ANALYZING CHANGES IN COBOL PROGRAMS DURING MAINTENANCE

by

IE-HONG LIN

B.S., NATIONAL CHANG KUNG UNIVERSITY
Tainan, Taiwan 1980

M.S., KANSAS STATE UNIVERSITY
Manhattan, Kansas 1985

A MASTER'S THESIS

submitted in partial fulfillment of the

requirement for the degree

MASTER OF SCIENCE

Department of Computing and Information Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1988

Approved by:

Major professor

TABLE OF CONTENTS

LIST OF TABLES AND FIGURES

Acknowledgments

I sincerely appreciate the knowledgeable assistance that was given to me by my major professor, Dr. David A. Gustafson. A special note of thanks is extended to my parents for their help, understanding, and guidance. Also, I am truly grateful to all my friends who express their moral support.

Chapter 1. Introduction

1-1 The Definition and Importance of Software Maintenance

Software maintenance is the final phase of the software life cycle. It is frequently viewed as a phase of lesser importance than the design and development phases. The definition of software maintenance is the performance of those activities required to keep a software system operational and responsive after it is released for use [Liu76]. The software maintenance activities modify a program to generate new output, to change the logic to incorporate a new feature, to expand functions, to add new files, etc [Liu76]. Generally, software maintenance covers not only changes to source code but also changes to specification and design notation. The reasons to perform software maintenance are to correct error and design defects, to improve the design, to convert the program to meet more advanced features, to interface the program to other programs, and to satisfy users' demands.

Maintaining an application software system tends to consume a major portion of the total life cycle costs. Statistical data shows that maintaining 2 to 10 years old software systems demands possibly as high as 40% to 60% or even 70% of the amount of the development effort for most

companies [Lie78]. Many organizations expend approximately three-fourths of their data processing budget on maintaining existing programs. And the effort is increasing as more software is produced. Many managers are dismayed by the actual expenses on maintenance.

Although it is a complex and costly phenomenon, software maintenance remains the least understood of the software processes and receives little attention. Little research and few technical approaches or "methods" have been proposed for the software maintenance. In order to differentiate the types of maintenance and bring maintenance under control, the manager needs methods to classify different types of maintenance. Proper methods of classifying types of maintenance should help in managing the maintenance effort.

A characterization of three types of maintenance activities has been proposed by Swanson[Swa76]. The three types are corrective, adaptive, and perfective maintenances. As defined, corrective maintenance is performed to correct errors that are uncovered after the software is brought to use. Adaptive maintenance is applied to properly interface with changes in the external processing environment. Perfect maintenance is applied to eliminate inefficiencies, enhance performance, or improve maintainability based on the requests from the user group [Lie78].

1-2 Objectives and Scopes

The purpose of this research focuses on classifying different types of maintenance activities based on data obtained from analyzing COBOL programs. The classification is basically a refinement of the earlier work of Swanson. Whether the previous classification methods are good enough to distinguish the maintenance activities will be discussed and compared with a proposed method from this study.

Two sets of COBOL programs, each with several versions, were used as data programs. The first set, from a Kansas company, is named as organization A programs, or program set A, throughout the study. The second set, organization B programs or program set B, however, came from a data processing environment. A shell program was developed as a tool to analyze the differences between two consecutive versions from program set A. The result lists the numbers of each statement in the first version as well as the altered, deleted, and added statements changed from the first to the second version. The rules for classifying the types of maintenance were identified from the results and then converted into a second shell program. The input for the second shell program is the output from the first shell program.

The organization B programs were later analyzed with the two shell programs to test the results and verify the rules. For convenience, the first shell program was named Maintain and the second one as Classify.

The objectives of this research are to study real-life COBOL programs to better understand what goes on in the software maintenance phase, to develop a method of classifying types of maintenance from program set A , and to check the proposed method program set B.

Chapter 2 discusses the data collection process. Explanation of the COBOL programs and a brief description of program sets A and B are given in the first section. Section 2 gives the definition of the measurements applied to calculate the data. The changes and altered, deleted, added statements are defined in Section 3. The shell programs Maintain and Classify are described in Section 4. The shell programs are the basic implementation tools employed to analyze the data programs.

Chapter 3 gives the results from running the organization A programs. Characteristics of program set A are illustrated in tables and figures. The changes between two consecutive versions are displayed. The rules of classifying the types of maintenance are listed. The reasons for iden-

tifying the maintenance are explained in detail. The rules were then written on to shell program Classify. The last section in Chapter 3 presents the results from running COBOL programs A and discusses the insights into the maintenance of the programs.

Chapter 4 involves the verification of the results in chapter 3 by classifying the program set B. All the procedures and tools employed are the same as in Chapter 3. Program data and corresponding results are represented in table or graphic forms.

Chapter 5 concludes the study and suggests recommendations for the future work.

Chapter 2. Data Collection

2-1 The COBOL Programs being Analyzed

Why choose COBOL program to analyze? COBOL is a programming language that has been designed expressly for administrative data processing. It is a high-level language and provides efficient data collection, data processing, and production of required reports. COBOL is widely used in industry and business fields.

In Chapter 1, we mentioned the program sets A and B which are the data programs in the study. The program set A, which consists of 5 COBOL programs, was analyzed in the beginning. These programs have various numbers of versions. The number of versions are 4, 5, 6, 7 and 11, respectively. The total number of versions is 33. The lines of codes also vary quite differently. The average number of the shortest program is 270; while the value of the largest is more than 4650. Table 1 displays the number of versions and average number of lines in the 5 COBOL programs.

As stated earlier, the program set B was applied to verify the results from running the program set A. The program set B, which includes 8 COBOL programs, has 20

versions on each program. All B programs have been operational for many years.

Program no.	Number of versions	Average number of lines
1	6	270
2	4	1430
3	11	4650
4	7	2070
5	5	470

Table 1. Number of Versions and Average Number of Lines of Program Set A

2-2 Measures on Types of Statements

In a COBOL program, a statement is defined as a syntatically valid combination of words and characters. Measuring the numbers of statements that have been changed between two sequential versions of a program is the basic step of collecting data for the entire research. Classified by their functions, types of statement fall into 8 categories: comment, declaration, assignment, conditional, branch, input-output, label, and other statements.

The following notations are used throughout the study. The notations were devised by Dr. David A. Gustafson and the participants in a software seminar at Kansas State University.

TYPE represents the types of statements and ALL stands for the collection of all statement types.

```
TYPE    ::=  ALL | comment | declaration |
               assignment | conditional | branch |
               input-output | label | other

comment ::=  spacing purposes | textual

assignment ::= MOVE | ADD | SUBSTRACT | COMPUTE

conditional ::= IF | ELSE | ON | AT END

branch  ::=  CALL | PERFORM | GOTO | NEXT | EXIT

input-output ::= DELETE | DISPLAY | OPEN | READ|
                   WRITE | REWRITE

other   ::=  EXAMINE | INSPECT | SEARCH | SORT |
               SET | EXEC CICIS | GOBACK
```

2-3 The Changes; Altered, Deleted, and Added Statements

Measuring the change to the code is an objective indicator of the maintenance process itself. Analyzing changes between versions is a good approach to investigate what types of maintenance are really made to the programs.

In reality, statements referring to changes have three different kinds: altered, deleted, and added statements. Altered statements can be meant to specified statements existing in two versions; however, a variable is different in its values, a statement is moved to "comment" statement because of putting asterisk in front of it by special purpose, or statements switched to another type based on programmer's need, etc. Deleted statements show on the origi-

nal version but are missing from the second version. Added statements are inserted to the original version.

For the convenience of notational representation, let

Changes ::= Altered | Deleted | Added

Changes[ALL] ::= Altered[ALL] + Deleted[ALL] + Added[ALL]

Changes[TYPE] ::= Altered[TYPE] + Deleted[TYPE] + Added[TYPE]

Altered[TYPE] : Number of Statements of specified TYPE that have been altered.

Deleted[TYPE] : Number of Statements of specified TYPE that have been deleted

Added[TYPE] : Number of Statements of specified TYPE that have been added.

2-4 Tools for Analyzing Data Programs

The shell program Maintain which invokes several UNIX utilities such as diff and grep, was written to analyze the COBOL programs. Six modules are included in the program; they are checking, preprocessing, distinguishing, difference, calculation, and report modules. Each module has its special function. Figure 1 gives the flow chart of the shell program Maintain. The inputs are two versions of a COBOL programs. The input sequence has to be the same order for the comparison purposes. The executing command "Maintain

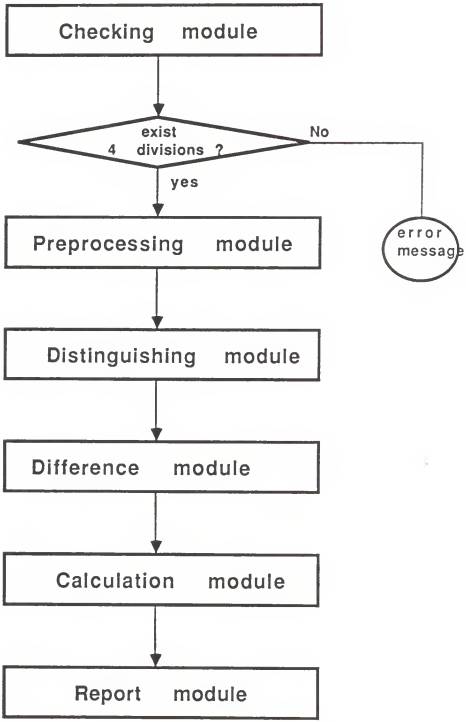

Figure 1 Flow chart of the Shell program Maintain

version.1 verion.2" reports the output which lists the results changed from version.1 to version.2. In reverse, the command "Maintain version.2 version.1" generates the report which contains the data modified from version.2 to version.1. The output of the program Maintain, assuming the command "Maintain version.1 version.2", includes the numbers of statements in version.1 and the overall numbers of statements in altered, deleted, and added statements from version.1 to version.2. Appendix A lists the program Maintain. Two example COBOL programs Version.1 Version.2 are given in Appendix B and C. Appendix D shows the result from running the Maintain program on the two versions. The function of each module in the program Maintain is described as below.

a. Checking module

Every COBOL program consists of four divisions in the following order: identification, environment, procedure and data divisions. The checking module checks to verify the existence of all four divisions in the two input data programs. Error messages referring to the absence of divisions are printed out in case of the missing of any division. Program terminates if errors are detected.

b. Preprocessing module

The preprocessing module removes all superfluous blank

spaces, tabs, and blank lines. If the difference of a statement in version.1 and the version.2 is only the addition or deletion of a blank space, it should not be marked as altered. The same situation can be extended to the insertion or removal of blank lines simply for spacing purposes. This module also removes numbers from tail end of lines. The numbers attached at the end of lines have no meaning. COBOL programs use it as marked symbol to easily identify a sequence of codes.

c. Distinguishing module

This module assigns unique characters to every statement in order to identify types of each particular statement. The function ensures that each statement can be properly identified after changes are made. For example, all the statements in the identification division are prefixed with "Comment". In the environment division, the statements are marked with "Env". The FILE SECTION and WORKING-STORAGE statement in procedure division are attached with "DeSetn". The rest of statements in procedure division are added with "Dlrtn".

d. Difference module

The difference module utilizes the "diff" function to find the differences between two versions. It compares two versions of program and notes altered, deleted, and added statements. Three temporary files are created once the

altered, deleted, and added statements are in existence. The files which store the deleted and added statements copy the statements from the analyzed versions. The file having the altered statements contains the old statements and the new statements. If there are several places that statements are altered, we name each place as a block of altered statements.

e. Calculation module

The calculation module computes the numbers of respective type of statements. The module generates overall numbers of types of statements for version.1. Three temporary files, if they exist, are also analyzed by this module to produce output.

f. Report module

The report module produces output for the shell program Maintain. The output includes the overall analysis of the statements in version.1. It displays the number of statements in version.1. The result also lists the numbers of statements in altered, deleted, and added statement, if they exist. The actual altered, deleted and added statements are displayed at the end of the output. It is easy to identify the statements by the use of the special characters which were added in the distinguishing module.

