83	F3.1	Project and analysis
31	84-86	F3.1	Development
			TEST - INTERFACE MANAGEMENT
32	87-89	F3.1	Analysis
33	90-92	F3.1	Development
34	93-95	F3.1	Test - not defined
35	96-98	F3.1	Overall - programmers
			OVERALL - TECHNICAL STAFF
36	99-101	F3.1	Programmers and project managers
37	102-104	F3.1	Programmers, project managers, and analysis managers

Item	Location	Format	Description
38	105-107	F3.1	Programmers and development managers
			OVERALL - DEVELOPMENT MANAGEMENT
39	108-110	F3.1	Project
40	111-113	F3.1	Project and analysis
41	114-116	F3.1	Development
			OVERALL - INTERFACE MANAGEMENT
42	117-119	F3.1	Analysis
43	120-122	F3.1	Development
44	123-125	F3.1	Overall - not defined
45	126	I1	Status flag for the Years of Professional Experience (YP) measure: = 1, unchecked = 2, hand checked = 3, verified by application
46	127	I1	Evaluation code for the YP measure
47	128-130	F3.1	Design - programmers
			DESIGN - TECHNICAL STAFF
48	131-133	F3.1	Programmers and project managers
49	134-136	F3.1	Programmers, project managers, and analysis managers
50	137-139	F3.1	Programmers and development managers
			DESIGN - DEVELOPMENT MANAGEMENT
51	140-142	F3.1	Project
52	143-145	F3.1	Project and analysis
53	146-148	F3.1	Development
			DESIGN - INTERFACE MANAGEMENT
54	149-151	F3.1	Analysis

Item	Location	Format	Description
55	152-154	F3.1	Development
56	155-157	F3.1	Design - Not defined
57	158-160	F3.1	Implementation - programmers
			IMPLEMENTATION - TECHNICAL STAFF
58	161-163	F3.1	Programmers and project managers
59	164-166	F3.1	Programmers, project managers, and analysis managers
60	167-169	F3.1	Programmers and development managers
			IMPLEMENTATION - DEVELOPMENT MANAGEMENT
61	170-172	F3.1	Project
62	173-175	F3.1	Project and analysis
63	176-178	F3.1	Development
			IMPLEMENTATION - INTERFACE MANAGEMENT
64	179-181	F3.1	Analysis
65	182-184	F3.1	Development
66	185-187	F3.1	Implementation - not defined
67	188-190	F3.1	Test - programmers
			TEST - TECHNICAL STAFF
68	191-193	F3.1	Programmers and project managers
69	194-196	F3.1	Programmers, project managers, and analysis managers
70	197-199	F3.1	Programmers and development managers
			TEST - DEVELOPMENT MANAGEMENT
71	200-202	F3.1	Project
72	203-205	F3.1	Project and analysis
73	206-208	F3.1	Development

Item	Location	Format	Description
			[illegible] - INTERFACE MANAGEMENT
74	209-211	F3.1	Analysis
75	212-214	F3.1	Development
76	215-217	F3.1	Test - not defined
77	218-220	F3.1	Overall - programmers
			OVERALL - TECHNICAL STAFF
78	221-223	F3.1	Programmers and project managers
79	224-226	F3.1	Programmers, project managers, and analysis managers
80	227-229	F3.1	Programmers and development managers
			OVERALL - DEVELOPMENT MANAGEMENT
81	230-232	F3.1	Project
82	233-235	F3.1	Project and analysis
83	236-238	F3.1	Development
			OVERALL - INTERFACE MANAGEMENT
84	239-241	F3.1	Analysis
85	242-244	F3.1	Development
86	245-247	F3.1	Overall - not defined
87	248	I1	Status flag for the Years of Applicable Experience (YA) measure: = 1, unchecked = 2, hand checked = 3, verified by application
88	249	I1	Evaluation code for the YA measure
89	250-252	F3.1	Design - programmers
			DESIGN - TECHNICAL STAFF
90	253-255	F3.1	Programmers and project managers
91	256-258	F3.1	Programmers, project managers, and analysis managers

Item	Location	Format	Description
92	259-261	F3.1	Programmers and development managers
			DESIGN - DEVELOPMENT MANAGEMENT
93	262-264	F3.1	Project
94	265-267	F3.1	Project and analysis
95	268-270	F3.1	Development
			DESIGN - INTERFACE MANAGEMENT
96	271-273	F3.1	Analysis
97	274-276	F3.1	Development
98	277-279	F3.1	Design - not defined
99	280-282	F3.1	Implementation - programmers
			IMPLEMENTATION - TECHNICAL STAFF
100	283-285	F3.1	Programmers and project managers
101	286-288	F3.1	Programmers, project managers, and analysis managers
102	289-291	F3.1	Programmers and development managers
			IMPLEMENTATION - DEVELOPMENT MANAGEMENT
103	292-294	F3.1	Project
104	295-297	F3.1	Project and analysis
105	298-300	F3.1	Development
			IMPLEMENTATION - INTERFACE MANAGEMENT
106	301-303	F3.1	Analysis
107	304-306	F3.1	Development
108	307-309	F3.1	Implementation - not defined
109	310-312	F3.1	Test - programmers
			TEST - TECHNICAL STAFF
110	313-315	F3.1	Programmers and project managers

Item	Location	Format	Description
111	316-318	F3.1	Programmers, project managers, and analysis managers
112	319-321	F3.1	Programmers and development managers
			TEST - DEVELOPMENT MANAGEMENT
113	322-324	F3.1	Project
114	325-327	F3.1	Project and analysis
115	328-330	F3.1	Development
			TEST - INTERFACE MANAGEMENT
116	331-333	F3.1	Analysis
117	334-336	F3.1	Development
118	337-339	F3.1	Test - not defined
119	340-342	F3.1	Overall - programmers
			OVERALL - TECHNICAL STAFF
120	343-345	F3.1	Programmers and project managers
121	346-348	F3.1	Programmers, project managers, and analysis managers
122	349-351	F3.1	Programmers and development managers
			OVERALL - DEVELOPMENT MANAGEMENT
123	352-354	F3.1	Project
124	355-357	F3.1	Project and analysis
125	358-360	F3.1	Development
			OVERALL - INTERFACE MANAGEMENT
126	361-363	F3.1	Analysis
127	364-366	F3.1	Development
128	367-369	F3.1	Overall - not defined
129	370	I1	Status flag for the Years of Environment Experience (YE) measure: = 1, unchecked = 2, hand checked = 3, verified by application

Item	Location	Format	Description
130	371	I1	Evaluation code for the YE measure
131	372-374	F3.1	Design - programmers
			DESIGN - TECHNICAL STAFF
132	375-377	F3.1	Programmers and project managers
133	378-380	F3.1	Programmers, project managers, and analysis managers
134	381-383	F3.1	Programmers and development managers
			DESIGN - DEVELOPMENT MANAGEMENT
135	384-386	F3.1	Project
136	387-389	F3.1	Project and analysis
137	390-392	F3.1	Development
			DESIGN - INTERFACE MANAGEMENT
138	393-395	F3.1	Analysis
139	396-398	F3.1	Development
140	399-401	F3.1	Design - not defined
141	402-404	F3.1	Implementation - programmers
			IMPLEMENTATION - TECHNICAL STAFF
142	405-407	F3.1	Programmers and project managers
143	408-410	F3.1	Programmers, project managers, and analysis managers
144	411-413	F3.1	Programmers and development managers
			IMPLEMENTATION - DEVELOPMENT MANAGEMENT
145	414-416	F3.1	Project
146	417-419	F3.1	Project and analysis
147	420-422	F3.1	Development

Item	Location	Format	Description
			IMPLEMENTATION - INTERFACE MANAGEMENT
148	423-425	F3.1	Analysis
149	426-428	F3.1	Development
150	429-431	F3.1	Implementation - not defined
151	432-434	F3.1	Test - programmers
			TEST - TECHNICAL STAFF
152	435-437	F3.1	Programmers and project managers
153	438-440	F3.1	Programmers, project managers, and analysis managers
154	441-443	F3.1	Programmers and development managers
			TEST - DEVELOPMENT MANAGEMENT
155	444-446	F3.1	Project
156	447-449	F3.1	Project and analysis
157	450-452	F3.1	Development
			TEST - INTERFACE MANAGEMENT
158	453-455	F3.1	Analysis
159	456-458	F3.1	Development
160	459-461	F3.1	Test - not defined
161	462-464	F3.1	Overall - programmers
			OVERALL - TECHNICAL STAFF
162	465-467	F3.1	Programmers and project managers
163	468-470	F3.1	Programmers, project managers and analysis managers
164	471-473	F3.1	Programmers and development managers
			OVERALL - DEVELOPMENT MANAGEMENT
165	474-476	F3.1	Project
166	477-479	F3.1	Project and analysis
167	480-482	F3.1	Development

Item	Location	Format	Description
			OVERALL - INTERFACE MANAGEMENT
168	483-485	F3.1	Analysis
169	486-488	F3.1	Development
170	489-491	F3.1	Overall - not defined
171	492-501	A10	Spares

Primary key: Project code and record sequence number (bytes 1 through 3)

A.5.6 SEF RECORD 6

Item	Location	Format	Description
1	1-2	I2	Project code from ENCODE.HDR
2	3	I1	Record sequence number
3	4	I1	Status flag for the Walston-Felix Model (WF) measure: = 1, unchecked = 2, hand checked = 3, verified by application
4	5	I1	Evaluation code for the WF measure
5	6-7	F2.1	Experience with application
6	8-9	F2.1	Participation in requirements definition
7	10-11	F2.0	Percentage of programmers in design
			PROGRAMMERS'
8	12-13	F2.1	Qualifications
9	14-15	F2.1	Familiarity with machine
10	16-17	F2.1	Familiarity with language
11	18-19	F2.1	Familiarity with graphics
12	20-21	F2.1	Familiarity with application
13	22-23	F2.1	Degree to which personnel worked together
14	24-25	F2.1	Not defined
15	26-27	F2.1	Customer participation in requirements definition
16	28-29	F2.1	Customer interface
17	30-31	F2.1	Customer-originated design changes
18	32-33	F2.1	Application processing
19	34-35	F2.1	Program flow
20	36-37	F2.1	Interprogram communications
21	38-39	F2.1	External communications

Item	Location	Format	Description
22	40-41	F2.1	Data base structure
23	42-43	F2.1	Percentage of code, real-time or graphics
24	44-45	F2.1	Storage constraint
25	46-47	F2.1	Timing constraint
26	48-49	F2.1	I/O constraint
27	50-51	F2.0	Items in data base
28	52-53	F2.1	Hardware under development
29	54-55	F2.1	Unclassified
30	56-57	F2.1	Not defined
31	58-59	F2.1	Not defined
32	60-61	F2.1	Not defined
33	62-63	F2.1	Not defined
34	64-65	F2.1	Not defined
			PERCENTAGE OF DEVELOPMENT
35	66-68	F3.1	On IBM S/360-95
36	69-71	F3.1	On IBM S/360-75
37	72-74	F3.1	At STL
38	75-77	F3.1	Percentage of programmers in design
39	78-80	F3.1	Percentage of previous personnel interactions
			PERCENTAGE OF ENVIRONMENT
40	81-83	F3.1	Closed
41	84-86	F3.1	Open with respect
42	87-89	F3.1	Open
43	90-92	F3.1	RJE
44	93-95	F3.1	TSO
			PERCENTAGE OF CODE
45	96-98	F3.1	Structured
46	99-101	F3.1	Read
47	102-104	F3.1	Developed top-down
48	105-107	F3.1	Via chief programmer

Item	Location	Format	Description
			PERCENTAGE OF EFFORT
49	108-110	F3.1	Management
50	111-113	F3.1	Administration
51	114-116	F3.1	Programmers
52	117-119	F3.1	Analysts
53	120-122	F3.1	Operators
54	123-125	F3.1	Others
55	126-130	F5.2	Total staff-months
56	131-135	F5.2	Total cost in programmer units (staff-months)
57	136-138	F3.1	Not defined
58	139-141	F3.1	Percentage of schedule to complete acceptance testing (actual workweeks)
59	142-144	F3.0	Total weeks to complete project (workweeks)
60	145-147	I3	Not defined
61	148-150	I3	Not defined
62	151-153	I3	Not defined
63	154-156	I3	Not defined
64	157-159	I3	Not defined
			PERCENTAGE OF CODE
65	160-162	F3.1	Nonmathematical and I/O formatting
66	163-165	F3.1	Mathematical and computational
67	166-168	F3.1	CPU and I/O control
68	169-171	F3.1	Fallback and recovery
69	172-174	F3.1	Other
70	175-177	F3.1	Real-time or graphics
			DEVELOPED LINES
71	178-183	I6	Of ALC
72	184-189	I6	Of macros
73	190-195	I6	Of FORTRAN
74	196-201	I6	Total developed lines

Item	Location	Format	Description
			DELIVERED LINES
75	202-207	I6	Of ALC
76	208-213	I6	Of macros
77	214-219	I6	Of FORTRAN
78	220-225	I6	Total delivered lines
79	226-229	I4	Items in data base
80	230-233	I4	Pages of documentation
81	234-237	I4	Not defined
82	238-241	I4	Not defined
83	242-245	I4	Not defined
84	246-249	I4	Not defined
			SUMS
85	250-252	F3.1	Items 5 through 13
86	253-255	F3.1	Items 15 through 29
87	256	I1	Status flag for the PRICE S3 Model (PS) measure: = 1, unchecked = 2, hand checked = 3, verified by application
88	257	I1	Evaluation code for the PS measure
			PERCENTAGE OF SCHEDULE
89	258-260	F3.1	Design phase (from start)
90	261-263	F3.1	Design activity (from start)
91	264-266	F3.1	Coding phase (from design phase)
92	267-269	F3.1	Coding activity (from design phase)
93	270-272	F3.1	Test phase (from coding phase)
94	273-275	F3.1	Test activity (from documentation phase)
95	276-278	F3.1	System documentation phase (from end)
96	279-281	F3.1	Documentation activity (from end)