The algorithms for developing the shell program Classi-

fy are based on the rules for classifying types of maintenance. The rules and corresponding algorithms will be described in chapter 3.

Chapter 3. Result of Analysis

3-1 Program Characteristics

Program set A, consisting of 5 different programs with 33 versions, was initially introduced to be analyzed by the tools, Maintain and Classify shell programs. Table 2 illustrates the characteristics of program set A. The minimum and maximum numbers of statements are given to represent the structure of statements in each program. Figure 2 gives the programs' characteristics by means of graphic form. Of the 8 types of statements, numbers of input-output, label, and other statements are ignored because of their relatively small number compared to the rest of the five types of statements. Investigating the graph, it is easy to realize that assignment statements play an important role in program set A. However, comment statements also are significant due to their frequent occurrence.

3-2 The Changes

The shell program Maintain runs two sequential versions of a program. The results of analysis on program set A

are 28 deltas. The contents of a delta includes Number[TYPE] of each statement and Altered[TYPE], Deleted[TYPE], and Added[TYPE] between the original and new versions. In addition to these numbers, the statements being changed are also listed as part of the content of a delta. The raw data of Altered[TYPE], Deleted[TYPE], and Added[TYPE] in the 28 delta are shown on Table 3, 4, and 5, respectively.

3-3 Types of Maintenance

From the 28 deltas with attached listings of changed statements, six types of maintenance were identified; they were corrective, adaptive, retrenchment, retrieving, pretty printing, and documentation maintenance. Compared with the three classical types of maintenance proposed by Swanson, it is clear that perfective maintenance was excluded from the classification and replaced with retrenchment, retrieving, pretty printing, and documentation maintenances. The reason of excluding perfective maintenance from classification is due to the difficulty of predicting the intention of the programmer doing the enhancement. The reason for the programmer to update sources code is too complicated to trace simply from investigating changed statements. The changes on the rest of statements may be a side effect of

Program No.	1		2		3		4		5	
Number of Version	6		4		1 1		7		5	
Max./Min.	Min	Max.	Min.	Max.	Min	Max.	Min.	Max.	Min.	Max.
ALL	270	278	1428	1438	4641	4684	2057	2102	426	513
comment	58	69	195	248	782	870	390	399	79	108
declaration	80	80	607	607	1171	1178	533	539	228	233
assignment	51	51	378	406	1436	1483	653	660	71	104
MOVE	41	41	313	340	1302	1347	568	574	30	58
conditional	30	31	78	94	606	627	218	229	16	31
IF	22	23	61	71	433	446	166	173	9	19
ELSE	3	3	14	18	166	171	48	52	4	9
ON	5	5	3	5	6	9	3	3	3	3
branch	21	21	63	66	390	394	166	170	14	32
CALL	0	0	6	6	25	25	13	13	1	1
PERFORM	4	4	10	10	165	167	76	78	3	6
GOTO	17	17	45	48	190	190	67	69	10	21
EXIT	0	0	2	2	9	12	9	9	0	0
STOP	0	0	0	0	0	0	0	0	0	0
input-output	0	0	0	0	2	2	0	0	1	1
label	16	17	45	46	162	164	78	80	7	9

Table 2. Numbers of Statements of Program Set A

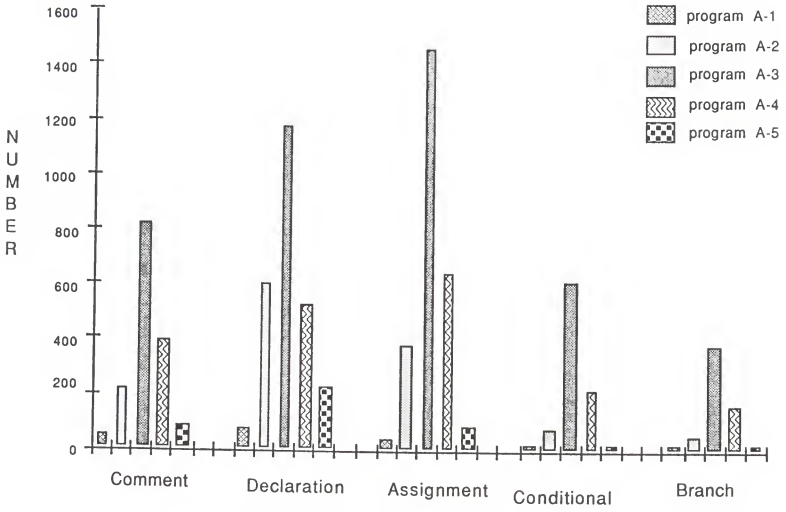

Figure 2. Numbers[TYPE] of Program set A

delta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
ALL	1	3	11	1	0	28	17	4	3	3	73	1	2	3	0	0	2	21	1	3	0	0	3	6	3	1	8	17
comment	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	15
declaration	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
assignment	0	0	0	0	0	19	9	0	2	3	47	0	0	3	0	0	2	13	1	3	0	0	2	0	1	0	2	1
MOVE	0	0	0	0	0	18	9	0	2	3	45	0	0	0	0	0	2	13	1	3	0	0	2	0	1	0	2	1
conditional	1	1	0	1	0	7	7	2	1	0	21	0	1	0	0	0	0	7	0	0	0	0	1	0	0	1	1	1
IF	1	1	0	1	0	5	3	2	0	0	13	0	1	0	0	0	0	4	0	0	0	0	1	0	0	1	1	0
ELSE	0	0	0	0	0	1	3	0	1	0	5	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0
ON	0	0	0	0	0	1	1	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
branch	0	0	0	0	0	2	0	2	0	0	4	0	1	0	0	0	0	1	0	0	0	0	0	5	0	0	2	0
CALL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PERFORM	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
GOTO	0	0	0	0	0	2	0	2	0	0	4	0	0	0	0	0	0	1	0	0	0	0	0	4	0	0	2	1
EXIT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STOP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
input-output	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
label	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
other	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0

Table 3. Numbers of Altered Statements in 28 Deltas

delta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
ALL	0	0	0	0	3	0	0	0	0	0	0	0	0	0	5	6	0	0	0	0	0	0	0	2	15	0	4	0
comment	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	1	0
declaration	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	6	0	3	0
assignment	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	2	9	0	0	0
MOVE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	2	0	0	0	0	0	0	0	2	6	0	0	0
conditional	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
IF	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0
ELSE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
ON	0	0	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
branch	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
CALL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PERFORM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
GOTO	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0
EXIT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STOP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
input-output	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
label	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
other	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 4. Numbers of Deleted Statements in 28 Deltas

delta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28
ALL	0	0	0	0	10	0	0	9	0	0	0	3	9	1	7	9	0	0	0	0	4	8	0	29	26	14	51	6
comment	0	0	0	0	5	0	0	4	0	0	0	0	3	0	0	4	0	0	0	0	1	3	0	5	0	11	19	0
declaration	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	10	1	0	0
assignment	0	0	0	0	0	0	0	0	0	0	0	2	0	1	3	0	0	0	0	0	1	0	0	9	13	0	13	2
MOVE	0	0	0	0	0	0	0	0	0	0	0	2	0	1	3	0	0	0	0	0	1	0	0	8	13	0	13	2
conditional	0	0	0	0	0	0	0	0	0	0	0	1	3	0	2	0	0	0	0	0	1	0	0	4	2	0	6	2
IF	0	0	0	0	0	0	0	0	0	0	0	1	3	0	2	0	0	0	0	0	1	0	0	4	1	0	5	1
ELSE	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
ON	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
branch	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0	0	0	0	0	1	0	0	3	1	2	8	2
CALL	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
PERFORM	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	2	1	0
GOTO	0	0	0	0	0	0	0	0	0	0	0	0	2	0	2	0	0	0	0	0	0	0	0	2	1	0	5	2
EXIT	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
STOP	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
input-output	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
label	0	0	0	0	0	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	0	1	0	1	0	0	2	0
other	0	0	0	0	4	0	0	4	0	0	0	0	0	0	0	4	0	0	0	0	0	4	0	2	0	0	3	0

Table 5. Numbers of Added Statements in 28 Deltas

changes to comment statements.

The definitions of six types of maintenance are given as below:

a. Corrective maintenance:

The corrective maintenance is the maintenance attributed to minor revision on the original version. Correcting errors or failures from source codes is also identified as corrective maintenance. The corrective maintenance results in statements being altered as well as a few statements being added and/or deleted.

b. Adaptive maintenance:

The adaptive maintenance concerns the addition of new functions or the deletion of old functions from the original codes to meet external environment requirement. In this type of maintenance, there are a lot of statements added or deleted in addition to a few statements changed.

c. Retrenchment maintenance:

The retrenchment maintenance temporarily removes a function from executable code by adding asterisks in front of the statements. The original code is converted to document or comment statements. An example of retrenchment maintenance is as follow:

Statements in original version:

```
IF MAORD = 'A'
```

```
   GO TO 0530-READ-MATLDESC-PURCHSPECS
IF MAORD = 'N'
   GO TO 0530-READ-MATLDESC-PURCHSPECS
```

Statements in new version:

```
*  IF MAORD = 'A'  GO TO 0530-READ-MATLDESC-PURCHSPECS
*  IF MAORD = 'N'  GO TO 0530-READ-MATLDESC-PURCHSPECS
```

Note that they are not only adding comments but also combining four lines into two lines.

The result of retrenchment maintenance increases the Number[comment] from the original version to the new version. It may also decrease the numbers of statements between two versions.

d. Retrieving maintenance:

The retrieving maintenance removes asterisks from comment or documentation statements and brings the statements back into service. The changed statements become part of the executable code. The retrieving maintenance is the reverse of retrenchment maintenance. An example of retrieving maintenance is listed as below:

Statements in original version:

```
*
*  IF EIBTRMID NOT = '302M' OR EIBTRMID NOT = '700Q'
*     EXEC CICS
*     SEND TEXT FROM(NICE-TRY-MESSAGE_
```

```
*              LENGTH
*              ERASE
*              FREEKB
*    END-EXEC
*    EXEC CICS
*    RETRUN
*    END-EXEC
```

Statements in new version:

```
IF EIBTRMID NOT = '302M' OR EIBTRMID NOT = '700Q'
   EXEC CICS SEND TEXT FROM(NICE-TRY-MESSAGE LENGTH ERASE FREEKB
   END-EXEC EXEC CICS RETRUN  ND-EXEC
```

e. Pretty printing maintenance:

The pretty printing maintenance simply add asterisks for spacing purposes. There is no other function added or deleted. The objective of pretty printing is to allow program to be easy to read. The pretty printing maintenance increases Number[comment] in the new version and does not change other statements.

f. Documentation maintenance:

The documentation maintenance is the addition of comment statement to a program. This is different from pretty printing maintenance. The documentation maintenance puts descriptions or explanations just before a block of source codes. Well-documented program can reduce the effort of the

reader to understand the program. The number of statements changed in documentation maintenance is similar to that of pretty printing maintenance.

3-4 Precise Rules

The precise rules for classifying types of maintenance are analyzed from the empirical data received from executing the shell program Maintain on program set A. The rules were converted into the shell program Classify (see algorithm in Appendix E). The input for the Classify program is the output from the Maintain program. Appendix F gives an example result from running the program Classify. The discussion of the rules on correction and adaptive maintenances is grouped together because of their similar situation. The same condition can be applied to retrenchment and retrieving, as well as pretty printing and documentation.

a. Corrective and Adaptive maintenances:

Here a block of statements altered is defined as a series of statements altered. If there are more than three blocks of statements in which the lines of codes are modified, or the addition and deletion of statements other than comment statements is greater than 10, the delta is classified as adaptive maintenance. Otherwise, it is said to be corrective maintenance. The term "modified block" is de-

fined as a block where the ratio of Number[TYPE] in two versions is greater than 2 if Number[TYPE] in two versions are both more than 10, or the ratio is greater than 5 if one of Number[TYPE] is less than 10. Detailed algorithms can be found in the shell program Classify in Appendix B.

b. Retrenchment and Retrieving maintenances:

In both types of maintenance, there are some altered and no deleted or added statements. The altered statements cause the changes of comment statements. Number[comment] is decreased from original version to new version in retrenchment maintenance. The value, however, is increased in retrieving maintenance.

c. Pretty printing and Documentation maintenances:

For both types of maintenance, the increase or decrease of Number[comment] is due to the addition or deletion of comment statements. If the goal of added or deleted comment statements is for spacing purpose only, the type of maintenance is classified as pretty printing maintenance. Otherwise, it is documentation maintenance. The shell program Classify can distinguish between these two types of maintenance.

3-5 Result and Discussion

The result of types of maintenance in 28 deltas from program set A is illustrated on Table 6. The empirical data are collected in Table 3, 4 and 5. Of the 28 deltas, there exists single types and combination of two or three types of maintenance. For simplicity, the combination of corrective and documentation maintenance is expressed as corrective & documentation maintenances. This example extends to any combination.

Program A-1 consists of 6 versions and 5 deltas. As there is only 1 statement altered, delta 1 is classified as corrective maintenance. In delta 2, original Number[comment] and new Number[comment] are equal to 59 and 69, respectively. The increment of 10 comment statements is due to 3 statements in original version converting to 10 comment statements in new version. As result, delta 2 is classified as retrenchment maintenance. The decrement of 11 comment statements in delta 3 is from 11 original statements modifying to 3 new statements. From the classification rules, it is clear that delta 3 is retrieving maintenance. The modification in delta 4 is similar to that in delta 1. Delta 5 has two types of maintenance, corrective & pretty printing. The pretty printing maintenance can be identified from the value of Added[comment] which is equal to the

increment of Number[comment] between two versions. Corrective maintenance is determined by the values of Deleted[TYPE] and Added[TYPE].

Program A-2 includes 4 versions. Delta 6 and 7 are retrenchment and the reason of classification is the same as from delta 2. Delta 8 has three types of maintenance, corrective & retrenchment & pretty printing. The 4 added comment statements result in the pretty printing maintenance. The retrenchment maintenance is attributed to altered statements from 4 to 2 statements. The 9 added statements, however, are classified as corrective maintenance.

Deltas 9 to 18 belong to program A-3. Of the ten deltas, five deltas are classified as corrective maintenance. Delta 11 has corrective & retrenchment types. Delta 13 has three types of maintenance. The 3 added comment statements belong to pretty printing maintenance. One block of altered statements belongs to retrenchment maintenance. The other block and the added statements, however, are classified as corrective maintenance. Delta 15 and 16 have corrective & pretty printing maintenances. In delta 18, the increment of comment statement is identified as retrenchment maintenance which came from altered statements. Of the altered statements, some contributed to corrective maintenance.

Of six deltas in program A-4, there are 3 deltas classified as corrective maintenance. Deltas 21 and 22 have the similar modification except that delta 21 is documentation and delta 22 is pretty printing maintenance. The shell program Classify can make the distinction. In delta 24, 5 added comment statements have two types, pretty printing & documentation. The other changed statements contribute to adaptive maintenance.

In program A-5, delta 25 is adaptive type because lots of statements added or deleted. Deltas 26 and 27 are classified as adaptive & documentation maintenance. The altered 15 comment statements cause the decrease of comment statements in delta 28, therefore, the delta is said to be retrieving maintenance in addition to corrective maintenance.

Figures 3 and 4 illustrate the overall analysis on program set A. Figure 3 displays the numbers of deltas on types of maintenance. The number of deltas which only has corrective type is 10. None of the deltas which have documentation and pretty printing maintenance. There exists only 3 deltas owning three types of maintenance. It is verified that program maintainer did not change many things on any version of program set A. Figure 4 lists the percentage of occurrence on each type of maintenance.

The rules described in previous section were used to classify types of maintenance in program set A. In next chapter, the rules are verified with program set B, which are received from different environment.

delta	1	2	3	4	5	6	7	8	9	10	11	12	13	14
Altered[ALL]														
from	1	3	11	1	0	28	17	4	3	3	73	1	2	3
to	1	10	3	1	0	28	20	2	3	9	76	2	2	3
block	1	1	1	1	0	2	2	1	3	3	6	1	2	3
Deleted[ALL]	0	0	0	0	3	0	0	0	0	0	0	0	0	0
Added[ALL]	0	0	0	0	10	0	0	9	0	0	0	3	9	1
Number[comment]														
original	59	59	69	58	58	195	222	242	782	782	782	856	856	860
new	59	69	58	58	63	223	242	248	782	782	856	856	860	860
Altered[comment]	0	0	11	0	0	0	0	0	0	0	0	0	0	0
Deleted[comment]	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Added[comment]	0	0	0	0	5	0	0	4	0	0	0	0	3	0

Classification														
Corrective	*			*	*			*	*	*	*	*	*	*
Adaptive														
Retrenchment		*				*	*	*			*		*	
Retrieving			*											
Pretty printing					*			*					*	
Documentation														

Table 6a. Classification of Program Set A

delta	15	16	17	18	19	20	21	22	23	24	25	26	27	28
Altered[ALL]														
original	0	0	2	21	1	3	0	0	3	6	3	1	8	17
new	0	0	8	30	1	9	0	0	2	7	3	11	12	10
block	0	0	1	5	1	3	0	0	1	6	3	1	4	2
Deleted[ALL]	5	6	0	0	0	0	0	0	0	2	15	0	4	0
Added[ALL]	7	9	0	0	0	0	4	8	0	29	26	14	51	6
Number[comment]														
original	860	856	860	860	390	390	390	391	394	394	79	79	90	108
new	856	860	860	870	390	390	391	394	394	399	79	90	108	93
Altered[comment]	0	0	0	0	0	0	0	0	0	0	0	0	2	15
Deleted[comment]	4	0	0	0	0	0	0	0	0	0	0	0	1	0
Added[comment]	0	4	0	0	0	0	1	3	0	5	0	11	19	0

Classification														
Corrective	*	*	*	*	*	*	*	*	*					*
Adaptive										*	*	*	*	
Retrenchment				*										
Retrieving														*
Pretty printing	*	*						*		*				
Documentation							*			*		*	*	

Table 6b. Classification of Program Set A

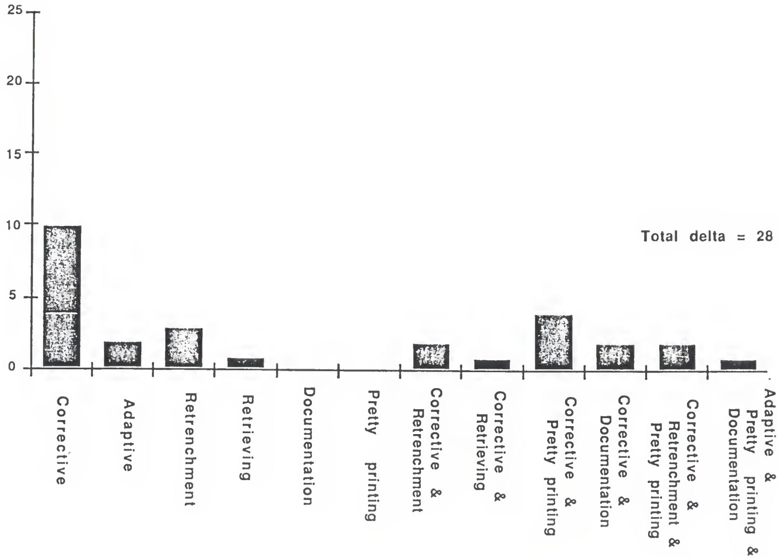

Figure 3. Numbers of Delta on Types Maintenance of Program Set A

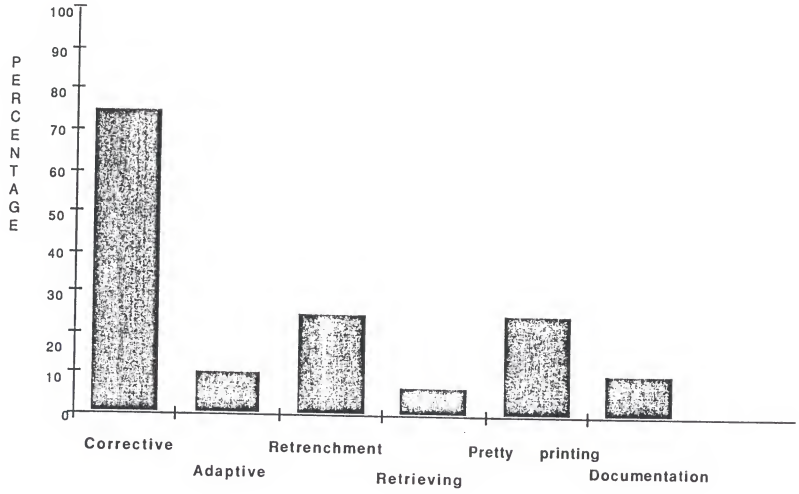

Figure 4. Percentages of Ocurrence on Each type of Maintenance of Program Set A

Chapter 4. Classification

4-1 Data Programs

Program set B, which includes 8 programs, was obtained from a data processing environment. These programs have been in use for more than 10 years. For each program, 20 versions were received and analyzed with the shell programs Maintain and Classify to verify the classification rules proposed in chapter 3. The result generates 19 deltas on each program and a total number of 152 on program set B. From the sequence of maintenance on a specific program, it indicates the types of maintenance and identifies what is really made to the modification on the source codes during the maintenance period. The classifications were then compared to the maintainers' comments.

4-2 Result

In the 19 deltas from program B-1, there are 3 corrective, 13 corrective & documentation , 2 corrective & retrenchment & documentation maintenance, and 1 adaptive & retrenchment & documentation maintenance (see Table 7). The 13

corrective & documentation maintenance delta have similar changed value set. It is easy to make distinctions on these types of maintenance. All the increment of Number[comment] results from the number of Added[comment]. Additional changed statements, however, contribute to corrective maintenance.

On program B-2, 10 corrective and 7 corrective & document maintenance were found during the analysis. Delta 11 is classified as corrective & retrenchment maintenance. Delta 1 does not have any modification at all. The result of program B-2 is represented on Table 8.