Item	Location	Format	Description
97	282-285	F4.3	Ratio of actual schedule to 67-week schedule
			COMPLEXITY FACTOR
98	286-288	F3.2	Total
99	289-291	F3.2	Personnel only
100	292-294	F3.2	Product only
101	295-297	F3.2	External effects only
102	298-300	F3.1	New design - percentage of code in wholly new components
103	301-303	F3.1	New code - percentage of code in new and extensively modified components
104	304-306	F3.1	New test - percentage of code in new or modified components
105	307-309	F3.2	Application - instruction mix
106	310-312	F3.2	Resource - skill mix and experience for cost
107	313-315	F3.2	Utility - fraction of storage and timing capacity
108	316-318	F3.2	Platform - strictness of standards, e.g., MIL-Spec
109	319-321	F3.2	Sum items 98 through 101
110	322	I1	Status flag for the COCOMO Model (CO) measure: = 1, unchecked = 2, hand checked = 3, verified by application
111	323	I1	Evaluation code for the CO measure
			PRODUCT
112	324-326	F3.2	Required software reliability
113	327-329	F3.2	Data base size
114	330-332	F3.2	Product complexity

Item	Location	Format	Description
			COMPUTER
115	333-335	F3.2	Execution time constraint
116	336-338	F3.2	Main storage constraint
117	339-341	F3.2	Virtual machine volatility
118	342-344	F3.2	Computer turnaround time
			PERSONNEL
119	345-347	F3.2	Analyst capability
120	348-350	F3.2	Applications experience
121	351-353	F3.2	Programmer capability
122	354-356	F3.2	Virtual machine experience
123	357-359	F3.2	Programming language experience
			PROJECT
124	360-362	F3.2	Use of modern programming practices
125	363-365	F3.2	Use of software tools
126	366-368	F3.2	Required development schedule
127	369-378	A10	Spares

Primary key: Project code and record sequence number (bytes 1 through 3)

A.5.7 SEF RECORD 7

Item	Location	Format	Description
1	1-2	I2	Project code from ENCODE.HDR
2	3	I1	Record sequence number
3	4	I1	Status flag for the Miscellaneous (MS) measure: = 1, unchecked = 2, hand checked = 3, verified by application
4	5	I1	Evaluation code for the MS measure
			PRODUCT
5	6-7	F2.0	Number of programs
6	8-9	F2.0	Number of subsystems
			DATA SETS
7	10-11	F2.0	Input
8	12-13	F2.0	Input/output
9	14-15	F2.0	Output
10	16-17	F2.0	Total
			DATA BASE
11	18-21	I4	Input
12	22-25	I4	Input/output
13	26-29	I4	Output
14	30-33	I4	Total
			PROCESSING
15	34-35	F2.0	Number of programs
16	36-37	F2.0	Number of subsystems
			DATA SETS
17	38-39	F2.0	Input
18	40-41	F2.0	Input/output
19	42-43	F2.0	Output
20	44-45	F2.0	Total
			DATA BASE
21	46-49	I4	Input
22	50-53	I4	Input/output

Item	Location	Format	Description
23	54-57	I4	Output
24	58-61	I4	Total
			DOCUMENTATION
25	62-65	I4	Pages of design document
26	66-69	I4	Pages of test plan
27	70-73	I4	Pages of user's guide/ system description
28	74-77	I4	Pages of prologs
29	78-82	I5	Total pages
			AVERAGE STAFF
30	83-85	F3.1	Programmers
31	86-88	F3.1	Programmers and managers
32	89-91	F3.1	All personnel
33	92-95	I4	Not defined
34	96-99	I4	Not defined
35	100-103	I4	Not defined
36	104-107	I4	Not defined
37	108-111	I4	Not defined
38	112-115	I4	Not defined
39	116-119	I4	Not defined
40	120-123	I4	Not defined
41	124-127	I4	Not defined
42	128-131	I4	Not defined
43	132-135	I4	Not defined
44	136-139	I4	Not defined
45	140	I1	Status flag for the Code Breakdown (SW) measure: = 1, unchecked = 2, hand checked = 3, verified by application
46	141	I1	Evaluation code for the SW measure
			BASELINE DIAGRAM COMPONENTS
47	142-145	I4	New
48	146-149	I4	Extensively modified

Item	Location	Format	Description
49	150-153	I4	Slightly modified
50	154-157	I4	Old
51	158-161	I4	Total
			DECISION MODULES
52	162-165	I4	New
53	166-169	I4	Extensively modified
54	170-173	I4	Slightly modified
55	174-177	I4	Old
56	178-181	I4	Total
			LOC ALC
57	182-187	I6	New
58	188-193	I6	Extensively modified
59	194-199	I6	Slightly modified
60	200-205	I6	Old
61	206-211	I6	Total
			LOC MACROS
62	212-217	I6	New
63	218-223	I6	Extensively modified
64	224-229	I6	Slightly modified
65	230-235	I6	Old
66	236-241	I6	Total
			LOC FORTRAN
67	242-247	I6	New
68	248-253	I6	Extensively modified
69	254-259	I6	Slightly modified
70	260-265	I6	Old
71	266-271	I6	Total
			LOC TOTAL
72	272-277	I6	New
73	278-283	I6	Extensively modified
74	284-289	I6	Slightly modified
75	290-295	I6	Old
76	296-301	I6	Total

Item	Location	Format	Description
			EXECUTABLE ALC
77	302-307	I6	New
78	308-313	I6	Extensively modified
79	314-319	I6	Slightly modified
80	320-325	I6	Old
81	326-331	I6	Total
			EXECUTABLE MACROS
82	332-337	I6	New
83	338-343	I6	Extensively modified
84	344-349	I6	Slightly modified
85	350-355	I6	Old
86	356-361	I6	Total
			EXECUTABLE FORTRAN
87	362-367	I6	New
88	368-373	I6	Extensively modified
89	374-379	I6	Slightly modified
90	380-385	I6	Old
91	386-391	I6	Total
			EXECUTABLE TOTAL
92	392-397	I6	New
93	398-403	I6	Extensively modified
94	404-409	I6	Slightly modified
95	410-415	I6	Old
96	416-421	I6	Total
			DECISIONS
97	422-426	I5	New
98	427-431	I5	Extensively modified
99	432-436	I5	Slightly modified
100	437-441	I5	Old
101	442-446	I5	Total
			LIBRARY CHANGES
102	447-451	I5	New
103	452-456	I5	Extensively modified

Item	Location	Format	Description
104	457-461	I5	Slightly modified
105	462-466	I5	Old
106	467-471	I5	Total
			SOFTWARE CHANGES
107	472-475	I4	New
108	476-479	I4	Extensively modified
109	480-483	I4	Slightly modified
110	484-487	I4	Old
111	488-491	I4	Total
			SOFTWARE ERRORS
112	492-495	I4	New
113	496-499	I4	Extensively modified
114	500-503	I4	Slightly modified
115	504-507	I4	Old
116	508-511	I4	Total
			PERCENTAGE OF COMMENTS
117	512-513	F2.0	New
118	514-515	F2.0	Extensively modified
119	516-517	F2.0	Slightly modified
120	518-519	F2.0	Old
121	520-521	F2.0	Total
			ERRORS
122	522-525	F4.2	Per 1000 LOC
123	526-529	F4.2	Per 1000 executable LOC
124	530-533	F4.1	Per 1000 decisions
125	534-536	F3.2	Per baseline diagram component
126	537-539	F3.2	Per decision module
			DECISIONS
127	540-542	F3.0	Per 1000 LOC
128	543-545	F3.0	Per 1000 executable LOC
129	546-548	F3.1	Per baseline diagram component
130	549-551	F3.1	Per decision module

Item	Location	Format	Description
131	552-554	F3.3	Ratio of LOC to expanded LOC EXECUTABLE LOC
132	555-557	F3.0	Per 1000 LOC
133	558-560	F3.1	Per baseline diagram component
134	561-563	F3.0	Per decision module
135	564-566	F3.2	Data set components per change
136	567-568	F2.0	Percentage of errors in changes
137	569-578	A10	Spares

Primary key: Project code and record sequence number (bytes 1 through 3)

A.6 SUBJECTIVE EVALUATIONS DIRECTORY (DIR) FILE

Item	Location	Format	Description
1	1-4	A4	Code
2	5-12	A8	Name (measure)
3	13-18	I6	Minimum value
4	19-24	I6	Maximum value
5	25	I1	Data record sequence number (1 through 7)
6	26-28	I3	Byte location in data record
7	29-100	72A1	Verbal description of the measure

Primary key: Code (bytes 1 through 4)

Secondary key: Name (bytes 5 through 12)

A.7 ATTITUDE MAINTENANCE CHANGE REPORT (ATM) FILE

Item	Location	Format	Description
1	1-6	A6	Form number
2	7-8	I2	Project code from ENCODE.HDR
3	9-14	I6	Form date (YYMMDD)
4	15-20	I6	Date change determined to be necessary (YYMMDD)
5	21	A1	Description comment flag: = T, true = F, false
6	22-23	I2	Number of components changed
7	24-38	I3	Component codes from CIF: = T, true = F, false
			TYPE OF CHANGE (nonerrors only)
8	39	A1	Requirements
9	40	A1	New information or data
10	41	A1	Specification
11	42	A1	Design
12	43	A1	Hardware environment
13	44	A1	Software environment
14	45	A1	Optimization
15	46	A1	Other
			ERROR DETECTION ACTIVITIES: = D, detection = I, isolation = B, both
16	47	A1	Normal use
17	48	A1	Test runs
18	49	A1	Code reading
19	50	A1	Reading documentation
20	51	A1	Trace/dump
21	52	A1	Cross-reference/attitude list

Item	Location	Format	Description
22	53	A1	System error messages
23	54	A1	Project-specific error message
24	55	A1	Other
25	56	I1	Primary error type from ENCODE.HDR: = 1, requirements error = 2, design error = 3, error translating design or specifications to code = 4, specifications error = 5, clerical error = 6, other = 7, no response
26	57	A1	Related to previous change: = Y, yes = N, no = C, can't tell = blank, no response
27	58-62	I5	Programmer code from ENCODE.HDR
28	63-68	I6	Change start date (YYMMDD)
29	69	I1	Time spent on change: = 1, less than 1 day = 2, 1 day to 1 week = 3, more than 1 week = 4, no response
30	70	I1	Status flag: = 1, unchecked = 2, hand checked = 3, verified by application
31	71	A1	Comment flag: = Y, yes = N, no
32	72-77	A6	Spares

Primary key: Form number (bytes 1 through 6)

A.8 CHANGE REPORT FORM (CRF) FILE

Item	Location	Format	Description
1	1-6	A6	Form number (e.g., D00633)
2	7-8	I2	Project code from ENCODE.HDR
3	9-13	I5	Programmer code from ENCODE.HDR
4	14-19	I6	Form date (YYMMDD)
5	20-21	I2	Number of components changed (may be greater than 5)
6	22-23	I2	Number of components examined
7	24	I1	More than one component affected: = Y, yes = N, no = blank, no response
8	25-30	I6	Date change was determined to be necessary (YYMMDD)
9	31-36	I6	Date change started (YYMMDD)
10	37	I1	Effort for change from ENCODE.HDR: = 1, less than 1 hour = 2, 1 hour to 1 day = 3, 1 day to 3 days = 4, over 3 days = blank, no response
11	38-41	4A1	Type of change (up to 4 responses, from ENCODE.HDR): = 1, error correction = 2, planned enhancement = 3, implement requirements change = 4, improve clarity = 5, improve user service = 6, develop utility only

Item	Location	Format	Description
			= 7, optimization = 8, adapt to environmental change = 9, other = blank, no response
12	42-56	5I3	Codes of changed components from CIF
13	57-60	4I1	Type of error (up to 4 responses, from ENCODE.HDR): = 1, requirements incorrect = 2, functional specifications incorrect = 3, design error of several components = 4, design error of one component = 5, misunderstanding of external environment = 6, error in language use = 7, clerical error = 8, other = blank, no response
14	61	I1	When error entered system from ENCODE.HDR: = 1, requirements = 2, functional specification = 3, design = 4, code and test = 5, other = 6, can't tell = blank, no response
15	62	A1	Data structure error: = X, yes = blank, no
16	63	A1	Control logic error: = X, yes = blank, no

Item	Location	Format	Description
17			ACTIVITIES USED TO ISOLATE ERROR (up to 5 responses, each from ENCODE.HDR): = 1, preacceptance test = 2, acceptance test = 3, postacceptance test = 4, inspection of output = 5, code reading by programmer = 6, code read by another = 7, talk with other programmers = 8, special debug code = 9, system error message = A, project-specific error message = B, reading documentation = C, trace = D, dump = E, cross-reference = F, proof technique = G, other = blank, no response
	64-68	5A1	For program validation
	69-73	5A1	For detection symptoms
	74-78	5A1	Tried in finding cause
	79-83	5A1	For finding cause
18	84	I1	Time to isolate error from ENCODE.HDR: = 1, less than 1 hour = 2, 1 hour to 1 day = 3, more than 1 day = 4, never found = blank, no response
19	85	A1	Workaround used: = Y, yes = N, no = blank, no response

Item	Location	Format	Description
20	86	A1	Related to previous change: = Y, yes = N, no = C, can't tell = blank, no response
21	87-91	I5	Previous form number (excludes first character, includes leading zeros, e.g., 00633)
22	92-97	I6	Previous form date (YYMMDD)
23	98	A1	Reason comment flag: = Y, yes = N, no
24	99	A1	Description comment flag: = Y, yes = N, no
25	100	A1	General comment flag: = Y, yes = N, no
26	101	I1	Status flag: = 1, unchecked = 2, hand checked = 3, verified by application

Primary key: Form number (bytes 1 through 6)

A.9 COMPONENT STATUS REPORT (CSR) FILE

Item	Location	Format	Description
1	1-6	A6	Form number (e.g., B00952)
2	7-8	I2	Sequence number
3	9-10	I2	Project code from ENCODE.HDR
4	11-15	I5	Programmer code from ENCODE.HDR
5	16-21	I6	Form date (YYMMDD)
6	22-24	I3	Component code from CIF
7	25-60	9F4.1	Work hours spent in each phase (in tenths)
8	61-68	A8	Other activity name
9	69-72	F4.1	Other activity work hours (in tenths)
10	73	I1	Status flag: = 1, unchecked = 2, hand checked = 3, verified by application
11	74	A1	Source of Data (Phase) flag: = R, requirements team = D, development team = M, maintenance team
12	75-79	A5	Spares

Primary key: Form and sequence number (bytes 1 through 8)

Secondary key: Component code (bytes 22 through 24)

Tertiary key: Programmer code (bytes 11 through 15)