From delta 1 through 11, the modifications on program B-3 were steady except for delta 4 being classified as adaptive & pretty printing maintenance. The changes in deltas 12, 13 , and 14 were large comparing to the rest of deltas. In these phases, a combination of 4 types of maintenance were identified. In addition to the high occurrence of maintenance types, the altered numbers were also very high. Delta 15 was back to general modification. Delta 16 was adaptive & documentation maintenance. Delta 17 includes retrenchment and retrieving maintenance together, which is seldom found in the classification of program set B. Delta 18, like delta 1 in program B-2, has not any change. The last delta is only corrective maintenance.

During the changes on program B-4, the modifications were steady from the result shown on Table 10. Only types of corrective and corrective & documentation maintenance were identified. Of the 19 deltas, 16 delta were corrective & documentation and 3 were corrective maintenance.

There is only corrective & documentation maintenance in changes after delta 9 on program B-5. During the early changes, 4 of 8 deltas were also classified as corrective & documentation maintenance. Two deltas were corrective maintenance only. Delta 3 is classified as corrective & retrenchment & documentation maintenances and delta 5 is adaptive & documentation maintenance (see Table 11).

Program B-6 includes 7 corrective and corrective & documentation maintenances, respectively. The rest of the deltas contains combination of types of maintenance. Delta 16, 17, and 18 all include adaptive maintenance. Delta 9 is the only one without corrective maintenance. It is classified as retrenchment & documentation maintenances. Detailed results are shown in Table 12.

Corrective and documentation maintenances are two types which exist in the changes on program B-7 (see Table 13). Nine corrective and 9 corrective & documentation maintenance types are classified from the program Classify.

The results on program B-8 is similar to that on program B-7. Fourteen deltas with corrective & documentation maintenance were classified. Delta 16 is the only one classified as adaptive maintenance.

From the results collected from 152 deltas on program set B, the combination of corrective and documentation types were the most frequently occurring maintenance; 81 deltas belong to this combination types of maintenance. Added comment statements in the identification division contribute to documentation maintenance and the changed statements resulting in corrective maintenance. It is concluded that most maintainers explained what they modified in the identification division and gave actual changes in the procedure division. The percentages of occurrence on each type of maintenance for program set B are represented on Table 5. In contrast to the same representation from program set A, pretty printing maintenance happened at lower percentage.

4-3 Validation

The classifications were checked against the explanations for the changes given by the maintainers as comments in the environment section. None of the explanations contradicted the classifications made by the classify program. However,

the classifications gave a better indication than the comments about what types of activity had occurred. Thus, we feel that rules were successful in classifying the maintainence activities.

delta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Altered[ALL]																			
from	6	15	9	52	41	412	45	15	19	42	26	45	19	14	6	2	6	10	21
to	12	18	9	99	46	430	87	28	33	13	36	50	25	30	6	2	7	14	34
block	5	4	5	26	22	132	13	10	4	26	9	8	6	6	4	2	4	7	9
Deleted[ALL]	0	0	0	5	1	0	0	3	0	1	0	0	0	0	0	0	0	0	0
Added[ALL]	7	9	4	44	18	8	0	6	1	3	3	5	8	3	6	7	0	6	10
Number[comment]																			
original	213	216	219	219	228	235	238	248	251	257	258	262	267	271	277	279	285	286	291
new	216	219	219	228	235	238	248	251	257	258	262	267	271	277	279	285	286	291	293
Altered[comment]	0	0	2	0	0	5	0	0	0	1	2	0	1	1	6	0	0	3	0
Deleted[comment]	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
Added[comment]	3	3	0	3	7	3	9	3	5	3	3	5	4	0	2	7	0	5	2
Classification																			
Corrective	*	*	*		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Adaptive				*															
Retrenchment				*			*		*										
Retrieving																			
Pretty printing																			
Documentation	*	*		*	*	*	*	*	*	*	*	*	*		*	*		*	*

Table 7. Classification of Program B-1

delta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Altered[ALL]																			
from	0	77	30	1	2	146	1	35	1	7	27	4	0	1	46	39	109	2	1
to	0	84	53	1	2	163	1	59	1	7	35	2	0	1	41	95	101	2	1
block	0	42	14	1	2	10	1	7	1	3	6	1	0	1	5	5	24	2	1
Deleted[ALL]	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
Added[ALL]	0	0	0	0	0	7	0	0	0	0	4	0	0	5	3	33	22	5	0
Number[comment]																			
original	157	157	160	162	162	162	169	169	180	180	180	181	181	181	183	186	189	193	198
new	157	160	162	162	162	169	169	180	180	180	181	181	181	183	186	189	193	198	198
Altered[comment]	0	56	0	0	0	0	0	1	0	1	1	0	0	0	0	0	0	0	0
Deleted[comment]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Added[comment]	0	0	2	0	0	7	0	0	0	0	0	0	0	2	3	3	4	5	0

Classification																			
Corrective		*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Adaptive																			
Retrenchment											*								
Retrieving																			
Pretty printing																			
Documentation			*			*								*	*	*	*	*	

Table 8. Classification of Program B-2

delta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Altered[ALL]																			
from	16	14	2	79	7	5	2	49	186	9	4	2782	1096	1091	4	1420	913	0	3
to	22	25	1	69	16	11	2	62	184	3	2	2633	1090	1087	3	1460	904	0	3
block	5	6	1	12	2	5	2	3	18	2	2	189	32	32	3	5	11	0	3
Deleted[ALL]	0	0	3	101	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Added[ALL]	3	2	0	12	4	6	3	13	5	4	5	0	10	9	2	137	3	0	0
Number[comment]																			
original	388	391	393	393	397	399	404	407	413	417	421	424	436	439	441	443	467	470	470
new	391	393	393	397	399	404	407	413	417	421	424	436	439	441	443	467	470	470	470
Altered[comment]	3	0	0	0	0	0	0	0	0	0	0	424	125	125	0	165	134	0	0
Deleted[comment]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Added[comment]	3	2	0	4	2	5	3	6	4	4	3	0	11	10	2	24	3	0	0
Classification																			
Corrective	*	*	*		*	*	*	*	*	*	*	*	*	*	*		*		*
Adaptive				*												*			
Retrenchment													*	*			*		
Retrieving												*	*	*			*		
Pretty printing				*								*							
Documentation	*	*			*	*	*	*	*	*	*	*	*	*	*	*			

Table 9. Classification of Program B-3

delta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Altered[ALL]																			
from	4	10	1	10	4	5	16	3	19	5	2	46	3	1	17	4	3	38	3
to	1	10	1	16	4	5	18	3	21	6	2	54	8	1	31	4	3	38	5
block	1	4	1	2	2	2	2	3	2	3	1	2	2	1	5	3	2	5	3
Deleted[ALL]	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	2	0
Added[ALL]	0	3	6	3	2	2	8	2	3	3	0	5	6	7	6	0	2	3	4
Number[comment]																			
original	95	95	98	101	104	106	108	112	114	117	120	120	125	128	130	136	136	138	141
new	95	98	101	104	106	108	112	114	117	120	120	125	128	130	136	136	138	141	143
Altered[comment]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Deleted[comment]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Added[comment]	0	3	3	3	2	2	4	2	3	3	0	5	3	2	6	0	2	3	2
Classification																			
Corrective	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*
Adaptive																			
Retrenchment																			
Retrieving																			
Pretty printing																			
Documentation		*	*	*	*	*	*	*	*	*		*	*	*	*		*	*	*

Table 10. Classification of Program B-4

delta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Altered[ALL]																			
from	75	2	60	2	516	2	3	71	42	70	4	274	3	1	5	1	20	69	284
to	114	2	59	2	492	2	3	158	86	76	4	378	9	1	5	1	24	71	319
block	37	2	17	2	86	2	3	16	2	11	4	38	3	1	1	1	2	6	42
Deleted[ALL]	0	0	8	0	9	0	0	0	0	0	0	1	0	0	0	0	0	0	0
Added[ALL]	6	2	3	2	12	2	2	4	12	3	2	76	5	3	4	2	6	2	5
Number[comment]																			
original	289	289	291	329	331	338	340	342	353	361	364	366	375	379	382	386	388	391	392
new	289	291	329	331	338	340	342	353	361	364	366	375	379	382	386	388	391	392	395
Altered[comment]	2	0	0	0	3	0	0	2	0	0	0	4	0	0	0	0	1	1	0
Deleted[comment]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Added[comment]	0	2	3	2	7	2	2	0	8	3	2	9	4	3	4	2	3	2	3

Classification

Corrective	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•	•	•	•
Adaptive					•														
Retrenchment			•																
Retrieving																			
Pretty printing																			
Documentation		•	•	•	•	•	•		•	•	•	•	•	•	•	•	•	•	•

Table 11. Classification of Program B-5

delta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Altered[ALL]																			
from	9	36	13	4	135	3	11	7	6	36	32	16	2	11	11	1	34	106	14
to	9	125	32	4	110	5	5	15	11	41	31	20	2	11	14	1	25	136	27
block	6	11	4	4	61	2	1	3	3	5	6	3	2	7	2	1	14	28	5
Deleted[ALL]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	1
Added[ALL]	5	7	0	7	3	0	0	5	6	0	0	0	2	5	4	17	0	24	2
Number[comment]																			
original	440	445	493	493	500	502	502	502	505	510	510	510	510	512	517	521	525	519	530
new	445	493	493	500	502	502	502	505	510	510	510	510	512	517	521	525	519	530	532
Altered[comment]	0	0	0	0	6	0	3	3	0	3	3	3	0	0	0	0	0	5	0
Deleted[comment]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Added[comment]	5	3	0	7	0	0	0	3	4	0	0	0	2	5	4	4	0	8	2

Classification																			
Corrective	*	*	*	*	*	*	*	*		*	*	*	*	*	*				*
Adaptive																*	*	*	
Retrenchment		*	*						*										
Retrieving																		*	
Pretty printing																			
Documentation	*	*		*				*	*				*	*	*	*		*	*

Table 12. Classification of Program B-6

delta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Altered[ALL]																			
from	1	2	0	2	1240	1380	1	2	1	1	1273	9	1	1	399	902	105	0	52
to	1	4	0	2	1239	1383	1	2	1	1	1285	7	4	1	411	909	100	0	27
block	1	2	0	2	50	9	1	2	1	1	46	1	1	1	8	30	7	0	8
Deleted[ALL]	0	0	4	0	0	0	0	0	4	0	2	0	0	0	0	1	0	0	0
Added[ALL]	0	3	0	2	2	1	1	2	0	0	2	0	2	8	7	6	2	3	5
Number[comment]																			
original	96	96	96	96	98	98	102	102	104	104	104	107	107	107	109	116	126	129	132
new	96	96	96	98	98	102	102	104	104	104	107	107	107	109	116	126	129	132	136
Altered[comment]	0	0	0	0	8	8	0	0	0	0	7	0	0	0	0	3	0	0	0
Deleted[comment]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Added[comment]	0	0	0	0	0	0	0	2	0	0	3	0	0	2	7	5	3	0	4
Classification																			
Corrective	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*		*
Adaptive																			
Retrenchment																			
Retrieving																			
Pretty printing																			
Documentation				*		*		*			*			*	*	*	*	*	*

Table 13. Classification of Program B-7

delta	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19
Altered[ALL]																			
from	3	3	1	2	32	11	5	2	20	41	7	1	1	14	13	642	132	8	10
to	6	7	1	2	14	17	6	2	35	58	3	2	1	17	55	654	113	12	15
block	3	3	1	2	7	7	1	2	7	7	3	1	1	5	7	129	14	4	4
Deleted[ALL]	1	0	0	0	0	0	0	0	0	0	1	0	0	0	0	5	0	0	0
Added[ALL]	5	3	3	2	16	5	0	2	6	15	4	0	0	11	7	16	0	7	4
Number[comment]																			
original	108	113	116	118	120	128	133	133	135	136	138	141	141	141	141	145	151	154	157
new	113	116	118	120	128	133	133	135	136	138	141	141	141	141	145	151	154	157	161
Altered[comment]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	94	6	0	0
Deleted[comment]	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Added[comment]	5	3	2	2	8	5	0	2	1	2	3	0	0	0	4	0	3	3	4
Classification																			
Corrective	*	*	*	*	*	*	*	*	*	*	*	*	*	*	*		*	*	*
Adaptive																*			
Retrenchment																			
Retrieving																			
Pretty printing																			
Documentation	*	*	*	*	*	*		*	*	*	*				*		*	*	*

Table 14. Classification of Program B-8

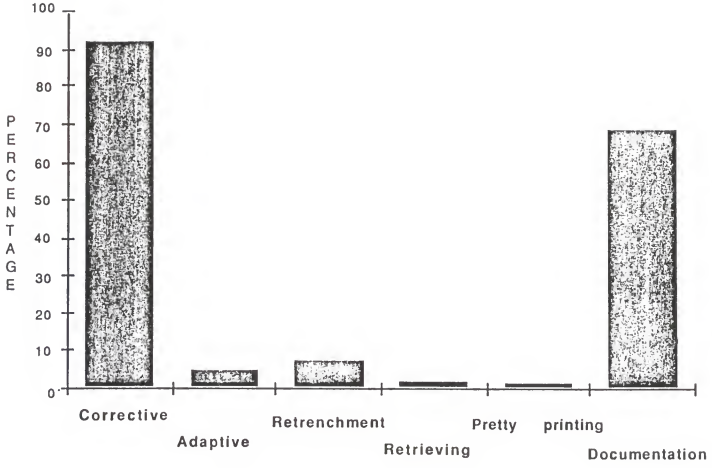

Figure 5. Percentages of Ocurrence on Each type of Maintenance of Program Set B

Chapter 5. Conclusion and Future Work

Chapter 3 gave the results of this study on the COBOL program set A. The classification rules were presented to distinguish various types of maintenance. Six types of maintenance were identified from the changes on sequential versions of the programs. The rules are in the effort to classify types of maintenance on program set A. Then the classification rules were used to verify the results from the analysis on the program set B, which came from another environment. The result of classification are also found to be satisfactory by comparing them with the maintainers' comments.

The study presents a method of classifying types of maintenance from empirical data of changes between two versions during maintenance. By means of the tools, the shell programs Maintain and Classify, managers can easily identify the types of maintenance existing in the sequential modification to a specific program. In an effort to reduce the cost of maintenance, managers can take the method as a reference with which to evaluate their maintenance effort; this method will give them an objective classification of their effort.

Recommendations for the future work for this study are proposed as following:

1) Validate the presented method with more COBOL programs from different sources to check the effectiveness of classifying types of maintenance from the tools.

2) Modify and extend the tools to test the results of programs written by other languages. Are the method and rules good enough to classify types of maintenance in other languages?

3) Six types of maintenance were classified in this study from the analysis on program set A. More classification types could be embedded in COBOL programs.

4) The presented rules were satisfactory in classifying maintenance types. Refinement of the rules is suggested to distinguish more maintenance types.

References

[Ben87] Bendifallah, S. and Scacchi, W. "Understanding Software Maintenance Work", IEEE Trans on Software Engineering, SE-3(3):311-323, Mar. 1987

[Ber84] Berns, G. M., "Assessing Software Maintainability", Communication of the ACM, 27(1):14-23, June 1984

[Boe76] Boehm, E. W. and Brown, J. R. and Lipow, M., "Quantitative Evaluation of Software Quality", Proc. 2nd International Conference on Software Engr., San Francisco, Oct. 1976

[Bra85] Branch, M. A. and Jackson, M. C. and Laviolette, M. C., "Software Maintenance Management", pp.62-68, 1985 Software Maintenance

[Che85] Chen, V. Y. and Yu, T. and Thebaut, S. M. and Paulsen, "Identifying Error-prone Software - an Empirical Study", IEEE Trans Software Engr. SE-11 : 317-324, 1985

[Ein88] Einbu, J. M., "An Architectual Approach to Improved Program Maintainability", Software Practice & Experience, 18(1):51-62, Jan. 88

[Gun73] Gundermann, E. E., "A Glimpse into Program Maintenance", Datamantion, 19(6):99-105, June, 1973

[Gus85] Gustafson, D. A. and Melton, A. C. and Hsieh, S., "An Analysis of Software Changes during Maintenance", KSU, 1985

[Inc85] Ince, D. C. "A Program Design Language Based Software Maintenance Tool", Software Practice & Experience, 15(6):583-594, June 1985

[Lan83] Langergan, R. G. and Grasso, C. A., "Reusable Design and Code: A Strategy for Designing Software with Maintenance in Mind", pp.55-56, 1983 Software Maintenance Workshop

[Lie78] Lientz, B. P. and Swanson, E. B. and Tompkins, G. E., "Characteristics of Application Software Maintenance", Communication of the ACM, 21(6):466-471, June 1978

[Liu76] Liu, C., "A Look at Software Maintenance", Datamation, 22(11):51-55, Nov. 1976

[Rig69] Riggs, R., "Computer System Maintenance" , Datamation, 15(11)227-235, Nov. 1969

[Rub83] Rubin, H. A., "Macro and Micro-Estimation of Maintenance Effort: The ESTIMACS Maintenance Models", Hunter Colledge, New York, pp.78-79, 1983 Software Maintenance Workshop

[Sch87] Schneidewind, N. F. "The State of Software Maintenance", IEEE Trans on Software Engr., SE-3(3):303-355, Mar. 1987