A.10 COMPONENT SUMMARY FORM (CSF) FILE

Item	Location	Format	Description
1	1-6	A6	Form number (e.g., I00633)
2	7-8	I2	Project code from ENCODE.HDR
3	9-13	I5	Programmer filling out form from ENCODE.HDR
4	14-18	I5	Programmer implementing component from ENCODE.HDR
5	19-24	I6	Form date (YYMMDD)
6	25	A1	Form stage: = N, new = U, under development = C, complete = blank, no response
7	26-28	I3	Component code from CIF
8	29	I1	Precision of specification from ENCODE.HDR: = 1, very precise = 2, precise = 3, imprecise = blank, no response
9	30	A1	Complexity: = E, easy = M, moderate = H, hard = blank, no response
10	31-33	3I1	Type of software from ENCODE.HDR: = 1, I/O processing = 2, algorithmic = 3, logic control = 4, systems related = 5, data/COMMON block = 6, other = blank, no response
11	34-36	I3	Percentage of assignment statements
12	37-39	I3	Percentage of control statements

Item	Location	Format	Description
13	40-42	I3	Percentage of other statements
14	43-47	I5	Number of statements without comments
15	48-52	I5	Number of statements with comments
16	53-57	I5	Number of machine bytes
17	58	A1	Independent of other software: = Y, yes = N, no = blank, no response
18	59	I1	Relation to other software (if dependent) from ENCODE.HDR: = 1, inserted at lower level = 2, new driver or interface = 3, redesign existing components = 4, rename existing components = 5, regroup existing material = 6, other = blank, no response
19	60-63	4I1	Type of addition (up to 4 responses, from ENCODE.HDR): = 1, error correction = 2, planned enhancement = 3, implement requirement change = 4, improve clarity = 5, improve user service = 6, develop utility only = 7, optimization = 8, adapt to environmental change = 9, other = blank, no response
20	64-65	I2	Number of components called
21	66	A1	Not used
22	67-68	I2	Number calling this component
23	69	A1	Not used

Item	Location	Format	Description
24	70-71	I2	Number of shared components
25	72	A1	Not used
26	73-74	I2	Number of components descending
27	75	A1	Not used
28	76-77	I2	Primary language used from ENCODE.HDR: = 1, FORTRAN = 2, ASSEMBLY = blank, no response
29	78-80	I3	Percentage primary language
30	81-82	I2	Secondary language used from ENCODE.HDR: = 1, FORTRAN = 2, ASSEMBLY = blank, no response
31	83-85	I3	Percentage secondary language
32			LEVEL OF DESIGN DETAIL for forms of design (up to 2 responses, from ENCODE.HDR): = 1, component = 2, subcomponent = 3, basic block segment = 4, statement = 5, other = blank, no response
	86-87	2I1	Functional
	88-89	2I1	Procedural
	90-91	2I1	English
	92-93	2I1	Formal
	94-95	2I1	Other design form
33			CONSTRAINT: = X, yes = blank, no (First: Constraint present Second: Component meets constraint)
	96-97	2A1	Memory space

Item	Location	Format	Description
	98-99	2A1	Execution time
	100-101	2A1	Other
34	102-104	I3	Number of design computer runs
35	105-107	I3	Number of code computer runs
36	108-110	I3	Number of test computer runs
37	111-113	F3.1	Computer time for design runs (in tenths of minutes)
38	114-116	F3.1	Computer time for code runs (in tenths of minutes)
39	117-119	F3.1	Computer time for test runs (in tenths of minutes)
40	120-122	F3.1	Effort for design (in tenths of hours)
41	123-125	F3.1	Effort for code (in tenths of hours)
42	126-128	F3.1	Effort for test (in tenths of hours)
43	129-134	I6	Estimated design end date (YYMMDD)
44	135-140	I6	Estimated code end date (YYMMDD)
45	141-146	A1	Estimated test end date (YYMMDD)
46	147	A1	Description comment flag: = Y, yes = N, no
47	148-162	5I3	Components called (up to 5 codes from CIF)
48	163-177	5I3	Calling components (up to 5 codes from CIF)
49	178-192	5I3	Shared components (up to 5 codes from CIF)
50	193-207	5I3	Components affected by re-organization (from Section F, up to 5 codes, from CIF)
51	208-227	A20	Name of other form of design
52	228-247	A20	Constraint other name

Item	Location	Format	Description
53	248	A1	Useful items comment flag: = Y, yes = N, no
54	249	A1	Additional comment flag: = Y, yes = N, no
55	250	I1	Status flag: = 1, unchecked = 2, hand checked = 3, verified by application

Primary key: Form number (bytes 1 through 6)

Secondary key: Component code (bytes 26 through 28)

A.11 GENERAL PROJECT SUMMARY (GPS) FILE

Format has not been defined.

A.12 RESOURCE SUMMARY FORM (RSF) FILE

Item	Location	Format	Description
1	1-6	A6	Form number (e.g., C00633)
2	7-8	I2	Sequence number
3	9-10	I2	Project code from ENCODE.HDR
4	11	A1	Resource type: = M, manpower (technical staff and management work hours) = C, computer (computer usage hours) = O, other (support personnel work hours)
5	12-16	I5	Resource code from ENCODE.HDR (programmer code, computer code, or service code)
6	17-22	I6	Form date (YYMMDD)
7	23-25	I3	Percentage of hours that are management
8	26-31	I6	Beginning date of data (YYMMDD)
9	32-108	11(I3, F4.1)	Resources; number of runs followed by number of hours (in tenths of hours)
10	109	I1	Status flag: = 1, unchecked = 2, hand checked = 3, verified by application
11	110	A1	Source of Data (Phase) flag: = R, requirements team = D, development team = M, maintenance team
12	111-115	A5	Spares

Primary key: Form and sequence number (bytes 1 through 8)

A.13 RUN ANALYSIS FORM (RAF) FILE

Item	Location	Format	Description
1	1-6	A6	Form number (e.g., J00633)
2	7-8	I2	Sequence number
3	9-10	I2	Project code from ENCODE.HDR
4	11-15	I5	Programmer code from ENCODE.HDR
5	16-21	I6	Date of run (YYMMDD)
6	22-23	I2	Computer code from ENCODE.HDR: = 1, any IBM S/360 = 2, any PDP = 3, IBM S/360-75 = 4, IBM S/360-75(C1) = 5, IBM S/360-91 = 6, IBM S/360-95 = 7, PDP-11/70
7	24	A1	Interactive flag: = X, interactive = blank, not interactive
8	25-28	I1	Run purpose from ENCODE.HDR: = 1, unit test = 2, system test = 3, benchmark test = 4, maintenance or utility = 5, compile, assembly, or link = 6, debug run = 7, other = blank, no response
9	29-30	I2	Number of components
10	31-45	5I3	Component codes from CIF
11	46	A1	First-run indicator: = X, first run = blank, not first run

Item	Location	Format	Description
12	47	A1	Run met objectives: = Y, yes = N, no = blank, no response
13	48-51	4A1	Run results (up to 4 responses, from ENCODE.HDR): = 1, good run = 2, submit error = 3, JCL error = 4, other setup error = 5, hardware error = 6, software error = 7, compile error = 8, link error = 9, execution error = A, user-generated message = B, ran to completion = blank, no response
14	52	A1	Comment indicator: = Y, yes = N, no
15	53	I1	Status flag: = 1, unchecked = 2, hand checked = 3, verified by application

Primary key: Form and sequence number (bytes 1 through 8)

A.14 ACCOUNTING INFORMATION (ACC) FILE

Each record contains totals for a particular 4-hour block of wallclock time.

Item	Location	Format	Description
1	1-2	I2	Project code from ENCODE.HDR
2	3-8	I6	Date (YYMMDD)
3	9-10	I2	Time block (4-hour block) 0, 4, 8, 12, 16, or 20 hours from start of day
4	11-13	I3	TSO foreground computer runs
5	14-16	I3	TSO background computer runs
6	17-19	I3	RJE computer runs
7	20-22	I3	Card reader computer runs
			PRIMARY COMPUTER
8	[illegible]	I1	Computer code from ENCODE.HDR
9	24-28	F5.3	Total CPU time (in thousandths of hours)
10	29-33	F5.3	Total I/O time (in thousandths of hours)
11	34-36	I3	Total number of computer runs
12	37-39	I3	Number of runs excluding condition code 0000 or S00C
			SECONDARY COMPUTER
13	40	I1	Computer code from ENCODE.HDR
14	41-45	F5.3	Total CPU time (in thousandths of hours)

Item	Location	Format	Description
15	46-50	F5.3	Total I/O time (in thousandths of hours)
16	51-53	I3	Number of computer runs
17	54-56	I3	Number or computer runs excluding condition code 0000 or S00C
18	57	I1	Status flag: = 1, unchecked = 2, hand checked = 3, verified by application
19	58-67	A10	Spares

Primary key: Date and time block (bytes 3 through 10)

A.15 COMMENT (CMT) FILE

Item	Location	Format	Description
1	1-6	A6	Form number (e.g., D00633)
2	7-8	I2	Sequence number
3	9	A1	Comment type: = C, comment = D, description = R, reason = U, useful item
4	10	I1	Record continuation number of this comment
5	11-12	I2	Project code from ENCODE.HDR
6	13	A1	Comment is continued: = Y, yes = N, no
7	14-103	A90	Text
8	104	I1	Status flag: = 1, unchecked = 2, hand checked = 3, verified by application

Primary key: Form number + sequence number + comment type + record number (bytes 1 through 10)

A.16 COMPONENT INFORMATION FILE (CIF)

Item	Location	Format	Description
1	1-2	I2	Project code from ENCODE.HDR
2	3-10	A8	Component name (e.g., TPNAML)
3	11-13	I3	Component code
4	14-15	I2	PANVALET level number
5	16-17	I2	Module function from ENCODE.HDR
6	18-19	I2	Subsystem function from ENCODE.HDR
7	20	I1	Origin from ENCODE.HDR: = 1, new code = 2, extensively modified old code = 3, slightly modified old code = 4, exact copy of old code
8	21-24	I4	Number of executable source code statements
9	25-28	I4	Number of lines of code with comments
10	29-31	I4	Number of comment lines
11	32-34	I3	Number of unique operators (operators and operands are Halstead's measures (Reference 4))
12	35-37	I3	Number of unique operands
13	38-41	I4	Total number of operators
14	42-45	I4	Total number of operands
15	46-48	I3	Number of input and output variables from module
16	49-51	I3	Number of decisions (McCabe's measure (Reference 8))
17	52-54	I3	Number of FUNCTION references
18	55-57	I3	Number of I/O statements

Item	Location	Format	Description
19	58-60	I3	Number of assignment statements
20	61-63	I3	Number of subroutine CALL statements
21	64-66	I3	Number of FORMAT statements
22	67	I1	Status flag: = 1, unchecked = 2, hand checked = 3, verified by application
23	68-80	A13	Spares

Primary key: Component name (bytes 3 through 10)

Secondary key: Component name prefix (bytes 3 and 4)

Tertiary key: Component code (bytes 11 through 13)

A.17 GROWTH HISTORY (HIS) FILE

Item	Location	Format	Description
1	1-2	I2	Project code from ENCODE.HDR
2	3-8	I6	Date (YYMMDD)
3	9-14	I6	Number of lines of code with comments to date
4	15-17	I3	Number of modules to date
5	18-23	I6	Number of changes to date
6	24	I1	Status flag: = 1, unchecked = 2, hand checked = 3, verified by application
7	25-29	A5	Spares

Primary key: Date (bytes 3 through 8)

A.18 SOURCE ANALYZER PROGRAM (SAP) OUTPUT FILE

This is a single sequential file.

Item	Location	Format	Description
1	1-8	A8	Project name (e.g., MAGBIAS)
2	9-16	A8	Module name
3	17-19	I3	Number of parameters passed in calling sequence
4	20-22	I3	Number of comment lines
5	23-26	I4	Number of executable statements
6	27-28	I2	Number of I/O statements
7	29-32	I4	Number of lines with comments
8	33-35	I3	Number of unique operators
9	36-38	I3	Number of unique operands
10	39-42	I4	Total number of operators
11	43-46	I4	Total number of operands
12	47-49	I3	Number of IF and .IF statements
13	50-52	I3	Number of decisions
14	53-55	I3	Number of input and output variables to module
15	56-58	I3	Number of COMMON area variables
16	59-60	I2	Number of DO and DOWHILE statements
17	61-63	I3	Number of FUNCTION references
18	64-66	I3	Number of structured statements
19	67-69	I3	Number of variables passed out
20	70-72	I3	Number of assignment statements
21	73-75	I3	Number of subroutine CALL statements
22	76-78	I3	Number of FORMAT statements

A.19 TRANSACTION FILES

The Transaction Files are sequential disk backup files that contain records of all updates made to the corresponding data base files, as follows:

Transaction File	Corresponding Data Base File
TRANS.CRF	Change Report Form Files
TRANS.CSR	Component Status Report Files
TRANS.CSF	Component Summary Form Files
TRANS.RSF	Resource Summary Form Files
TRANS.RAF	Run Analysis Form Files
TRANS.CIF	Component Information Files
TRANS.HIS	Growth History Files

Each file has a format similar to its corresponding data base file. The first byte indicates whether the record has been added, changed, or deleted (A, C, or D). Bytes 2 through 7 contain the date (YYMMDD) the record was accessed. Bytes 8 through 13 are spares. Bytes 14 through the end of the record contain the record as stored on the corresponding data base file.

For example, the CRF Files have a record length of 101 bytes. The CRF Transaction File has a record length of 101 + 13 = 114 bytes. All additions, changes, and deletions of records on any of the CRF Files by DBAM are recorded on a single CRF Transaction File, which has the same record format except that byte 1 will be an A, C, or D; bytes 2 through 7 contain the date; and bytes 8 through 13 are blank.

APPENDIX B - SAMPLE DATA COLLECTION FORMS

The forms reproduced here are used by the SEL at the Goddard Space Flight Center to collect data on development projects. The terms used in these forms are defined in Section B.2.

B.1 SAMPLE DATA COLLECTION FORMS AND INSTRUCTIONS

This section contains sample data collection forms and instructions for their use. The instructions precede the forms. The following forms are included:

1. General Project Summary (GPS)
2. Resource Summary Form (RSF)
3. Component Summary Form (CSF)
4. Component Status Report (CSR)
5. Run Analysis Form (RAF)
6. Change Report Form (CRF) and Attitude Maintenance Change Report (ATM)

INSTRUCTIONS FOR COMPLETING THE GENERAL PROJECT SUMMARY - FORM 580-1 (2/77)

This form is used to classify the project and will be used in conjunction with the other reporting forms to measure the estimated versus actual development progress. It should be filled out by the project manager at the beginning of the project, at each major milestone, and at the end. Numbers and dates used at the initiation of the project are assumed to be estimated; intermediate reports should change estimates to actuals (if known) and update estimates. The final report should accurately describe the system development life cycle.