[Swa76] Swanson, E. B., "The Dimention of Maintenance", Proc. 2nd International Conference. on Software Eng., San Francisco, pp.492-497, Oct. 1976

```
#
#   Shell program:  Maintain
#
#    The program calls the following subprogram:
#         maintain.aux          maintain.auxa
#         maintain.auxb         maintain.auxc
#         maintain.aux1
echo ANALYSIS FOR:  $1  $2 > $1.list
echo >> $1.list
echo -n $1' '$2' ' >> 0.stats
#
#   ######################################
#   #         Checking module            #
#   ######################################
#
#   checks to see if divisions are present in file one
awk '
BEGIN { iflag = 0; eflag = 0; dflag = 0; pflag = 0 }
/IDENTIFICATION DIVISION./ { iflag = 1 }
/ENVIRONMENT DIVISION./ { eflag = 1 }
/DATA DIVISION./ { dflag = 1 }
/PROCEDURE DIVISION/ { pflag = 1 }
END { if (iflag + eflag + dflag + pflag != 4) { print 5 > "err1" }}' $1
if (test -f err1)
then  exit 1
fi
# checks to see if divisions are present in file two
awk '
BEGIN { iflag = 0; eflag = 0; dflag = 0; pflag = 0 }
/IDENTIFICATION DIVISION./ { iflag = 1 }
/ENVIRONMENT DIVISION./ { eflag = 1 }
/DATA DIVISION./ { dflag = 1 }
/PROCEDURE DIVISION/ { pflag = 1 }
END { if (iflag + eflag + dflag + pflag != 4) { print 5 > "err2" }}' $2
if (test -f err2)
then exit 2
fi
#
#   ##########################################
#   #         Preprocessing  module          #
#   ##########################################
#
#    removes numbers from tail end of lines
awk '{printf "%s ",  $0;  printf "#%##\n" }' $1  |
sed 's/........#%##//' > $1.n
awk '{printf "%s ",  $0;  printf "#%##\n" }' $2  |
sed 's/........#%##//' > $2.n
#
# removes skips and ejects from file
sed '/ EJECT/ d                 :
     / SKIP/ d' $1.n > $1.nnn
sed '/ EJECT/ d
     / SKIP/ d' $2.n > $2.nnn
rm $1.n $2.n
#
#   install End of program marker on both files
echo EnD 0000000 > $1.temp
cat $1.temp >> $1.nnn
cat $1.temp >> $2.nnn
```

```
# Print out messages regarding the presence or absence of divisions
#  for file 1 the file has been preprocessed and appropriate
#  divisions inserted  without leading blanks
echo "LIST OF MISSING DIVISIONS FOR " $1 >> $1.list
#
echo >> $1.list
awk '
BEGIN { iddiv = 0; envdiv = 0; datadiv = 0; prodiv = 0 }
/ IDENTIFICATION DIVISION./ { iddiv++; next }
/ ENVIRONMENT DIVISION./ { envdiv++; next }
/ DATA DIVISION./ { datadiv++; next }
/ PROCEDURE DIVISION/ { prodiv++; next }
END {printf "%d %d %d %d ",iddiv,envdiv,datadiv,prodiv >> "0.stats";
     if (iddiv == 0) { print "**** IDENTIFICATION DIVISION MISSING **** };
     if (envdiv == 0) { print "**** ENVIRONMENT DIVISION MISSING **** };
     if (datadiv == 0) { print "**** DATA DIVISION MISSING **** };
     if (prodiv==0) {print "**** PROCEDURE DIVISION MISSING ****"}}' $1.nnn>>$1.list
echo >> $1.list
echo "END OF LIST" >> $1.list
echo >> $1.list
#
# print out messages regarding the presence or absence of divisions
#   for file2 the file has been preprocessed and appropriate divisions
#   inserted without leading blanks
echo "LIST OF MISSING DIVISIONS FOR " $2 >> $1.list
echo >> $1.list
awk '
BEGIN { iddiv = 0; envdiv = 0; datadiv = 0; prodiv = 0 }
/ IDENTIFICATION DIVISION./ { iddiv++; next }
/ ENVIRONMENT DIVISION./ { envdiv++; next }
/ DATA DIVISION./ { datadiv++; next }
/ PROCEDURE DIVISION/ { prodiv++; next }
END {printf "%d %d %d %d ",iddiv,envdiv,datadiv,prodiv >> "0.stats";
     if (iddiv == 0) { print "**** IDENTIFICATION DIVISION MISSING **** };
     if (envdiv == 0) { print "**** ENVIRONMENT DIVISION MISSING **** };
     if (datadiv == 0) { print "**** DATA DIVISION MISSING **** };
     if (prodiv==0) {print "**** PROCEDURE DIVISION MISSING ****"}}' $2.nnn>>$1.list
echo >> $1.list
echo "END OF LIST" >> $1.list
echo >> $1.list
#
#    ****************************************
#    *      Distinguishing module           *
#    ****************************************
#
#    flags statements in the environment division in file one
awk '/ENVIRONMENT DIVISION./,/DATA DIVISION./ {
if (($1 == "ENVIRONMENT" || $1 == "DATA") && $2 == "DIVISION.")
{ print $0; next }
else
{ printf "%s ", $0; printf "Env\n"; next } }
/./ { print $0 }' $1.nnn > $1.nnnn
rm $1.nnn

# flags statements in the environment division in file two
awk '/ENVIRONMENT DIVISION./,/DATA DIVISION./ {
if (($1 == "ENVIRONMENT" || $1 == "DATA") && $2 == "DIVISION.")
{ print $0; next }
else
{ printf "%s ", $0; printf "Env\n"; next } }
/./ { print $0 }' $2.nnn > $2.nnnn
rm $2.nnn
```

```
#
#   flags comment lines in file one
awk '/IDENTIFICATION DIVISION./,/ENVIRONMENT DIVISION./ {
if (($1 == "IDENTIFICATION" || $1 == "ENVIRONMENT") && $2 == "DIVISION.")
   { print $0; next }
else
   { printf "Comment   %s\n", $0; next } }
/./ { print $0 }' $1.nnnn > $1.nnnnn
rm $1.nnnn

#   flags comment lines in file two
awk '/IDENTIFICATION DIVISION./,/ENVIRONMENT DIVISION./ {
if (($1 == "IDENTIFICATION" || $1 == "ENVIRONMENT") && $2 == "DIVISION.")
   { print $0; next }
else
   { printf "Comment   %s\n", $0; next } }
/./ { print $0 }' $2.nnnn > $2.nnnnn
rm $2.nnnn
#
#  separates declarations into declaration part and initialization part
#    for file one
awk '
/DATA DIVISION./,/PROCEDURE DIVISION/ {
if (($1=="DATA" || $1=="PROCEDURE") && ($2=="DIVISION.") || (substr($1,1,1)=="*"))
{ print $0; next }
else
{ if ($1 == "FILE" || $1 == "WORKING-STORAGE" && $2 == "SECTION.")
  { printf "DeSetn    %s\n",$0; next }
  else
  { if (substr($1,1,1) ~ /[0-9]/) { printf "Dlrtn      " }
    if (substr($NF, length ($NF), 1) == ".")
    { i = 1
      while (i <= NF )
         { if (($i == "REDEFINES") || ($i == "RENAMES") || ($i == "VALUE"))
           { printf "\nDlrtn      " }
           printf "%s ", $i; i++ }
      printf "\n"; next
    }
    else
    { i = 1
      while ( i <= NF )
         { if (($i == "REDEFINES") || ($i == "RENAMES") || ($i == "VALUE"))
           { printf "\nDlrtn      " }
           printf "%s ", $i; i++ }
      next
    }
  }
} }
/./ { print $0 }' $1.nnnnn > $1.nn
rm $1.nnnnn

# separates declarations into declaration part and initialization part
#   for file two
awk '
/DATA DIVISION./,/PROCEDURE DIVISION/ {
if (($1=="DATA" || $1=="PROCEDURE") && ($2=="DIVISION.") || (substr($1,1,1)=="*"))
{ print $0; next }
else
{ if ($1 == "FILE" || $1 == "WORKING-STORAGE" && $2 == "SECTION.")
  { printf "DeSetn    %s\n",$0; next }
  else
  { if (substr($1,1,1) ~ /[0-9]/) { printf "Dlrtn      " }
```

```
    if (substr($NF, length ($NF), 1) == ".")
    { i = 1
      while (i <= NF )
         { if (($i == "REDEFINES") || ($i == "RENAMES") || ($i == "VALUE"))
            { printf "\nDlrtn      " }
           printf "%s ", $i; i++ }
      printf "\n"; next
    }
    else
    { i = 1
      while ( i <= NF )
         { if (($i == "REDEFINES") || ($i == "RENAMES") || ($i == "VALUE"))
            { printf "\nDlrtn      " }
           printf "%s ", $i; i++ }
      next
    }
  }
} }
/./ { print $0 }' $2.nnnnn > $2.nn
rm $2.nnnnn
#
#   break procedure division into statements
maintain.auxc $1.nn
mv $1.nn.modules $1.modules
mv $1.nn.calls $1.calls
maintain.auxc $2.nn
#
#  Printing the heading of the overall number for file one
#
echo '------------------------------------------------------------' >> $1.list
echo >> $1.list
echo 'OVERALL ANALYSIS OF STATEMENTS' >> $1.list
echo >> $1.list
#
#  Calculation module is embedded in maintain.aux1
#     compute number[TYPE] of each statement for file one and list the result
maintain.aux1 $1.nn >> $1.list
echo -n $1' '$2' ' >> 0.stats.totals
echo -n $1' '$2' ' >> 0.stats.alters
echo -n $1' '$2' ' >> 0.stats.deletes
echo -n $1' '$2' ' >> 0.stats.adds
cat auxtemp >> 0.stats.totals
echo >> 0.stats.totals
rm auxtemp
#
#         fix up divisions file one
awk '
/IDENTIFICATION DIVISION/ { printf "%s %s\n",$1,$2; next }
/ENVIRONMENT DIVISION/ { printf "%s %s\n",$1,$2; next }
/DATA DIVISION/ { printf "%s %s\n",$1,$2; next }
/PROCEDURE DIVISION/ { printf "%s %s\n",$1,$2; next }
/./ { print $0; next }' $1.nn > $1.na
rm $1.nn
mv $1.na $1.nn
#
# fix up divisions file two
awk '
/IDENTIFICATION DIVISION/ { printf "%s %s\n",$1,$2; next }
/ENVIRONMENT DIVISION/ { printf "%s %s\n",$1,$2; next }
/DATA DIVISION/ { printf "%s %s\n",$1,$2; next }
/PROCEDURE DIVISION/ { printf "%s %s\n",$1,$2; next }
/./ { print $0; next }' $2.nn > $2.na
rm $2.nn
mv $2.na $2.nn
#
```

```
#     ************************************
#     *       Difference module          *
#     ************************************
#
#   flag alters, deletions and addition in file one --> file two
diff -e $1.nn $2.nn  |grep '^[0-9]' |
sed 's/\,/ /g
     s/a/ a /g
     s/c/ c /g
     s/d/ d /g' |
awk '
BEGIN {printf "BEGIN {i=0}\n"}
NF==2 {if ($2 == "a")
        {printf "NR== %d {print \"a\",$0 ;i=1}\n",$1
         printf "NR== %d {print \"b\",$0 ;i=1}\n",($1+1) }
       else {printf "NR== %d {print \"%s\",$0 ;i=1}\n",$1,$2 }}
NF==3 {for (j=$1;j<=$2;j++)
      printf "NR== %d {print \"%s\",$0 ;i=1}\n",j,$3}
END {print "{if (i == 0) print \" \",$0 }\n{if (i!=0) i=0}"}' > anoth
awk -f anoth $1.nn > $1.r
rm anoth
#
# flag alters, deletions, and additions in file two --> file one
diff -e $2.nn $1.nn  |grep '^[0-9]' |
sed 's/\,/ /g
     s/a/ a /g
     s/c/ c /g
     s/d/ d /g' |
awk '
BEGIN {printf "BEGIN {i=0}\n"}
NF==2 {if ($2 == "a")
        {printf "NR== %d {print \"a\",$0 ;i=1}\n",$1
         printf "NR== %d {print \"b\",$0 ;i=1}\n",($1+1) }
       else {printf "NR== %d {print \"%s\",$0 ;i=1}\n",$1,$2 }}
NF==3 {for (j=$1;j<=$2;j++)
      printf "NR== %d {print \"%s\",$0 ;i=1}\n",j,$3}
END {print "{if (i == 0) print \" \",$0 }\n{if (i!=0) i=0}"}' > anoth
awk -f anoth $2.nn > $1.h
rm anoth
#
#     ************************************
#     *     Report  module               *
#     ************************************
#
maintain.aux $1.r >> $1.list
maintain.auxa $1.h >> $1.list

# if there were altered, then print out an analysis of those altered
if (test -f $1.r.c)
then
maintain.auxb $1.nn $2.nn >> $1.list
rm $1.r.c
fi

.



rm $1.nn $2.nn
# if there were deleted, then list the deleted statements
if (test -f delefile)
then
   echo '-------------------------------------------------------'  >> $1.list
   echo >> $1.list
```

```
    echo 'LIST THE DELETED STATEMENTS'  >> $1.list
    echo  >> $1.list
    cat  delefile >> $1.list
    rm  delefile
    echo >> $1.list
fi
#  if there were added, then list the added statements
if (test -f addsfile)
then
  echo '-------------------------------------------------------'  >> $1.list
  echo >> $1.list
  echo 'LIST THE ADDED STATEMENTS'  >> $1.list
  echo >> $1.list
  cat  addsfile >> $1.list
  rm  addsfile
  echo >> $1.list
fi
rm   0.stats
rm   0.stats.*
rm   $1.*
rm   $2.nn.*
# eof: maintain
```

```
#
#  program:  maintain.aux
#
# tabulating number of addition sections, deletions, altered
#     create a new file "ctemp", "alterfile" to store altered statements
#     create a new file "dtemp", "delefile" to store deleted statements
awk '