A. PROJECT DESCRIPTION

Description. Give an overview of the project.

Inputs. Specifications and requirements (etc.) of project. Give the format of these.

Requirements. How requirements are established and changed.

Products Developed. List all items developed for the project (e.g., operational system, testing system, simulator, etc.).

Products Delivered. List all items required to be delivered (e.g., source of the operational system, object code of the operational system, design documents, etc.).

B. RESOURCES

Target Computer System. System for which software was developed.

Development Computer System. System on which software was developed.

Constraints. List any size or time constraints for the finished product. Do you anticipate any problems in meeting these constraints?

Useful Items From Similar Projects:

1. List previous projects, which will contribute various aspects to this project.
2. For each project, give the percent of the current project it makes up in each of the 3 listed aspects.
3. For each of the 3 listed aspects (specification, design, code) check what level of modifications are necessary.

C. TIME

Start Date. First date of work, including design and modification of the specifications.

End Date. Delivery date.

Estimated Lifetime. Estimate the operational life of the system.

Mission Date. Scheduled operation date of the system (write unknown if not known or undecided yet on any of these dates). Date project must be operational.

Confidence Level. Give the percent probability you think the end date is realistic. (e.g., 100% means certain delivery on that date. 0% means no chance of delivery.)

D. COST

Cost. Total amount of money the project costs, including both contract and in-house costs.

Maximum Available. Maximum amount available, independent of what estimated cost is.

Confidence Level. Rate percent reliability in cost estimate.

How Determined. At initiation how is it estimated, at completion how is it calculated.

Personnel. Give the number of full time equivalent persons required at inception of the project, 1/3 of the way into the project, 2/3 of the way into the project, at the completion of the project.

Total Person Months. Give the total number of months that full time equivalent personnel (managers, designers, programmers, keypunchers, editors, secretaries, etc.) are assigned to the project. Do not include all overhead items such as vacation and sick leave.

Computer Time. Give the total number of hours on all systems normalized to one machine (e.g., the IBM 360/75) and name the machine.

E. SIZE

Size of the System. Include the total amount of machine space needed for all instructions generated on the project plus the space for data, library routines (e.g., FORTRAN I O package) and other code already available. Break down size into data space and instruction space.

Confidence Level. Rate percent reliability in size estimates.

Total Number of Source Statements. Give the number of FORTRAN, ALC, or any other language instructions generated specifically for this project.

Structure of System. Give overall structure of system. Is it a single load module, is it an overlay structure, or is it a set of independent programs? For overlay and separate programs, give the number and average size of each.

Define Your Concept of a Module. Give the criteria you are using to divide the software into modules.

Estimated Number of Modules. Include only the number of new modules to be written.

Range in Module Size. Give the number of instructions in the minimum, maximum and average module and the language in which they are written as a reference.

Number of Different I/O Formats Used. Give the number of distinct external data sets that are required for the system including card reader, printer, graphics device, and temporary files.

F. COMPUTER ACCESS

A librarian is a person who can be used to perform any of the clerical functions associated with programming, including those given on the chart. Check the appropriate boxes for those persons who have access to the computer to perform the given functions. Give the percentage of time spent by each in batch and interactive access to the computer.

G. TECHNIQUES EMPLOYED

For "level," specify to what level of detail in the finished project the technique is used. (e.g., subroutine, module, segments of 1000 lines, top level, etc.)

Specifications

Functional - Components are described as a set of functions, each component performing a certain action.

Procedural - Components are specified in some algorithmic manner (e.g., using a PDL).

English - Components are specified using an English Language prose statement of the problem.

Formal - Some other formal system is used to specify the components.

Design and Development

Top Down - The implementation of the system one level at a time, with the current level and expansion of the yet to be defined subroutines at the previous higher level.

Bottom Up - The implementation of the system starting with the lowest level routines and proceeding one level at a time to the higher level routines.

Iterative Enhancement - The implementation of successive implementations, each producing a usable subset of the final product until the entire system is fully developed.

Hardest First - The implementation of the most difficult aspects of the system first.

Other - Describe the strategy used if it is not a combination of any of the above.

None Specified - No particular strategy has been specified.

Coding. The final encoding of the implementation in an executable programming language.

Structured Code With Simulated Constructs - The language does not support structured control structures (e.g., FORTRAN) but they are simulated with the existing structures; please state the structured control structures you are using (e.g., WHILE, CASE, IF).

Structured Control Constructs - The language supports structured control structures (e.g., a FORTRAN preprocessor) please list structures you are using.

Other Standard - Describe any other standard you are using.

None Specified - No particular strategy has been specified.

Validation/Verification. Testing: execution of the system, via a set of test cases.

Top Down - Stubs or dummy procedures are written to handle the yet to be implemented aspects of the system and testing begins with the top level routines and proceeds as new levels are added to the system.

Bottom Up - Check out of a module at a time using test drivers and starting at the bottom level modules first.

Structure Driven - Using structure of program to determine test data (e.g., every statement of program executed at least once).

Specification Driven - Using specifications of program to determine test data (e.g., all input output relationships hold for a set of test data).

Other - Describe any other strategy you are using.

None Specified - No testing strategy has been specified.

Validation/Verification. Inspection: visual examination of the code or design.

Code Reading - Visual inspection of the code or design by other programmers.

Walk Throughs - Formal meeting sessions for the review of code and design by the various members of the project, for technical rather than management purposes.

Proofs - Formal proofs of the design or code; please specify the techniques used, e.g., axiomatic, predicate transforms, functional, etc.

None Specified - No inspection techniques have been specified.

There is some space given to permit the further explanation of any of the strategies that may be used.

H. FORMAL NOTATIONS USED AT VARIOUS LEVELS AND PHASES

Give the phases (e.g., design, implementation, testing, etc.) and levels (subroutine, module, segments of 1000 lines, top level, etc.) at which any type of formalism (flowchart, PDL, etc.) will be used in the development of the system.

I. AUTOMATED TOOLS USED

Name all automated tools used, including automated versions of the formalisms given above and compilers for the programming languages used, and at which phase and at what level they are used. Include any products that may be developed as part of this project (e.g., simulator).

J. ORGANIZATION

Describe how the personnel are subdivided with respect to responsibilities into teams or groups, giving titles, brief job descriptions, the number of people satisfying that title and their names and organizational affiliations if known.

K. STANDARDS

List all standards used, whether they are required or optional, and the title of the document describing the standard.

L. MILESTONES

Give the phase at which management may check on progress of the development of the system (e.g., specification, design, implementation of version 1, etc.). State also the date at which it should take place (at completion of the project), how it is to be determined that the milestone was reached, who will be responsible for reviewing the progress at that point and what the review procedure will be. Also give the resources used since the last milestone. For

size of system give the current size of the system at that milestone. Each milestone has 2 confidence levels, one for time estimates and one for resource expenditures. For estimated future milestone, the first confidence level for the probability of reaching the milestone at that date. The second is for the accuracy of the resources used. For past milestones, the first confidence level is normally 100% (actual date) while the second is an estimate on the accuracy of the accounting system.

M. DOCUMENTATION

For each time of documentation developed, state the type of documentation, its purpose, the date it should be completed, its size and list any tools used in its production. (At the beginning of the project these should be estimates, at the end of the project, they should be accurate figures.)

N. PROBLEMS

Give the three most difficult problems you expect to encounter managing this project. Please be as specific as possible.

O. QUALITY ASSURANCE

To what do you attribute your confidence in the completed system. Be as specific as possible.

GENERAL PROJECT SUMMARY

PROJECT NAME _____ DATE _____

A. PROJECT DESCRIPTION

Description _____

_____

Form of Input _____

Requirements _____

Products Developed _____

_____

Products Delivered _____

_____

B. RESOURCES

Target Computer Systems _____ Development Computer Systems _____

Constraints: Execution Time _____ Size _____

Other _____

Any Problems in Meeting Constraints? _____

_____

Useful Items from Similar Projects:

Project	Specification				Design				Code			
	%	Major	Minor	None	%	Major	Minor	None	%	Major	Minor	None

C. TIME

Start Date _____ End Date _____ Estimated Lifetime _____ Mission Date _____

Confidence Level _____

D. Cost

Cost $ _____ Maximum Available $ _____ Confidence Level _____

How Cost Determined _____

Personnel: Inception _____ 1/3 Way _____ 2/3 Way _____ Completion _____

Total Person Months _____

Other Costs. Computer Time _____ (hrs) Documentation $ _____

Other ( ) _____ Other ( ) _____

E. SIZE

Size of System _____ Words. _____ Data Words _____ Instructions

Maximum Space Available _____ Words. Confidence Level _____

Total Number of Source Statements: FORTRAN _____ ALC _____

Other ( ) _____

Structure of System (Check One):

___ Single Overlay

___ Overlay Structure (Number of Overlays _____ Avg. Size _____)

___ Independent Programs (Number of Programs _____ Avg. Size _____)

Define Your Concept of a Module _____

_____

Number of Modules _____ Range in Module Size: Min. _____ Max. _____ Avg. _____

Number of Different I/O Formats _____

SEL-1 (2/77)

F. COMPUTER ACCESS (Check All That Apply. Who Has Access to What.)

	Librarian	Programmer
Keying in New Source Code		
Keying in Update of Source Code		
Inclusion of Code into System		
Submitting Compilation		
Module Testing		
Integration Testing		
Utility Runs (Tape Backup, Etc.)		

Give Percentage for Types of Access:

	Librarian	Programmer
% Batch		
% Interactive		

G. TECHNIQUES EMPLOYED (Check All That Apply and Give Level at Which Used.)

Specification:	Used	Level		Used	Level
Functional			Procedural		
English			Formal		
Design:					
Top Down			Bottom Up		
Iterative Enhance			Hardest First		
Other:			None Used		
Development:					
Top Down			Bottom Up		
Iterative Enhance			Hardest First		
Other:			None Used		
Coding:					
Simulating Construct			Structured Code		
Other:			None		
Validation/Verification: Testing					
Top Down (Stubs)			Bottom Up (Drivers)		
Other:			Specification Driven		
Structure Driven			None		
Validation/Verification: Inspection					
Code Reading			Walk Through		
Proof:			None		

H. FORMALISMS USED

	Used	Level	Phases
PDL			
HIPO			
Flowcharts			
Timeline Diag. (Tree Ch.)			
HOS			
Functions			
Other:			
Other:			

140-1 (2/77) Continuation

I. AUTOMATED TOOLS USED

Name	Phases in Which Used	Level

J. ORGANIZATION

How are the Personnel Organized: _____

Project Personnel:

Title	Job Description	Number	Names and Affiliations (If Known)

K. STANDARDS

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

Type _____ Optional _____ Required _____
Title of Document _____

SSD-1 (8/77) Continuation

ORIGINAL PAGE IS
OF POOR QUALITY

L. MILESTONES

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

Phase ______ Estimated Date ______ Confidence Level ______
How Determined ______
Reviewers ______
Reporting Procedure ______
Resource Expenditures: Cost ______ Person Months ______ Computer Time ______ hrs. ______
Size of System ______ Confidence Level ______

1681 (2/77) Continuation

M. DOCUMENTATION

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

Type _____ Purpose _____
Estimated Date _____ Estimated Size _____ Tools Used _____

N. PROBLEMS

State the three most difficult problems you expect to encounter in completing the project. (1 = most difficult)

1. _____
_____

2. _____
_____

3. _____
_____

O. QUALITY ASSURANCE

State the three most important aspects of the design, development and testing of the system to which you attribute your confidence in the completed system. (1 = most important)

1. _____
_____

2. _____
_____

3. _____
_____

PERSON FILLING OUT FORM _____

569-1 (2/77) Continuation

INSTRUCTIONS FOR COMPLETING THE RESOURCE SUMMARY

This form keeps track of the project costs on a weekly basis. It should be filled out by the project manager every week of the project duration.

PROJECT. Give project name.

DATE. List date form turned in.

NAME. Name of project manager.

WEEK OF. List date of each successive Friday.

MANPOWER. List all personnel on the project on separate lines. Give the number of hours each spent that week on the project.

% OF MANAGEMENT. Add the % of time this person spent managing the project during this reporting period. A new form should be used if this % changes.

COMPUTER USAGE. List all machines used on the project. For each machine give the number of runs during each week and the amount of computer time used.

OTHER. List any other charges to the project.

RESOURCE SUMMARY

PROJECT _____ DATE _____

NAME _____

WEEK OF:											
MANPOWER (HOURS)											% OF MGMT.
COMPUTER USAGE (NO. RUNS/HOURS CHARGED)											
OTHER CHARGES TO PROJECT											

360-3 (6/78)

INSTRUCTIONS FOR COMPLETING THE COMPONENT SUMMARY

This form is used to keep track of the components of a system. A component is a piece of the system identified by name or common function (e.g., an entry in a tree chart or baseline diagram for the system at any point in time, or a shared section of data such as a COMMON block). With the information on this form combined with the information on the Component Status Report, the structure and status of the system and its development can be monitored.

This form should be filled out for each component at the time that the component is defined, at the time it is completed, and at any point in time when a major modification to the component is made. It should be filled out by the person responsible for the component.

PROJECT. Give project name.

DATE. Give date form filled out.

NAME OF COMPONENT. Give name (up to 8 characters) by which the component will be referred to in other forms.

BRIEF DESCRIPTION. State function of component.

TYPE OF SOFTWARE. Check all classifications that apply. All common blocks are separate components.

STATUS OF COMPONENT. Check whether this is a new component, whether it is a component under development (e.g., a previous component summary has already been submitted), or whether the component is now complete.

A. CODE SPECIFICATIONS. Give the form of design for this component, and tell to what level of detail the specifications are given.

Functional—Components are described as a set of functions, each component performing a certain action.

Procedural—Components are specified in some algorithmic manner (e.g., using a PDL).

English—Components are specified using an English Language prose statement of the problem.

Formal—Some other formal system is used to specify the components.

Relative to the one developing the component, rate the precision of the specifications. Very precise means that no additional analysis on the problem is needed, precise means that only easy or trivial ideas have to be developed, and imprecise means that much work still remains in developing this component and its basic structure.

B. INTERFACES

Give the relative position of this component in the system. Give the number and list the names of all components that call this component, and are called by this component. Also, give the names of any components or other items this component shares with other components (e.g., COMMON blocks, external data). The components directly descended from this component refers to the tree chart of the system. If the interfaces are not yet complete, check "Not Fully Specified".

C. PROGRAMMING LANGUAGES

List languages (or assembly languages) to be used to implement this component. If more than one, list percentages of each (in lines of source code). If there are any constraints on the component (e.g., size, execution time) list them. Also give estimated size of finished component in terms of source statements, (estimate size with comments and without comments) and resulting machine languages (including data areas, but not COMMON blocks).

Useful Items From Similar Projects

1. List previous components and projects which contribute various aspects to this component.
2. For each such component, give the percent of each of the three listed aspects it makes up (e.g., a component may be 50% of design but only 25% of code due to changed interfaces, etc.).
3. For each of the three listed aspects, check what level of modifications are necessary.

D. COMPLEXITY

Rate your belief in the complexity of the implementation. Also approximate the number (by %) of assignment type statements (input statements are included), and control statements (those that alter the flow of control e.g., IF, CALL, GOTO). The sum of these two may not be 100% (e.g., CONTINUE, DIMENSION and REAL statements will not be counted). I/O and declarations should be listed as other.

E. RESOURCES TO IMPLEMENT

For each of the three listed phases (Design, Code, Test), estimate computer runs, time needed, hours to implement, and estimated completion date. If not known, or no estimate can be given, write "unknown".

F. ORIGIN OF COMPONENT

If this component is independent of any other component of the system (e.g., is a low level component which is designed first, or is the root node of the tree chart) then check yes, otherwise check no.

If no is checked, then explain why the component was added. (Usually only one reason will be checked, although more may be checked, if appropriate).

A lower level elaboration of a higher level component means that an existing component was expanded to include new components (e.g., expanding tree chart). List the higher level component time.

Added as a driver or interface means that a calling program was added to call existing components. List these called components.

A redesign of an existing component means that new capabilities were added to an already existing component. Write its name.

A renaming of an older component. Give the old name.

A regrouping of existing material means that several components were redesigned with a new component resulting from this redesign. Give the old component names.

Type of addition. Why was this component added to the system at this time? Check the appropriate reason. (Normally, only one should be checked, although more can be if appropriate.)

G. ADDITIONAL COMMENTS. Add any other comments that will help explain the purpose, design, and complexity of this component.

H. PERSON RESPONSIBLE. Include name of person responsible for implementing component.

I. PERSON FILLING OUT FORM. Give name of person filling out form. This normally is the same name as in H.

COMPONENT SUMMARY

PROJECT _____ DATE _____

NAME OF COMPONENT _____ CREATION DATE _____

BRIEF DESCRIPTION _____

STATUS OF COMPONENT NEW _____ UNDER DEVEL _____ COMPLETED _____

TYPE OF SOFTWARE (Check All That Apply)

- ____ I/O Processing
- ____ Algorithmic
- ____ Logic Control
- ____ Systems Related
- ____ DATA/COMMON Block
- ____ Other

A. CODE SPECIFICATIONS (Check All That Apply)

FORM OF DESIGN	LEVEL OF DETAIL				
	Component	Subcomponent	Basic Block Segment	Stmt	Other
Functional					
Procedural					
English					
Formal					
Other ( )					

Precision of Code Specification Very Precise _____ Precise _____ Imprecise _____

B. INTERFACES

Number Components Called _____ Names _____
_____ Not Fully Specified _____

Number Calling This Component _____ Names _____
_____ Not Fully Specified _____

Number Shared Items _____ Names _____
_____ Not Fully Specified _____

Number of Components Directly Descended from This Component _____ Names _____
_____ Not Fully Specified _____

C. PROGRAMMING LANGUAGES

Languages Used and Percentages _____ ( ) _____ ( )

CONSTRAINT PROBLEM EXPECTED:

	Constraint Present	Component Meets Constraint
Memory Space		
Execution Time		
Other ( )		

Size: Source Statements (Including Comments) _____ Machine Bytes _____
Source Statements (Not Including Comments) _____

Useful Items From Similar Projects

Component	Project	Specification				Design				Code			
		%	Major	Minor	None	%	Major	Minor	None	%	Major	Minor	None

S80-5 (6/78)

D COMPLEXITY

Complexity of Function Easy _____ Moderate _____ Hard _____

_____ % Assignment Statements _____ % Control Statements _____ % Other Statements (e.g., Data Decl., I/O)

E. RESOURCES TO IMPLEMENT

	Runs	Computer Time (min)	Effort (hrs)	Est. Completion Date
Design				
Code				
Test				

F Is this component independent of the existing components? _____ Yes _____ No
If No, describe relation of this component to the existing system:

_____ inserted as a lower level elaboration of higher level components (names) _____
_____ added as a driver or interface for existing components (names) _____
_____ a redesign (to add new capability) of existing components (names) _____
_____ a renaming of existing component (name) _____
_____ regrouping of existing material from several components (names) _____
_____ other _____

Type of Addition:

_____ error correction
_____ planned enhancement
_____ implementation of requirements change
_____ improvement of clarity, maintainability, or documentation
_____ other (explain below)
_____ improvement of user service
_____ utility for development purposes only
_____ optimization of time/space/accuracy
_____ adaptation to environment change

G. ADDITIONAL COMMENTS

H. PERSON RESPONSIBLE FOR IMPLEMENTING COMPONENT _____

I. PERSON FILLING OUT FORM _____

SEL-5 (6/76)

INSTRUCTIONS FOR COMPLETING THE COMPONENT STATUS REPORT

This form is to be used to accurately keep track of the development of each component in the system. A Component Summary Report should exist for each component mentioned. The form is to be turned in at the end of each week. Please fill out either daily or once each week. If daily, then a given component may be listed several times during the course of a week. For each component list the number of hours spent on each of the listed activities. This form should be filled out by persons working on the project.

PROJECT. Name of the project.

PROGRAMMER. Name of programmer.

DATE. Date report turned in. Usually the date of a Friday.

COMPONENT. Name of component. Either a part of the system structure for which there is a component summary form, or one of the following:

JCL. Developing command language instructions.

Overlay. Developing system overlay structure.

User Guide. User's Guide Documentation.

System Description. System Description Documentation.

DESIGN

Create. Writing of a component design.

Read. Reading (by peer) of design to look for errors. (e.g., peer review)

Formal Review. Formal meeting of several individuals for purpose of explaining design. Also include time spent in preparing for review. All those attending review should list components discussed in their own Component Status Report for that week.

CODE/DEVELOPMENT

Code. Writing executable instructions and desk checking program.

Read. Code reading by peer. Similar to Design Read above.

Formal Review. Review of coded components. Similar to Design Review above.

TESTING

Unit. Unit testing. Test run with test data on single module.

Integ. Integration testing of several components.

Review. Review of testing status.

OTHER. Any other aspect related to a component of the project not already covered other than Design, Code Development, Test (e.g., Documentation of a specific component). List type of activity, and hours spent on that activity. A set of activities has been listed for which time may be charged to the overall project:

Travel. Time spent on official travel related to this project, (including trips to and from GSFC).

Forms. Time spent on filling out reporting forms.

Meetings. Time spent in meetings which are not design or code review meetings.

Training. Training activities identified for project.

Acc Test. Acceptance Testing activities.

COMPONENT STATUS REPORT

PROJECT _____ DATE _____

PROGRAMMER _____

COMPONENT	DESIGN			CODE DEVELOPMENT			TEST			OTHER	
	CREATE	READ	FORMAL REVIEW	CODE	READ	FORMAL REVIEW	UNIT	INTEG	REVIEW	ACTIVITY	HRS
										Travel	
										Paperwork	
										Meetings	
										Adm Task	
										Training	
JCL											
Overlay											
User Guide											
System Docs											

INSTRUCTIONS FOR COMPLETING THE COMPUTER PROGRAM RUN ANALYSIS FORM

This form will be used to monitor the activities for which the computer is used in the course of a project life cycle. An entry should be made for each computer run-including all activities performed when the computer is used in an interactive mode.

PROGRAMMER. Write down name of person preparing computer runs. This may not necessarily be the person running the program (i.e., librarian).

PROJECT. Write down project name. Use a different form for each project.

COMPUTER. Indicate the machine on which these runs were made (e.g., S/360, PDP-11, Σ5).

DATE. Date form turned in.

JOB ID. Identification of job.

RUN DATE. Date run submitted in format MM-DD (month-day).

INTERACTIVE. Place an X if the run was submitted from an interactive terminal.

RUN PURPOSE. Place an X in all boxes that describe this run.

Unit Test. A purpose of the run is to test one or more components without the rest of the system being configured into the load module. A run which uses a 'test driver' would fall into this category.

System Test. This run executes a load module which contains all of the currently available system in order to test one or more components in a full system configuration.

Benchmark Test. This is a recertification type run. A run that has successfully executed in the past is now rerun to verify that certain capabilities still exist.

Maintenance/Utility. A purpose of this run is to perform a 'library-type' function. Examples are runs that update source, create backups, delete/compress/copy data sets.

Compile/Assembly/Link. A purpose of the run is to check for errors in the compile, assembly and/or link steps. A run which includes one or more of these steps simply as a prerequisite to a system execution would not fall into this category.

Debug Run. This run was submitted in order to investigate a known error.

Other. This run has a purpose which does not fall into one of the other categories. Examples are runs which access other systems in order to aid in the design, development and/or testing of the project under study.

COMPONENTS OF INTEREST. List all components important to this run (e.g., components being tested, compiled, copied, etc.)

FIRST RUN. Place an X here if this is the first time any of the listed components have been processed by the computer for the purpose of run specified.

MEETS OBJECTIVES. This is a subjective evaluation of whether the run satisfied your objectives. Runs that terminate in errors may be satisfactory if the objective was to locate errors or to test for correctness; runs that terminate normally may be unsatisfactory if the purpose was to locate an error known to be present. Thus this question is independent of whether the program contained any errors or not.

RUN RESULTS. Check the box that best describes the results of this run. Normally only one box is checked, although more than one may be checked if appropriate.

Good Run. Program ran to termination with no known errors.

Setup Error. Error in creating program deck.

Submit Error. Deck submitted incorrectly, resources unavailable, keypunch error, or general submission error.

JCL Error. JCL statement incorrect. (JCL cards mistyped should be listed under submit errors.)

Other Setup Error. Such as insufficient space or time specified for job step. This should not be caused by program error.

Machine Error. Errors outside of the control of the programmer.

Hardware Error. Machine malfunction.

Software Error. System crash or system program error (e.g., error in FORTRAN compiler).

Program Error. Error caused by the submitted program.

Compile Error. The source program contains an error which is found by the compiler or assembler.

Link Error. The loader or linkage editor finds an error.

Execute Error. System error messages are generated during the execution step, possibly causing an abend.

User Generated Error. The program terminates in a programmer generated error message which is not a system error.

Ran to Completion. The program terminated with no error message; however, the results are incorrect signifying that there is something wrong with the program.

COMMENTS. If you believe that your answers to these questions do not adequately characterize this run, you may add any additional comments that you wish. Also use this space to indicate if the run was lost before you had a chance to evaluate results.

COMPUTER PROGRAM RUN ANALYSIS

PROGRAMMER _____ PROJECT _____ COMPUTER _____ DATE ___/___/___

JOB ID	DATE MM DD	INTERACTIVE	RUN PURPOSE							COMPONENTS OF INTEREST				RUN RESULTS										COMMENT (e.g., Run Late, No Results)
			Unit Test	System Test	Benchmark Test	Maintenance/Utility	Compile/Assembly/Link	Debug Run	Other		First Run	Run Met Objectives	Run Did Not Met Objectives	Good Run	Setup Error			Machine Error		Program Error					
														Good Run	Submit Error	JCL Error	Other Setup Error	Hardware Error	Software Error	Compile Error	Link Error	Execute Error	User Generated Msg	Ran to Completion	

500 7 (6/78)

INSTRUCTIONS FOR COMPLETING THE CHANGE REPORT FORM

This form is used to keep track of all changes made to a system. A change is any alteration to the design, documentation, or code generated for a project. Each change can be thought of as a step in the process of transforming the original software design into a complete working system. The initial creation of sections of fresh code or design is not a change.

One change report form should be filled out for each change. Where several changes are made simultaneously for different reasons a separate form should be completed for each reason.

NUMBER. A unique identifier per form per day consisting of initials followed by a sequence number. The initials should be those of the person filling out the form. The sequence number should be a positive integer indicating the number of forms filled out so far during the day. Number DMW01 indicates the first form of the day filled out by DMW, DMW02 is the second form that day, etc.

PROJECT NAME. The name of the development project.

CURRENT DATE. The date on which an entry is first made on the form, even if the form is not completed on that day.

SECTION A—IDENTIFICATION

REASON. Explain why the change is being made.

DESCRIPTION. Describe the change that is being made. This should not be on the variable name or bit level, but should be sufficiently abstract so that the function of the changed code can be determined, e.g., "the input buffer was cleared," rather than "array buff was set to zero."

EFFECT. What components (or documents) are changed? List the names of all components and documents modified as part of the change, including version numbers.

EFFORT: What additional components (or documents) were examined in determining what change was needed? List all components and documents that were examined, but were not actually changed, in deciding what change to make, how to make it, and where to make it. This list should not overlap with the list of components and documents actually changed.

DATES OF CHANGE. Need for change determined on. Give the date on which it was first realized that a change was needed.

Change started on. Give the date on which the change was started.

What was the effort in person-time required to understand and implement the change?

Give the best available estimate of the total time needed to understand what change had to be made and how to make it, including the implementation time. This should include the time of all persons involved in making the change. As an example, if two people each worked 6 hours on the change, the space marked "one day to 3 days" should be checked.

SECTION B—TYPE OF CHANGE

Check the one box that best describes the change. If none of the change descriptions seem to fit, check other and give a detailed description of the change in Section E. If several of the descriptions seem equally appropriate, more than one box may be checked.

Error Correction. A change made to correct an error in previous work. If this box is checked Sections C and D of the change report form should be completed.

Planned Enhancement. The insertion of a body of code into a program stub that was initially created as a dummy for testing purposes, or adding capability to an already existing component as part of a planned incremental development.