BEGIN { A = 0; C = 0; D = 0
}
/a/ { A++ }
/c/ { print $0 > "ctemp"; C++ }
/d/ { print $0 > "dtemp"; D++ }
END { printf "\n##########################################################\n\n";
      printf "TOTAL NUMBER OF ADDED SECTIONS IS: %d \n",A ;
      printf "%d %d %d ",A,C,D >> "0.stats";
      printf "TOTAL NUMBER OF ALTERED: %d \n",C;
      printf "TOTAL NUMBER OF DELETIONS: %d \n",D;
      printf "\n##########################################################\n\n\n"}' $1
echo '----------------------------------------------------------'
echo
echo 'ANALYSIS OF STATEMENTS ALTERED'
# if there are altered
#     remove c in front -- for uniform treatment
#     tabulate numbers of each statement type via maintain.aux1
if (test -f ctemp)
then
  sed 's/c/ /' ctemp > alterfile
  #
  #  compute Number[TYPE] of each statement in altered file and list result
  maintain.aux1 alterfile
  cat auxtemp >> 0.stats.altered
  echo "altered" > $1.c
  rm ctemp alterfile auxtemp
else
  echo '*** NO STATEMENTS ALTERED ***'
fi
echo
echo
echo '----------------------------------------------------------'
echo
echo 'ANALYSIS OF STATEMENTS DELETED'
echo
# if there are deletions
#     remove d in front -- for uniform treatment
#     tabulate numbers of each statement type via maintain.aux1
if (test -f dtemp)
then
  sed 's/d/ /' dtemp > delefile
  #
  #  compute Number[TYPE] of each statement for delefile and list result
  maintain.aux1 delefile
  cat auxtemp >> 0.stats.deletes
  rm dtemp auxtemp
else
  echo '*** NO STATEMENTS DELETED ***'
fi
echo
echo '----------------------------------------------------------'
echo
echo 'ANALYSIS OF STATEMENTS ADDED'
echo
# eof: maintain.aux
```

```
# program:  maintain.auxa
#   tabulate the number of added statements via flipping of files
awk '
BEGIN { D = 0 }
/d/ { print $0 > "dtemp"; D++ }
END { printf "TOTAL NUMBER OF ADDED STATEMENTS: %d\n\n", D;
      printf "%d\n", D >> "0.stats"}' $1
# if there were additions
#    then remove d from front
#    tabulate number of each statement type via maintain.aux1
if (test -f dtemp)
then
   sed 's/d/ /' dtemp > addsfile
   #
   #  compute Number[TYPE] of each statement in added file
   maintain.aux1 addsfile
   cat auxtemp >> 0.stats.adds
   rm dtemp auxtemp
fi
# eof:  maintain.auxa
```

```
# program:  maintain.auxb
#   processes altered, listing original statement, and new statement
echo '-------------------------------------------------------'
echo
echo 'ANALYSIS OF ALTERED STATEMENTS'
echo
echo
diff -l $1 $2 |
awk '
/[a|d]/ {flag = 0}
/c/ {flag = 1; print $0}
/</,/[c|d|a]/ {if (flag == 1) print $0}'  |
awk '
/c/ {printf "\n\n"; next}
/</ {print "ORIGINAL LINE ", $0; next }
/>/ { print "NEW LINE ", $0; next }
/---/ {if (NF == 1) { printf "\nALTERED TO \n\n" }; next}'
#  eof:  maintain.auxb
```

```
# program: maintain.auxc
#
#   process the procedure division splitting statements up
#     split into two temporary files
awk '/PROCEDURE DIVISION/,/EnD/ { print $0 >> "last.part"; next }
/./ { print $0 >> "first.part" }' $1
rm $1

# place a q in front of all keywords we are looking at
sed '
s/ MOVE / qMOVE /g
s/ ADD / qADD /g
s/ SUBTRACT / qSUBTRACT /g
s/ MULTIPLY / qMULTIPLY /g
s/ DIVIDE / qDIVIDE /g
s/ COMPUTE / qCOMPUTE /g
s/ IF / qIF /g
s/ ELSE / qqELSE /g
s/ ON / qON /g
s/ AT END / qAT END /g
s/ CALL / qCALL /g
s/ PERFORM / qPERFORM /g
s/ GO / qGO /g
s/ ALTER / qALTER /g
s/ NEXT / qNEXT /g
s/ EXIT / qEXIT /g
s/ STOP / qSTOP /g
s/ COPY / qCOPY /g
s/ DELETE / qDELETE /g
s/ DISPLAY / qDISPLAY /g
s/ OPEN / qOPEN /g
s/ CLOSE / qCLOSE /g
s/ READ / qREAD /g
s/ REWRITE / qREWRITE /g
s/ WRITE / qWRITE /g
s/ ACCEPT / qACCEPT /g
s/ SEARCH / qSEARCH /g
s/ SORT / qSORT /g
s/ SET / qSET /g
s/ GOBACK / qGOBACK /g
s/ EXEC CICS/ qEXEC CICS/g
s/ TRANSFORM/ qTRANSFORM/g
s/ EXAMINE / qEXAMINE /g
s/ INSPECT / qINSPECT /g' last.part >> last.part.n
rm last.part

# split up into keyword per line
awk '
BEGIN { line = 0;}
/EnD/ { printf "\n"; next }
{ if (substr($1,1,1) == "q") { if (line == 1) { printf "\n" }
                               printf "%s\n",$0; line = 0 }
  else


  { i = 1
    while (i <= NF)
    { if (substr ($i,1,1) == "q")
        { if (line == 1) { printf "\n" }
          if (substr ($i,2,1) == "q")
          { printf "%s ",substr($i,3,length ($1)-2); line = 0 }
```

```
          else
          { printf "%s ",substr($i,2,length ($i)-1); line = 1 }
        }
      else { printf "%s ",$i; line = 1 }
      i++
    }
    if (substr($NF,length ($NF),1) == ".") { printf "\n"; line = 0 }
  }
} ' last.part.n > last.part
rm last.part.n

# remove q in comment
sed '
s/q//g' last.part > last.part.a
rm last.part
mv last.part.a last.part

# indent for if nesting levels printable version
awk 'BEGIN { level = 0 }
NF == 1 && substr($NF,length ($NF),1) == "." && substr($NF,1,1) ~ /[0-9]/ { level = 0 }
/./ { if (level != 0)
        { i = 1
          while (i <= level)
          { printf "    "; i++ }
        }
    }
/IF / { print $0
        level++
        if ((substr ($NF,length ($NF),1) == ".") && (level > 0)) { level-- }
        next
      }
/./ { print $0
      if ((substr ($NF,length ($NF),1) == ".") && (level > 0))
      { level = 0 } }' last.part > last.part.n
rm last.part

# return in original file
cat first.part last.part.n > $1

# create files to be used to hand generate the hierarchy diagram
maintain.auxm last.part.n
mv calls $1.calls
mv modules $1.modules
rm first.part last.part.n
# eof:  maintain.auxc
```

```
# program:  maintain.aux1
#
#  compute numbers of statement for each type
#   and tabulate the actual number of individual statement types
awk '
BEGIN { com_cnt = 0; r_com_cnt = 0; nl_com_cnt = 0;
        other_cnt = 0; cics_cnt = 0; goback_cnt = 0; transform_cnt = 0;
        tdeq_cnt = 0; deq_cnt = 0; dseq_cnt = 0; fd_cnt = 0;
# value is counted in both tdeq_cnt and assign_cnt
        value_cnt = 0; redefines_cnt = 0; renames_cnt = 0;
        assign_cnt = 0; move_cnt = 0; add_cnt = 0; divide_cnt = 0;
          multiply_cnt = 0; subtract_cnt = 0; compute_cnt = 0;
        branch_cnt = 0; next_cnt = 0; exit_cnt = 0; stop_cnt = 0;
          goto_cnt = 0; alter_cnt = 0; perform_cnt = 0; call_cnt = 0;
        I_O_cnt = 0; copy_cnt = 0; delete_cnt = 0; display_cnt = 0;
          open_cnt = 0; close_cnt = 0; read_cnt = 0; write_cnt = 0;
          rewrite_cnt = 0; accept_cnt = 0;
        examine_cnt = 0; inspect_cnt = 0;
        search_cnt = 0; sort_cnt = 0; set_cnt = 0;
        cond_cnt = 0; if_cnt = 0; else_cnt = 0; on_cnt = 0; onsize_cnt = 0;
        env_cnt = 0; configuration_cnt = 0; in_out_cnt = 0; file_cont_cnt = 0;
          select_cnt = 0; special_cnt = 0; sp_is_cnt = 0; scomp_cnt = 0;
          ocomp_cnt = 0; ibm_cnt = 0;
        label_cnt = 0; at_end_cnt = 0
}

NF == 1 && substr ($NF,length($NF),1) == "." && substr ($NF,1,1) ~  /[0-9]/
   { label_cnt++ }
# skip divisions
/ DIVISION/ { next }

# comments
substr ($1,1,1) == "*" { com_cnt++
                         if (NF == 1) { nl_com_cnt++ }
                         next }
/Comment/ { r_com_cnt++; com_cnt ++; next }

# environment section statements
/Env/ { if ($1 == "CONFIGURATION") { configuration_cnt++; env_cnt++; next }
        else { if ($1 == "INPUT-OUTPUT") { in_out_cnt++; env_cnt++; next }
        else { if ($1 == "FILE-CONTROL.") { file_cont_cnt++; env_cnt++; next }
        else { if ($1 == "SPECIAL-NAMES.") { special_cnt++; env_cnt++; next }
        else { if ($1 == "SOURCE-COMPUTER.") { scomp_cnt++; env_cnt++; next }
        else { if ($1 == "OBJECT-COMPUTER.") { ocomp_cnt++; env_cnt++; next }
        else { if ($1 == "IBM-3083.") { ibm_cnt++; env_cnt++; next }
        else { if ($1 == "IBM-370.") { ibm_cnt++; env_cnt++; next }
        else { if ($1 == "SELECT") { select_cnt++; env_cnt++; next }
        else { if ($2 == "IS") { sp_is_cnt++; env_cnt++; next }}}}}}}}}}}
 # declarations
 /Dlrtn/ { if ($2 == "FD") { fd_cnt++; tdeq_cnt++; next }
           else { if ($2 == "VALUE") { assign_cnt++; value_cnt++; tdeq_cnt++; next }
           else { if ($2 == "REDEFINES") { redefines_cnt ++; tdeq_cnt++; next }
           else { if ($2 == "RENAMES") { renames_cnt ++; tdeq_cnt++; next }
           else { deq_cnt ++; tdeq_cnt++; next } } } } }
 /DeSetn/ { dseq_cnt ++; tdeq_cnt++; next }

 /ELSE / { cond_cnt++; else_cnt++; next }
 # assignment
 /MOVE / { assign_cnt++; move_cnt++; next }
 /ADD / { assign_cnt++; add_cnt++; next }
 /SUBTRACT / { assign_cnt++; subtract_cnt++; next }
 /MULTIPLY / { assign_cnt++; multiply_cnt++; next }
 /DIVIDE / { assign_cnt++; divide_cnt++; next }
 /COMPUTE / { assign_cnt++; compute_cnt++; next }
```

```
    # conditionals
    /IF / { cond_cnt++; if_cnt++; next }
    /ON SIZE ERROR / { cond_cnt++; onsize_cnt++; next }
    /ON / { cond_cnt++; on_cnt++; next; }
    /AT END / { cond_cnt++; at_end_cnt++; next; }

    # looping -- branching
    /EXIT / { branch_cnt++; exit_cnt++; next }
    /CALL / { branch_cnt++; call_cnt++; next }
    /PERFORM / { branch_cnt++; perform_cnt++; next }
    /GO TO / { branch_cnt++; goto_cnt++; next }
    /NEXT /  { branch_cnt++; next_cnt++; next }
    /STOP / { branch_cnt++; stop_cnt++; next }

    # input-output
    /DELETE / { I_O_cnt++; delete_cnt++; next }
    /DISPLAY / { I_O_cnt++; display_cnt++; next }
    /OPEN / { I_O_cnt++; open_cnt++; next }
    /CLOSE / { I_O_cnt++; close_cnt++; next }
    /READ / { I_O_cnt++; read_cnt++; next }
    /REWRITE / { I_O_cnt++; rewrite_cnt++; next }
    /WRITE / { I_O_cnt++; write_cnt++; next }
    /ACCEPT / { I_O_cnt++; accept_cnt++; next }

    # other
    /COPY / { other_cnt++; copy_cnt++; next }
    /ALTER / { other_cnt++; alter_cnt++; next }
    /SEARCH / { search_cnt++; other_cnt ++; next }
    /SORT / { sort_cnt++; other_cnt ++; next }
    /SET / { set_cnt++; other_cnt ++; next }
    /TRANSFORM / { other_cnt++; transform_cnt++; next }
    /EXAMINE / { other_cnt++; examine_cnt++; next }
    /INSPECT / { other_cnt ++; inspect_cnt++; next }
    /EXEC CICS/ { cics_cnt ++; other_cnt ++; next }
    /GOBACK/ { goback_cnt ++; other_cnt ++; next }
END { u_cs = com_cnt - r_com_cnt - nl_com_cnt;
      print "NUMBER OF LINES OF COMMENTS : ",com_cnt;
      print "     IDENTIFICATION DIVISION : ",r_com_cnt;
      print "     SPACING PURPOSES        : ",nl_com_cnt;
      print "     USEFUL COMMENTS         : ",u_cs;
      printf "%d %d %d %d ",com_cnt,r_com_cnt,nl_com_cnt,u_cs >> "auxtemp";
      print " ";
      print "NUMBER OF ENVIRONMENT STATEMENTS  : ",env_cnt;
      print "          CONFIGURATION SECTION   : ",configuration_cnt;
      print "          SOURCE-COMPUTER         : ",scomp_cnt;
      print "          OBJECT-COMPUTER         : ",ocomp_cnt;
      print "          COMPUTER SPECIFICATION  : ",imb_cnt;
      print "          SPECIAL NAMES           : ",special_cnt;
      print "          SPECIAL NAME ASSIGNMENT : ",sp_is_cnt;
      print "          INPUT-OUTPUT SECTION    : ",in_out_cnt;