Implementation of Requirements Change. Altering the system to conform to a change in requirements imposed by the customer.

Improvement of Clarity, Maintainability, or Documentation. Changes made to improve code quality, such as improving indentation of code, resequencing labels for readability, adding or updating documentation or correcting literary errors in it, suppressing redundant information or replacing multiply-occurring sections of code with procedure calls. Corrections of violations of programming standards, and design improvements that should have been visible in the functional specifications of components of the system are to be treated as error corrections. Documentation updates made concomitantly with a change should be treated as a part of that change and classified with the primary cause of the change.

Improvement of User Services. During system development, individual programmers may find that with very little extra work they can provide the user with additional facilities on top of the functional requirements of the system. Such changes are classed as improvements to user services.

Insertion/Deletion of Debug Code. Changes made to the program text specifically to provide additional information during test runs so that errors can be isolated.

Optimize Time/Space/Accuracy. An optimization is a localized adjustment of the program whose main purpose is to reduce its execution time or memory requirements, or to obtain results of greater numerical accuracy by tuning the algorithms used to the specific problem being solved.

Adaptation to Environment Change. The "boundary" of a software system is defined to include just those programs whose development and maintenance is being monitored as part of the software engineering laboratory project. A change whose cause lies outside this boundary (e.g., in response to an operating system, compiler, or hardware change) is regarded as environmentally caused.

Was more than one component affected by the change? A component is defined to be directly involved in a change if it contains subroutines that are changed and it contains no subcomponents containing those subroutines. Check yes if the change directly involves more than one component of the system, no otherwise. It may be the case that a change to one subroutine/component will require some future adjustment in other components (these components may not even have been coded yet, or their adaptation may be postponed). In such cases, the effects of the change involve more than one component even though only one module was noted as changed on this form.

SECTION C—TYPE OF ERROR

Check the one box that best describes the error. If none of the error descriptions seem to fit, check other and give a detailed description of the error in Section E.

Requirements Incorrect or Misinterpreted. Requirements may be incorrect (inconsistent or ambiguous), or their meaning may be misinterpreted. In either case, an error of this type, if undetected early, may propagate through design and into code. Even undetected until acceptance testing (or maintenance), errors resulting from incorrect or misinterpreted requirements should be classified in the requirements error category.

Functional Specifications Incorrect or Misinterpreted. Functional specifications are taken to be a specification of a component as a set of functions defining the output for any input. Similar to requirements, specifications may be either incorrect or misinterpreted. Errors in the specifications that occur as a result of misunderstandings of requirements are classified as misinterpreted requirements errors and not incorrect specifications. Specification errors that result from misunderstandings among those writing the specifications are classified as incorrect specifications. Errors in code or design or documents resulting from incorrect or misinterpreted specifications should be classified in the specifications error category.

Design Error Involving Several Components. A design decision is a choice of organization of a component into subcomponents, including the specification of the interfaces among the subcomponents. A design error is a design decision that results in one of the following:

- interfaces that contain insufficient, unnecessary, or redundant information;
- a set of subcomponents that do not satisfy the specifications of the component (i.e., one or more of the subcomponents do not have the capabilities needed to satisfy the use intended for the component).

Note that a design error may result from incorrect or misinterpreted requirements or specifications. In such cases, the error should not be classified as a design error, but as a requirements or specification error.

Error in the Design or Implementation of a Single Component. Most simple, localized programming mistakes fall into this category. It contains those cases where the organization of the system into components and their interfaces is correct, but a particular component does not behave according to its intended use (i.e., does not correspond to its specification). This may occur because the algorithm used in designing the component is incorrect, or because the implementation of the algorithm is incorrect. If the algorithm has a written specification prior to code generation, and the specification is incorrect or misinterpreted, the error is not classified as a design or implementation error, but as a specification error. If the erroneous algorithm has no written specification, or if the implementation of the algorithm has errors not attributable to any other category, then the error is classified as an error in the design or implementation of a single component.

Misunderstanding of External Environment, Except Language. Check this box if the error resulted from mistaken assumptions about the hardware or software environment in which the program operates (i.e., that software outside the "boundary" of the project—see "adaptation to environment change" in Section B). Included here are mistaken assumptions about how the operating system works, about how the hardware is controlled, about response of peripherals to various commands, about the operation of the library system, about the interface to special display hardware or software, etc.

Error in Use of Programming Language/Compiler. Errors in the use of the language/compiler are those errors that result from some misunderstanding of how the compiler works, how the language provided run-time support system operates, or some misunderstanding of particular language features. Not included in this category are clerical errors (e.g., typos) that lead to compilation errors.

Clerical Error. Clerical errors are those errors that occur in the mechanical translation of an item from one format to another (e.g., one coding sheet to another), or from one medium to another (e.g., coding sheets to cards). No interpretation or semantic translation is involved in such a process.

FOR DESIGN OR IMPLEMENTATION ERRORS ONLY

This section should be filled out only if the error was a design error, involving several components, or if it was an error in the design or implementation of a single component. Errors that occur in the design of a system, subsystem, set of components, or single component, or in the implementation of a single component, may be categorized in one of two ways. Either there was an error in the use of data, or there was an error in the function of a component (such as an algorithmic or computational error resulting in program behavior not corresponding to the intended use of the program). Data use errors can be characterized as either incorrect values for data items or improper assumptions about the structure of data items (e.g., array size or dimensions, or ordering of items in a list). Errors involving the function of a component include control and computational errors, such as incorrect sequencing of statements, omitted statements (where such are not clerical errors), improperly computed expressions, omitted capabilities of the component(s), etc.

SECTION D—VALIDATION AND REPAIR

What were the activities used to validate the program, to detect the error, and find its cause?

The purpose of this section is to discover how it became known that an error existing and how the cause of the error was determined. A check should be put in the first column for each method used for validating the component(s) where the error was found. A check should be put in the second column on the same line as the method by which the symptoms of this particular error was first noted. The third and fourth columns refer to activities used to find the cause of the error, once it was known that the error existed. In the third column, check all techniques used in trying to find the cause of the error. In the fourth column, check those techniques that yielded the information needed to find the cause. In some cases, such as some errors found by code reading, the technique(s) used to find the error and discover its cause will be the same. Note that error messages have been divided into two categories: those produced by the support system (e.g., compiler, operating system), and those designed into the code for the specific purposes of the project. Testing has also been divided into two categories: test runs made prior to acceptance testing (pre-acceptance test runs), and acceptance tests. If activities other than those listed in the table were used in finding the error or discovering its cause, check other in the appropriate column, and describe the activities used in Section E. This table inevitably has some redundancy: a check in column 2 must always have a corresponding check in column 1, similarly with columns 4 and 3.

What was the time used to isolate the cause?

Check the space that most closely approximates the time required to isolate the cause of the error. This should be the total of the time that was spent in the activities tried to find the cause. If the cause of the error was never found, and a workaround was used, check the appropriate box. If the cause was never found and a workaround was not used, explain the circumstances in Section E.

Was this error related to a previous change?

Changes to software may result in errors because of one or more of several reasons:

- the change was incorrectly implemented, i.e., did not conform to its specification;
- the change invalidated an assumption made elsewhere in the software;
- an assumption made about the rest of the software in the design of the change was incorrect.

An error is related to a previous change if it results from one of the above three conditions. Errors that are uncovered by changes, i.e., an error masked by another that is revealed when the latter is corrected, do not belong in this category. If the error is related to a previous change, give the number and date of the change report form of the related change. When did the error enter the system?

Check the box that most closely represents the phase in the erroneous components' development in which the error was introduced.

SECTION E—ADDITIONAL INFORMATION

This section is intended to permit further explanation of any items you feel may be significant in categorizing the change (including error corrections). If the "other" category was checked in any of the previous sections of the form, a fuller explanation should be given here. Do not hesitate to give a full description of the error or change or any doubts you may have in classifying it. The accuracy of our analysis is dependent on the amount and accuracy of the data you provide for us. The study we are performing is an attempt to do a careful, detailed investigation of the processes that go on during software development, the kinds of changes and errors that occur during development, and the reasons for their occurrence. With your help, we hope to gain enough insight into the design, coding, and testing of programs so that proposed techniques for coping with software changes and reducing the number of errors can be evaluated. Your cooperation and patience in completing the change report form each time you make a change to a document or program are needed and appreciated.

NUMBER _____

CHANGE REPORT FORM

PROJECT NAME _____ CURRENT DATE _____

SECTION A - IDENTIFICATION

REASON Why was the change made? _____

DESCRIPTION What change was made? _____

EFFECT What components (or documents) are changed? (Include version) _____

EFFORT What additional components (or documents) were examined in determining what change was needed? _____

	(Month)	(Day)	(Year)
Need for change determined on			
Change started on			

What was the effort in person time required to understand and implement the change?

_____1 hour or less, _____1 hour to 1 day _____1 day to 3 days, _____more than 3 days

SECTION B - TYPE OF CHANGE (How is this change best characterized?)

- ☐ Error correction
- ☐ Planned enhancement
- ☐ Implementation of requirements change
- ☐ Improvement of clarity, maintainability, or documentation
- ☐ Improvement of user services
- ☐ Insertion/deletion of debug code
- ☐ Optimization of time/space/accuracy
- ☐ Adaptation to environment change
- ☐ Other (Explain in E)

Was more than one component affected by the change? Yes _____ No _____

FOR ERROR CORRECTIONS ONLY

SECTION C - TYPE OF ERROR (How is this error best characterized?)

- ☐ Requirements incorrect or misinterpreted
- ☐ Functional specifications incorrect or misinterpreted
- ☐ Design error, involving several components
- ☐ Error in the design or implementation of a single component
- ☐ Misunderstanding of external environment, except language
- ☐ Error in use of programming language/compiler
- ☐ Clerical error
- ☐ Other (Explain in E)

FOR DESIGN OR IMPLEMENTATION ERRORS ONLY

If the error was in design or implementation:

The error was a mistaken assumption about the value or structure of data _____

The error was a mistake in control logic or computation of an expression _____

SEL-81-3 (6/79)

FOR ERROR CORRECTIONS ONLY

SECTION D - VALIDATION AND REPAIR

What activities were used to validate the program, detect the error, and find its cause?

	Activities Used for Program Validation	Activities Successful in Detecting Error Symptoms	Activities Tried to Find Cause	Activities Successful in Finding Cause
Pre-acceptance test runs				
Acceptance testing				
Post-acceptance use				
Inspection of output				
Code reading by programmer				
Code reading by other person				
Talks with other programmers				
Special debug code				
System error messages				
Project specific error messages				
Reading documentation				
Trace				
Dump				
Cross-reference/attribute list				
Proof technique				
Other (Explain in E)				

What was the time used to isolate the cause?

____one hour or less, ____one hour to one day, ____more than one day, ____never found

If never found, was a workaround used?____Yes ____No (Explain in E)

Was this error related to a previous change?

____Yes (Change Report #/Date____) ____No ____Can't tell

When did the error enter the system?

____requirements ____functional specs ____design ____coding and test ____other ____can't tell

SECTION E - ADDITIONAL INFORMATION

Please give any information that may be helpful in categorizing the error or change, and understanding its cause and its ramifications.

Name:____ Authorized:____ Date:____

(4/78)

Current Date _____

_____Attitude System Maintenance Report

Project Name _____Need for Change determined on (Mo., Day, Yr.)_____

Describe Change _____

_____

_____

_____

What components/subroutines/modules are changed _____

_____

_____

CHANGE (NON-ERROR) (fill out this section if change is NOT an error correction)
This change is being made because of a change in: (Check all that apply)

_____ requirements
_____ new information/data
_____ specification
_____ design
_____ other (specify): _____
_____hardware environment
_____software environment
_____optimization

ERROR ONLY (fill out this section if change IS an error correction)
The following activities were used in error detection or isolation: (Check all that apply) (Put D for detection, I for isolation)

_____ normal use
_____ test runs
_____ code reading
_____ reading documentation
_____ other (Specify): _____
_____trace/dump
_____cross reference/attitude list
_____system error messages
_____project specific error messages

Which of the following best describes the error:

_____ requirements error
_____ design error
_____ error in translating design or specification to code
_____ other: Describe_____
_____specification error
_____clerical error

Was this error related to a previous maintenance change _____yes _____no _____can't tell

Please give any information that may be helpful in categorizing and understanding the change on the reverse side of this form.

Person filling out this form _____
Approved _____Date _____

Change started on date (month, day, year) _____

Time spent on this change:
_____ less than 1 day _____1 day to a week _____more than a week

B.2 SEL GLOSSARY OF TERMS USED WITH DATA COLLECTION FORMS

This section defines the terms used in the software engineering data collection forms reproduced in Section B.1. A more extensive glossary (based substantially on this one) is found in Reference 9.

assignment statements	All statements that change the value of a variable as their main purpose (e.g., assignment or READ statements, but the assignment of the DO loop variable in a DO statement should not be included).
attitude/orbit	Any component that is directly related to either the attitude determination (or control) task or to the orbit determination (or control) task falls into this category. This should include full systems in general (such as GTDS or ISEE-B Attitude) as well as specific modules such as Deterministic Attitude or DCCONES.
attribute list	A compiler-generated list of the identifiers used by a program that describes the characteristics of those identifiers and shows the source statements where they are first defined (or first used) and, for variables, their (relative) storage locations.
automated tools	Any programs whose purpose is to aid in software development (e.g., compiler, text editor, or dump or trace facility). This includes compilers but not standard operating system software (e.g., linkage editor).
baseline diagram	A structured chart listing all components in a system in which a connection from a higher component to a lower one indicates that the higher component calls the lower one.
batch	Use of a computer in which the entire job is read into the machine before the processing begins and in which there is no provision for interaction with the submitter during execution of the job. (Interactive usage is always via a terminal; batch usage may be via a terminal or a card deck.)

bottom-up	The design (or implementation) of the system starting with the lowest level routines and proceeding to the higher level routines that use the lower levels.
business/ financial	The second of the four major categories applies to components related to some accounting task, financial data formatting, business data retrieval or reporting, or possibly personnel data management. Very few of the components being studied will fall into this class.
change	A modification to design, code, or documentation. A change might be made to correct an error, to improve system performance, to add capability, to improve appearance, or to implement a requirements change, for example.
clerical	The process of copying an item from one format to another or from one medium to another, which involves no interpretation or semantic translation.
code reading	Visual inspection of the source code by persons other than the creator of the code.
command/ control	This class of components includes those used either to generate vehicle commands or to transmit these commands from the control center.
complexity	Measures the difficulty of implementing a component, independent of the implementer's experience. Easy (or simple) means that any good programmer can write down the correct code with little thought. Hard (or complex) means that much thought is involved in the design. (Compare this with "precise"; e.g., easy and imprecise may mean a vague specification, but once the approach is decided upon, the code is easy to write.)
component	A piece of the system identified by name or common function (e.g., separately compilable function, an entry in a tree chart or baseline diagram for the system at any point in time, or a shared section of data such as a COMMON block).

computer time	For batch usage, this is the billable time for all runs. For interactive usage, it is the number of hours spent at a terminal.
confidence level	Percentage probability that a given number is correct: 100 percent means that the number is absolute certainty; 0 percent means that the number must be incorrect.
constraints	Restrictions on resource availability (execution time, memory allocation) imposed by specifications.
constraints, space	All restrictions caused by space problems. On the Component Summary Report form, list each restriction separately (e.g., maximum number of words that component may occupy at one time or maximum disk space available during execution time or for program storage).
constraints, time	All restrictions caused by various machine and calendar time problems. On the Component Summary Report form, list each restriction separately (e.g., maximum execution time for component to process and respond to some input condition or time to complete a component or milestone).
control statements	All statements that potentially alter the sequence of executed instructions (e.g., GOTO, IF, RETURN, or DO).
correction	A change made to correct an error.
cosmetic	Changes in the source program that have little effect on the performance of program (e.g., correct comments, move code around as long as it does not alter the algorithm implemented, or change the name of a local variable).
create	The creation and recording of the idea.
creation date	Date that the component was first named (e.g., date it first appeared on a tree chart).
cross-reference	List of the identifiers used by a program showing (by means of indices or statement numbers) which statements of the program define and reference those identifiers.

data base applications	This category is to include components that retrieve, write to, or format information for a well-defined formatted bank of information available to the system. The user must decide whether or not the data set is to be considered a data base. An example of an acceptable data base would be the ADL file, SLP file, or Geodetics file, whereas a sequential telemetry file or tape would not be.
design	A description of what the system must do, its components, the interfaces among those components, and the system's interface(s) to the external environment.
design phase	The creation and recording of the design, including discussion about strategy with peers. This phase does not include the development of any code at the programming language level. It does include the creation of specifications for subcomponents of the current component.
design reading	Visual inspection of the design by persons other than the creator of the design.
development phase	The development and recording of code and inline comments based on the design. This phase includes the modification of code caused by design changes or errors found in testing. It does not include any time spent in entering the code into the computer.
documentation	Written material, other than source code statements, that describes a system or any of its components.
dump	Record of the state of the memory space used by a program at some point in its execution. A dump may include all or part of the program's memory space (including registers).
end date	Date that a project is scheduled to be completed.
English (or informal) specifications	Specifications given as readable English text, as opposed to some formal notation.

error	Discrepancy between a specification and its implementation. The specification might be requirements, design specifications, or coding specifications.
external environment	Combination of hardware and software used to maintain and execute the software, including the computer on which the software executes, the operating system for that computer, support libraries, text editors, and compilers.
formal specifications	Some specification technique based upon a strict set of rules for describing the specification and usually involving the use of an unambiguously defined notation (e.g., mathematical functions or formal PDL).
function	Mathematical notation used to specify the set of input, the set of output, and the relationship between input and output.
functional specifications	Specification of a component as a set of functions defining the output for any input. The specification emphasizes what the program is to do rather than how to do it. However, an algorithmic specification can be considered functional if it is not used to dictate the actual algorithm to be used. (See procedural specifications.)
hardest first	Design (or implementation) of the most difficult aspects of the system first.
HIPO (Hierarchical Input Process Output)	Graphical technique that defines each component by its transformation on its input data sets to its output data sets.
implementation	Implementation of a program is either a machine-executable form of the program or a form of the program that can be automatically translated (e.g., by compiler or assembler) into machine-executable form.
integration test	Test of several modules to check that the interfaces are defined correctly.
integration test, full	Test of the entire system (i.e., top-level component).

integration test, partial	Test of any set of modules but not the entire system.
intended use of	Result of invoking a program or segment of a program, including the actions performed by that program when invoked. Invocation may be by subroutine or function call or by a branch to a segment of code.
interface	Set of data passed between two or more programs or segments of programs and the assumptions made by each program about how the others operate.
interactive	Use of a computer via a terminal in which each line of input is immediately processed by the computer.
iterative enhancement	Design (or implementation) of successive versions, each producing a usable subset of the final product until the entire system is fully developed.
level	Unit corresponding to some partitioning of the final product (e.g., a single line of code, 10 lines of code, 25 lines of code, subroutine, or module). If the system is hierarchically structured, each component is at a higher level than its subcomponents, and the system may be described as the highest level component (the component at level 1), the component at level 2, or the lowest level component.
level, lowest	Smallest unit identified by the activity (e.g., code reading to the single statement, top-down design to the module level, or top-down design to level 3).
librarian	A clerk whose responsibilities include processing source statements but not writing them, (e.g., maintaining libraries, updating code, or producing tape backups).
machine words	Number of words in a main memory that a component occupies at one time.
manpower	Sum, over the number of people, of the number of hours per person charged to the contract.

mathematical/ numerical	This category is meant to be a more specific category than the scientific class. It contains those components that reflect a specific algebraic expression or mathematical algorithm. Such components as a dot product routine or a numerical integrator are in this category.
maximum space	Total number of machine words that the system may occupy at one time.
mission date	Date that system must be operational.
module test	Test of a single module.
none used	No explicit technique was specified to be used.
onboard processing	All components that are built for the purpose of satisfying some onboard processing need belong to this class. Although the component may be built and tested on a computer that is not the real flight computer, it should be classified as onboard if the final destination is the OBC (onboard computer).
optimization	Changes in the source code to improve program performance (e.g., run faster or use less space). Optimization changes are not error corrections; however, if a change is made to use less space to conform to the specified space constraint, then the term "error" applies.
PDL	Program design language (often called pseudocode). Used in the design and coding phases of a project, PDL is a language that contains a fixed set of control statements and a formal or informal way of defining and operating on data structures. PDL code may or may not be machine-readable, and for this study it is not considered as documentation, but as an integral part of the finished source program.

procedural specifications	Specification of a component in some algorithmic manner (e.g., using PDL or a flowchart). The specification says how the program is to work. (See functional specifications.)
proof technique	Method for formally demonstrating that a piece of software performs according to its specifications. Proof techniques usually use some form of mathematical notation to describe the result of executing a program.
range in module size	Number of source statements in a module, including comments.
read	The reading by peers of the recordings of the current phase to look for errors, invent tests, and so on.
real-time	This class includes components that are a direct function of events occurring at, or near, the current time. Typical components would be the Attitude Control Monitors. Since parts of most of the telemetry processors are required to process data as it is received, they too may be considered real-time components.
requirements	System specification written by the user to define a system to a developer. The developer uses these specifications in designing, implementing, and testing the system.
review	Formal meeting of several individuals for the purpose of explaining design (management review). Also includes the time spent in preparing for the review. All those attending a review should list the components discussed in their own Component Summary Report for that week.
scientific	A component may be in this category if it is related to some mathematical algorithm, engineering problem, law of physics, or celestial mechanics problem. Most of the full systems developed will fall into this category, whereas the various pieces of modules may fall into some of the other classes.

segment	Contiguous piece of code that is unnamed and, hence, cannot be referred to as a single entity in a program statement. A segment could be one or several lines of a subroutine, part of a data area, or an arbitrary contiguous section of memory.
shared items	Data and programs, accessible by several components, such as COMMON blocks, external files, and library subroutines.
simulating constructs	Statements that are used to simulate structured control structures when the language to be used does not contain structured control structures.
source instructions	See source statements.
source statements	All statements readable by and read by the compiler. This includes executable statements (e.g., assignment, IF, and GO TO); nonexecutable statements (e.g., DIMENSION, REAL, and END); and comments.
specification	Description of the input, output, and essential function(s) to be performed by a component of the system. The specification is produced by the organization that is to develop the system; that is, at the top level, it can be thought of as the contractor's interpretation of the requirements.
specification, imprecise	The input, output, and function of the component are loosely defined. Much of what is required is assumed rather than specified. The specification relies heavily on programmer experience and verbal communication to get an unambiguous interpretation and a full understanding of what is needed.
specification, precise	The input, output, and function of the component are well defined. There are underlying assumptions not specified, but it is assumed that any programmer working on the project, with experience on a similar project, will understand these assumptions. It is possible to arrive at an ambiguous interpretation or misunderstanding

specification, precise (Cont'd)	of the specifications if the reader does not have enough experience with the problem or does not obtain further verbal communication.
specification, very precise	Completely defined description of the input, output, and function of a component. The implementer of a very precise specification need make few, if any, assumptions. It is almost impossible to arrive at an ambiguous interpretation or misunderstanding of the specifications.
specification-driven	Using the specifications of the program to determine test data (e.g., test data is generated by examining the input/output requirements and specifications).
standards	Any specifications that refer to the method of development of the source program itself, and not to the problem to be implemented (e.g., using structured code, at most 100-line subroutines, or all names prefixed with subsystem name).
start date	Date on which initial work on a project began.
string processing	This includes components that perform operations on lists of characters. Normally, this class is assumed to include functions of compilers, hash code string hook-up, and array comparisons.
structure-driven	Using the structure of the program to determine test data (e.g., generating data to ensure that each branch of a program is executed at least once).
structure of data	Organization of a composite data item consisting of several variables or other array items. Examples of such composite data items are arrays (both singly- and multiply-dimensioned), strings, complex variables and constants, records on a disk file (each record containing several words), and multiple-word entries in a table.
structured code	The language supports structured control structures (e.g., a FORTRAN preprocessor).

systems	By system-related software, one includes any package designed to affect, modify, extend, or change the normal available processing procedure of the operating system. This could include such components as error tracing or extended I/O such as DAIO.
system size	Total number of machine words needed for all instructions generated on the project plus space for data, library routines, and other code. This is the total size of the system without using any overlay structure.
table handler	Includes components that are specifically designed to generate or interpret information in a table format such as the Generalized Telemetry Processor.
telemetry/ tracking	Includes all components that are specifically required to interface (either read, write, or format) with telemetry or tracking data.
testing phase	Design of tests, testing strategies, and the running of such tests. This phase does not include the writing of any code (even for debugging purposes), which should be recorded under coding.
top-down	Design (or implementation) of the system, starting with a single component, one level at a time, by expanding each component reference as an algorithm possibly calling other new components.
trace	Record of program execution showing the sequence of subroutine and function calls and, sometimes, the value of selected variables. Code used in producing a trace is automatically inserted into a program, usually by the compiler, sometimes by other support software.
type of software	The four major classifications of most of the applicable software being developed are: scientific, business/financial, systems, and utility. These classifications may be refined into the categories of: string processing, data base applications, real-time, and table

type of software (Cont'd)	handler. A further refinement includes the categories of: attitude/orbit, telemetry/tracking, command/control, mathematical, and numerical onboard.
utility	Any component that is generated to satisfy some general support function required by other applications software may be considered a utility. This class of components usually contains software that does not fit into any of the other three categories. Although components can fall into two of the primary categories (e.g., scientific and utility), it will be easier to use only the more descriptive of the categories (e.g., vector cross-product--scientific; data unpacking--utility).
value of data	The number and kind of number (e.g., integer, floating-point, or ASCII-encoded character) stored in a local variable or data area, parameter, common variable, or system-wide data item.
walkthrough	Formal meeting sessions for the review of source code and design by the various members of the project for technical rather than management purposes. The purpose is for error detection and not correction.
workaround	The method used to counteract the effects of an error in a program when the cause of the error and, consequently, the location of the statements containing the error is not known or is inaccessible (e.g., a compiler error).

APPENDIX C - ABBREVIATIONS

The following are explanations of abbreviations used throughout this document.

ACC	Accounting Information File
ATM	Attitude Maintenance Change Report File
CIF	Component Information File
CMT	Comment File
CRF	Change Report Form
CSC	Computer Sciences Corporation
CSF	Component Summary Form
CSR	Component Status Report
DB1:	Disk DB1
DBAM	Data Base Maintenance Software
DEC	Digital Equipment Corporation
DIR	Subjective Evaluations Directory File
ENC	Encoding Dictionary
GPS	General Project Summary
GSFC	Goddard Space Flight Center
HDR	Phase Dates File
HIPO	Hierarchical Input Processing Output
HIS	Growth History File
JCL	Job Control Language
PDL	Program Design Language
RAF	Run Analysis Form
RJE	Remote Job Entry
RJP	Remote Job Processing
RMS	Record Management System
RMSIAC	RMS Indexed Access routines
RSF	Resource Summary Form
RSX-11M	Current PDP-11/70 Operating System
SAP	FORTRAN Source Analyzer Program
SEF	Subjective Evaluations File
TSO	Timesharing Option (IBM)

UIC	User Identification Code
UM	University of Maryland
YYMMDD	Year-Year-Month-Month-Day-Day date format. For example, 810704 is July 4, 1981.
[n,m]	User Identification Code. For example, [204,1]

APPENDIX D - USER IDENTIFICATION CODE (UIC) LAYOUT

This appendix lists the organization of all production software located under the User Identification Code (UIC) of 204 on disk DB1.

1. [204,1]--All data base files.
2. [204,2]--Not used.
3. [204,3]--Indirect command files for DBAM or tape delivery, plus temporary intermediate files used by DBAM.
4. [204,4]--Indirect files for reports and other utility programs.
5. [204,5]--All task images. Help files associated with each task image.
6. [204,6]--Source code and object modules for all task images except DBAM. Command and overlay files to create task images. Fixed input data files to programs.
7. [204,7]--Utility source code and object modules used by several programs (e.g., generalized open and read routines).
8. [204,10]--DATATRIEVE record and domain-definition indirect files.
9. [204,11]--Profile reports and all reports produced.
10. [204,12]--Tape backup command files.
11. [204,15]--DBAM source code and object modules, plus task generation command files and overlay files.

REFERENCES

1. Software Engineering Laboratory, SEL-78-102, FORTRAN Static Source Code Analyzer (SAP) User's Guide Revision 1, W. Decker and W. Taylor, September 1982

2. --, SEL-81-103, Software Engineering Laboratory (SEL) Data Base Maintenance Software (DBAM) User's Guide and System Description, P. Lo and D. Card, March 1983

3. --, SEL-82-001, Evaluation of Management Measures of Software Development, G. Page, D. Card, and F. McGarry, September 1982

4. Maurice Halstead, Elements of Software Science. New York: Elsevier Publishing Company, 1977

5. Digital Equipment Corporation, AA-L763A-TC, Introduction to RSX-11M and RSX-11M-PLUS, November 1981

6. --, AA-L681A, RSX-11M/M-PLUS Utilities Manual, November 1981

7. --, AA-C742B-TC, DATATRIEVE-11 V2.0 User's Guide, July 1980

8. T. J. McCabe, "A Complexity Measure," IEEE Transactions on Software Engineering, December 1976, vol. 2, no. 4, pp. 308-320

9. Data Analysis Center for Software, GLOS-1, The DACS Glossary, A Bibliography of Software Engineering Terms, October 1979

BIBLIOGRAPHY OF SEL LITERATURE

The technical papers, memorandums, and documents listed in this bibliography are organized into two groups. The first group is composed of documents issued by the Software Engineering Laboratory (SEL) during its research and development activities. The second group includes materials that were published elsewhere but pertain to SEL activities.

SEL-Originated Documents

SEL-76-001, Proceedings From the First Summer Software Engineering Workshop, August 1976

SEL-77-001, The Software Engineering Laboratory, V. R. Basili, M. V. Zelkowitz, F. E. McGarry, et al., May 1977

SEL-77-002, Proceedings From the Second Summer Software Engineering Workshop, September 1977

SEL-77-003, Structured FORTRAN Preprocessor (SFORT), B. Chu and D. S. Wilson, September 1977

SEL-77-004, GSFC NAVPAK Design Specifications Languages Study, P. A. Scheffer and C. E. Velez, October 1977

SEL-78-001, FORTRAN Static Source Code Analyzer (SAP) Design and Module Descriptions, E. M. O'Neill, S. R. Waligora, and C. E. Goorevich, February 1978

†SEL-78-002, FORTRAN Static Source Code Analyzer (SAP) User's Guide, E. M. O'Neill, S. R. Waligora, and C. E. Goorevich, February 1978

SEL-78-102, FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 1), W. J. Decker and W. A. Taylor, September 1982

SEL-78-003, Evaluation of Draper NAVPAK Software Design, K. Tasaki and F. E. McGarry, June 1978

†This document superseded by revised document.

SEL-78-004, Structured FORTRAN Preprocessor (SFORT) PDP-11 70 User's Guide, D. S. Wilson and B. Chu, September 1978

SEL-78-005, Proceedings From the Third Summer Software Engineering Workshop, September 1978

SEL-78-006, GSFC Software Engineering Research Requirements Analysis Study, P. A. Scheffer and C. E. Velez, November 1978

SEL-78-007, Applicability of the Rayleigh Curve to the SEL Environment, T. E. Mapp, December 1978

SEL-79-001, SIMPL-D Data Base Reference Manual, M. V. Zelkowitz, July 1979

SEL-79-002, The Software Engineering Laboratory: Relationship Equations, K. Freburger and V. R. Basili, May 1979

SEL-79-003, Common Software Module Repository (CSMR) System Description and User's Guide, C. E. Goorevich, A. L. Green, and S. R. Waligora, August 1979

SEL-79-004, Evaluation of the Caine, Farber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Design Environment, C. E. Goorevich, A. L. Green, and W. J. Decker, September 1979

SEL-79-005, Proceedings From the Fourth Summer Software Engineering Workshop, November 1979

SEL-80-001, Functional Requirements/Specifications for Code 580 Configuration Analysis Tool (CAT), F. K. Banks, A. L. Green, and C. E. Goorevich, February 1980

SEL-80-002, Multi-Level Expression Design Language-Requirement Level (MEDL-R) System Evaluation, W. J. Decker and C. E. Goorevich, May 1980

SEL-80-003, Multimission Modular Spacecraft Ground Support Software System (MMS/GSSS) State-of-the-Art Computer Systems/Compatibility Study, T. Welden, M. McClellan, and P. Liebertz, May 1980

†SEL-80-004, System Description and User's Guide for Code 580 Configuration Analysis Tool (CAT), F. K. Banks, W. J. Decker, J. G. Garrahan, et al., October 1980

†This document superseded by revised document.

SEL-80-104, Configuration Analysis Tool (CAT) System Description and User's Guide (Revision 1), W. Decker and W. Taylor, December 1982

SEL-80-005, A Study of the Musa Reliability Model, A. M. Miller, November 1980

SEL-80-006, Proceedings From the Fifth Annual Software Engineering Workshop, November 1980

SEL-80-007, An Appraisal of Selected Cost/Resource Estimation Models for Software Systems, J. F. Cook and F. E. McGarry, December 1980

†SEL-81-001, Guide to Data Collection, V. E. Church, D. N. Card, F. E. McGarry, et al., September 1981

SEL-81-101, Guide to Data Collection, V. E. Church, D. N. Card, F. E. McGarry, et al., August 1982

†SEL-81-002, Software Engineering Laboratory (SEL) Data Base Organization and User's Guide, D. C. Wyckoff, G. Page, and F. E. McGarry, September 1981

SEL-81-102, Software Engineering Laboratory (SEL) Data Base Organization and User's Guide Revision 1, P. Lo and D. Wyckoff, March 1983

†SEL-81-003, Data Base Maintenance System (DBAM) User's Guide and System Description, D. N. Card, D. C. Wyckoff, and G. Page, September 1981

SEL-81-103, Software Engineering Laboratory (SEL) Data Base Maintenance System (DBAM) User's Guide and System Description, P. Lo and D. Card, April 1983

†SEL-81-004, The Software Engineering Laboratory, D. N. Card, F. E. McGarry, G. Page, et al., September 1981

SEL-81-104, The Software Engineering Laboratory, D. N. Card, F. E. McGarry, G. Page, et al., February 1982

†SEL-81-005, Standard Approach to Software Development, V. E. Church, F. E. McGarry, G. Page, et al., September 1981

SEL-81-105, Recommended Approach to Software Development, S. Eslinger, F. E. McGarry, and G. Page, May 1982

†This document superseded by revised document.

SEL-81-006, Software Engineering Laboratory (SEL) Document Library (DOCLIB) System Description and User's Guide, W. Taylor and W. J. Decker, December 1981

SEL-81-007, Software Engineering Laboratory (SEL) Compendium of Tools, W. J. Decker, E. J. Smith, A. L. Green, et al., February 1981

SEL-81-107, Software Engineering Laboratory (SEL) Compendium of Tools, W. J. Decker, W. A. Taylor, and E. J. Smith, February 1982

SEL-81-008, Cost and Reliability Estimation Models (CAREM) User's Guide, J. F. Cook and E. Edwards, February 1981

SEL-81-009, Software Engineering Laboratory Programmer Workbench Phase 1 Evaluation, W. J. Decker and F. E. McGarry, March 1981

SEL-81-010, Performance and Evaluation of an Independent Software Verification and Integration Process, G. Page and F. E. McGarry, May 1981

SEL-81-011, Evaluating Software Development by Analysis of Change Data, D. M. Weiss, November 1981

SEL-81-012, The Rayleigh Curve As a Model for Effort Distribution Over the Life of Medium Scale Software Systems, G. O. Picasso, December 1981

SEL-81-013, Proceedings From the Sixth Annual Software Engineering Workshop, December 1981

SEL-81-014, Automated Collection of Software Engineering Data in the Software Engineering Laboratory (SEL), A. L. Green, W. J. Decker, and F. E. McGarry, September 1981

SEL-82-001, Evaluation of Management Measures of Software Development, G. Page, D. N. Card, and F. E. McGarry, September 1982, vols. 1 and 2

SEL-82-002, FORTRAN Static Source Code Analyzer Program (SAP) System Description, W. A. Taylor and W. J. Decker, August 1982

SEL-82-003, Software Engineering Laboratory (SEL) Data Base Reporting Software User's Guide and System Description, P. Lo, September 1982

---

†This document superseded by revised document.

SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982

SEL-82-005, Glossary of Software Engineering Laboratory Terms, M. G. Rohleder, December 1982

SEL-82-006, Annotated Bibliography of Software Engineering Laboratory (SEL) Literature, D. N. Card, November 1982

SEL-82-007, Proceedings From the Seventh Annual Software Engineering Workshop, December 1982

SEL-82-008, Evaluating Software Development by Analysis of Changes: The Data From the Software Engineering Laboratory, V. R. Basili and D. M. Weiss, December 1982

SEL-Related Literature

††Bailey, J. W., and V. R. Basili, "A Meta-Model for Software Development Resource Expenditures," Proceedings of the Fifth International Conference on Software Engineering. New York: Computer Societies Press, 1981

Banks, F. K., "Configuration Analysis Tool (CAT) Design," Computer Sciences Corporation, Technical Memorandum, March 1980

††Basili, V. R., "Models and Metrics for Software Management and Engineering," ASME Advances in Computer Technology, January 1980, vol. 1

Basili, V. R., "SEL Relationships for Programming Measurement and Estimation," University of Maryland, Technical Memorandum, October 1979

Basili, V. R., Tutorial on Models and Metrics for Software Management and Engineering. New York: Computer Societies Press, 1980 (also designated SEL-80-008)

††Basili, V. R., and J. Beane, "Can the Parr Curve Help With Manpower Distribution and Resource Estimation Problems?", Journal of Systems and Software, February 1981, vol. 2, no. 1

---

††This article also appears in SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982.

††Basili, V. R., and K. Freburger, "Programming Measurement and Estimation in the Software Engineering Laboratory," Journal of Systems and Software, February 1981, vol. 2, no. 1

Basili, V. R., and B. T. Perricone, Software Errors and Complexity: An Empirical Investigation, University of Maryland, Technical Report TR-1195, August 1982

††Basili, V. R., and T. Phillips, "Evaluating and Comparing Software Metrics in the Software Engineering Laboratory," Proceedings of the ACM SIGMETRICS Symposium/Workshop: Quality Metrics, March 1981

Basili, V. R., R. W. Selby, and T. Phillips, Metric Analysis and Data Validation Across FORTRAN Projects, University of Maryland, Technical Report, November 1982

Basili, V. R., and R. Reiter, "Evaluating Automatable Measures for Software Development," Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity and Cost, October 1979

Basili, V.R., and D. M. Weiss, A Methodology for Collecting Valid Software Engineering Data, University of Maryland, Technical Report TR-1235, December 1982

Basili, V. R., and M. V. Zelkowitz, "Designing a Software Measurement Experiment," Proceedings of the Software Life Cycle Management Workshop, September 1977

††Basili, V. R., and M. V. Zelkowitz, "Operation of the Software Engineering Laboratory," Proceedings of the Second Software Life Cycle Management Workshop, August 1978

††Basili, V. R., and M. V. Zelkowitz, "Measuring Software Development Characteristics in the Local Environment," Computers and Structures, August 1978, vol. 10

Basili, V. R., and M. V. Zelkowitz, "Analyzing Medium Scale Software Development," Proceedings of the Third International Conference on Software Engineering. New York: Computer Societies Press, 1978

††This article also appears in SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982.

††Basili, V. R., and M. V. Zelkowitz, "The Software Engineering Laboratory: Objectives," Proceedings of the Fifteenth Annual Conference on Computer Personnel Research, August 1977

Card, D. N., "Early Estimation of Resource Expenditures and Program Size," Computer Sciences Corporation, Technical Memorandum, June 1982

Card, D. N., "Comparison of Regression Modeling Techniques for Resource Estimation," Computer Sciences Corporation, Technical Memorandum, November 1982

Card, D. N., and M. G. Rohleder, "Report of Data Expansion Efforts," Computer Sciences Corporation, Technical Memorandum, September 1982

††Chen, E., and M. V. Zelkowitz, "Use of Cluster Analysis To Evaluate Software Engineering Methodologies," Proceedings of the Fifth International Conference on Software Engineering. New York: Computer Societies Press, 1981

Freburger, K., "A Model of the Software Life Cycle" (paper prepared for the University of Maryland, December 1978)

Higher Order Software, Inc., TR-9, A Demonstration of AXES for NAVPAK, M. Hamilton and S. Zeldin, September 1977 (also designated SEL-77-005)

Hislop, G., "Some Tests of Halstead Measures" (paper prepared for the University of Maryland, December 1978)

Lange, S. F., "A Child's Garden of Complexity Measures" (paper prepared for the University of Maryland, December 1978)

Miller, A. M., "A Survey of Several Reliability Models" (paper prepared for the University of Maryland, December 1978)

National Aeronautics and Space Administration (NASA), NASA Software Research Technology Workshop (proceedings), March 1980

Page, G., "Software Engineering Course Evaluation," Computer Sciences Corporation, Technical Memorandum, December 1977

††This article also appears in SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982.

Parr, F., and D. Weiss, "Concepts Used in the Change Report Form," NASA, Goddard Space Flight Center, Technical Memorandum, May 1978

Reiter, R. W., "The Nature, Organization, Measurement, and Management of Software Complexity" (paper prepared for the University of Maryland, December 1976)

Scheffer, P. A., and C. E. Velez, "GSFC NAVPAK Design Higher Order Languages Study: Addendum," Martin Marietta Corporation, Technical Memorandum, September 1977

Turner, C., and G. Caron, A Comparison of RADC and NASA/SEL Software Development Data, Data and Analysis Center for Software, Special Publication, May 1981

Turner, C., G. Caron, and G. Brement, NASA/SEL Data Compendium, Data and Analysis Center for Software, Special Publication, April 1981

Weiss, D. M., "Error and Change Analysis," Naval Research Laboratory, Technical Memorandum, December 1977

Williamson, I. M., "Resource Model Testing and Information," Naval Research Laboratory, Technical Memorandum, July 1979

††Zelkowitz, M. V., "Resource Estimation for Medium Scale Software Projects," Proceedings of the Twelfth Conference on the Interface of Statistics and Computer Science. New York: Computer Societies Press, 1979

Zelkowitz, M. V., "Data Collection and Evaluation for Experimental Computer Science Research," Empirical Foundations for Computer and Information Science (proceedings), November 1982

Zelkowitz, M. V., and V. R. Basili, "Operational Aspects of a Software Measurement Facility," Proceedings of the Software Life Cycle Management Workshop, September 1977

††This article also appears in SEL-82-004, Collected Software Engineering Papers: Volume 1, July 1982.