      print "          FILE-CONTROL            : ",file_cont_cnt;
      print "          SELECT                  : ",select_cnt;
      printf "%d ",env_cnt >> "auxtemp";
      printf "%d ",configuration_cnt >> "auxtemp";
      printf "%d ",scomp_cnt >> "auxtemp";
      printf "%d ",ocomp_cnt >> "auxtemp";
      printf "%d ",imb_cnt >> "auxtemp";
      printf "%d ",special_cnt >> "auxtemp";
      printf "%d ",sp_is_cnt >> "auxtemp";
      printf "%d ",in_out_cnt >> "auxtemp";
      printf "%d ",file_cont_cnt >> "auxtemp";
      printf "%d ",select_cnt >> "auxtemp";
      print " ";
      print "NUMBER OF DECLARATIONS  : ",tdec_cnt;
```

```
print "           SECTIONS       : ",dsec_cnt;
print "           FD             : ",fd_cnt;
print "           DECLARATIONS   : ",dec_cnt;
print "           VALUE CLAUSES  : ",value_cnt;
print "           REDEFINES      : ",redefines_cnt;
print "           RENAMES        : ",renames_cnt;
printf "%d ",tdec_cnt >> "auxtemp";
printf "%d ",dsec_cnt >> "auxtemp";
printf "%d ",fd_cnt >> "auxtemp";
printf "%d ",dec_cnt >> "auxtemp";
printf "%d %d %d ",value_cnt,redefines_cnt,renames_cnt >> "auxtemp";
print " ";
print "NUMBER OF ASSIGNMENTS : ",assign_cnt;
print " *** note that the above total includes VALUES CLAUSES ***";
print "           MOVE           : ",move_cnt;
print "           ADD            : ",add_cnt;
print "           SUBTRACT       : ",subtract_cnt;
print "           MULTIPLY       : ",multiply_cnt;
print "           DIVIDE         : ",divide_cnt;
print "           COMPUTE        : ",compute_cnt;
printf "%d %d %d %d ",assign_cnt,move_cnt,add_cnt,subtract_cnt >> "auxtemp";
printf "%d %d %d ",multiply_cnt,divide_cnt,compute_cnt >> "auxtemp";
print " ";
print "NUMBER OF CONDITIONALS  : ",cond_cnt;
print "           IF             : ",if_cnt;
print "           ELSE           : ",else_cnt;
print "           ON             : ",on_cnt;
print "           ON SIZE ERROR  : ",onsize_cnt;
print "           AT END         : ",at_end_cnt;
printf "%d %d %d %d ",cond_cnt ,if_cnt ,else_cnt ,on_cnt >> "auxtemp";
printf "%d %d ",onsize_cnt,at_end_cnt >> "auxtemp";
print " ";
print "NUMBER OF BRANCHINGS : ",branch_cnt;
print "           CALL           : ",call_cnt;
print "           PERFORM        : ",perform_cnt;
print "           GO TO          : ",goto_cnt;
print "           NEXT           : ",next_cnt;
print "           EXIT           : ",exit_cnt;
print "           STOP           : ",stop_cnt;
printf "%d %d %d %d ",branch_cnt,call_cnt,perform_cnt,goto_cnt >> "auxtemp";
printf "%d %d %d",next_cnt,exit_cnt,stop_cnt >> "auxtemp";
print "NUMBER OF INPUT/OUTPUT : ",i_O_cnt;
print "           DELETE         : ",delete_cnt;
print "           DISPLAY        : ",display_cnt;
print "           OPEN           : ",open_cnt;
print "           CLOSE          : ",close_cnt;
print "           READ           : ",read_cnt;
print "           REWRITE        : ",rewrite_cnt;
print "           WRITE          : ",write_cnt;
print "           ACCEPT         : ",accept_cnt;
printf "%d ",i_O_cnt >> "auxtemp";
printf "%d %d %d %d ",delete_cnt,display_cnt,open_cnt,close_cnt >> "auxtemp";
printf "%d %d %d %d ",read_cnt,rewrite_cnt,write_cnt,accept_cnt >> "auxtemp";
print " ";
print "NUMBER OF LABELS : ",label_cnt;
printf "%d ",label_cnt >> "auxtemp";
print "NUMBER OF OTHER STATEMENTS : ",otner_cnt;
print "           COPY           : ",copy_cnt;
print "           ALTER          : ",alter_cnt;
print "           TRANSFORM      : ",transform_cnt;
print "           EXAMINE        : ",examine_cnt;
print "           INSPECT        : ",inspect_cnt;
printf "%d %d ",other_cnt,copy_cnt >> "auxtemp";
printf "%d %d ",alter_cnt,transform_cnt >> "auxtemp";
```

```
        printf "%d ",examine_cnt >> "auxtemp";
        printf "%d ",inspect_cnt >> "auxtemp";
        printf "%d ",search_cnt >> "auxtemp";
        printf "%d ",sort_cnt >> "auxtemp";
        printf "%d ",set_cnt >> "auxtemp";
        printf "%d ",cics_cnt >> "auxtemp";
        printf "%d",goback_cnt >> "auxtemp";
        print "          SEARCH          : ",search_cnt;
        print "          SORT            : ",sort_cnt;
        print "          SET             : ",set_cnt;
        print "          CICS EXEC       : ",cics_cnt;
        print "          GOBACK          : ",goback_cnt }' $1
# eof:  maintain.aux1
```

```
IDENTIFICATION DIVISION.                                                4840002
    PROGRAM-ID. XXXXX                                                   4840003
    AUTHOR. XXXXX, XXXXX                                                4840004
    VERSION 1.                                                          4840005
    INSTALLATION.  XXXXXXXXXXXX                                         4840006
    DATE-WRITTEN. APRIL 1984                                            4840007
    *  PROGRAM WAS WRITTEN FROM  PRG JEX 10-8-84 MVS                    4840008
    *  CONVERSION, MODIFIED SELECT CLAUSE                               4840009
ENVIRONMENT DIVISION.                                                   4840011
CONFIGURATION SECTION.                                                  4840012
SOURCE-COMPUTER. IBM-370.                                               4840013
OBJECT-COMPUTER. IBM-370.                                               4840014
INPUT-OUTPUT SECTION.                                                   4840015
FILE-CONTROL.                                                           4840016
    SELECT PRINT-FILE ASSIGN SYS003-UR-1403-S.                          4840017
    SELECT CARD-FILE ASSIGN SYS004-UR-2520R-S.                          4840018
                                                                        4840019
DATA DIVISION.                                                          4840020
FILE SECTION.                                                           4840021
                                                                        4840022
FD  PRINT-FILE LABEL RECORDS ARE OMITTED                                4840023
    REPORT IS REPORT-DETAIL.                                            4840024
                                                                        4840025
FD  CARD-FILE LABEL RECORDS ARE OMITTED.                                4840026
01  CARD-REC          PIC X(80).                                        4840027
                                                                        4840028
WORKING-STORAGE SECTION.                                                4840029
                                                                        4840030
01  WRK-REC.                                                           4840031
    02 C-FILL PIC X(5).                                                 4840032
    02 FILLER REDEFINES C-FILL.                                         4840033
       03 FILLER PIC X.                                                 4840034
          88 DOLLAR-SIGN VALUE '$'.                                     4840035
       03 FILLER PIC X(4).                                              4840036
    02 C-DATE REDEFINES C-FILL PIC 9(5).                                4840037
    02 C-REF  PIC X(6).                                                 4840038
    02 FILLER PIC X(5).                                                 4840039
    02 C-ACTA PIC X(4).                                                 4840040
    02 C-ACTB PIC X(3).                                                 4840041
    02 C-ACTC PIC X(3).                                                 4840042
    02 FILLER PIC X.                                                    4840043
    02 C-AMNTX  PIC X(11).                                              4840044
    02 C-AMNT REDEFINES C-AMNTX PIC 9(9)V99.                            4840045
    02 C-AMNTSIGN PIC X.                                                4840046
    02 FILLER PIC X(8).                                                 4840047
    02 C-TYPE PIC X.                                                    4840048
    02 FILLER PIC X(12).                                                4840049
    02 C-VEND PIC X(5).                                                 4840050
    02 C-DESC PIC X(15).                                                4840051
                                                                        4840052
01  AMT                    PIC S9(9)V99 VALUE ZERO.                     4840053
                                                                        4840054
REPORT SECTION.                                                         4840055
RD  REPORT-DETAIL                                                       4840056
    PAGE LIMIT IS 65 LINES                                              4840057
    HEADING 1                                                           4840058
    FIRST DETAIL 4.                                                     4840059
01  PAGE-HEADER TYPE IS PAGE HEADING.                                  4840060
    05  LINE NUMBER IS 1.                                              4840061
      10  COLUMN  02 PIC X(6) VALUE 'ED0283'.                          4840062
      10  COLUMN  30 PIC X(21) VALUE 'DISTRIBUTION KICKOUTS'.          4840063
      10  COLUMN  78 PIC X(4) VALUE 'DATE'.                            4840064
```

```
     10  COLUMN  83 PIC X(8) SOURCE CURRENT-DATE.
   05  LINE NUMBER IS PLUS 1.
     10  COLUMN  78 PIC X(4) VALUE 'PAGE'.
     10  COLUMN  82 PIC Z(4) SOURCE PAGE-COUNTER.
   05  LINE NUMBER IS PLUS 1.
     10  COLUMN 03 PIC X(4) VALUE 'DATE'.
     10  COLUMN 12 PIC X(6) VALUE 'REF NO'.
     10  COLUMN 23 PIC X(7) VALUE 'ACCT NO'.
     10  COLUMN 57 PIC X(3) VALUE 'AMT'.
     10  COLUMN 67 PIC X(4) VALUE 'TYPE'.
     10  COLUMN 74 PIC X(6) VALUE 'VENDOR'.
     10  COLUMN 85 PIC X(11) VALUE 'DESCRIPTION'.
   05  LINE NUMBER IS PLUS 1.
     10 COLUMN 20 PIC X(10) VALUE SPACES.

01  REPORT-LINE TYPE IS DETAIL.
    05  LINE NUMBER IS PLUS 2.
      10 COLUMN 02 PIC Z9B99B99 SOURCE C-DATE.
      10 COLUMN 04 PIC X VALUE '-'.
      10 COLUMN 07 PIC X VALUE '-'.
      10 COLUMN 12 PIC X(6) SOURCE C-REF.
      10 COLUMN 20 PIC X(4) SOURCE C-ACTA.
      10 COLUMN 24 PIC X VALUE '-'.
      10 COLUMN 25 PIC X(3) SOURCE C-ACTB.
      10 COLUMN 28 PIC X VALUE '-'.
      10 COLUMN 29 PIC X(3) SOURCE C-ACTC.
      10 COLUMN 51 PIC ZZZ,ZZZ,ZZ9.99- SOURCE AMT.
      10 COLUMN 68 PIC X SOURCE C-TYPE.
      10 COLUMN 74 PIC X(5) SOURCE C-VEND.
      10 COLUMN 83 PIC X(15) SOURCE C-DESC.


01  ERROR-DETAIL TYPE IS DETAIL.
    05  LINE NUMBER IS PLUS 3.
      10 COLUMN 02 PIC X(9) VALUE '****ERROR'.
      10 COLUMN 10 PIC X(80) SOURCE WRK-REC.

PROCEDURE DIVISION.

    OPEN  OUTPUT PRINT-FILE
          INPUT CARD-FILE.
    INITIATE REPORT-DETAIL.

1000-LOOP.
    READ CARD-FILE INTO WRK-REC AT END GO TO 3000-EOF.
    IF DOLLAR-SIGN
      GO TO 1000-LOOP.
    EXAMINE C-AMNTX REPLACING ALL SPACES BY ZERO.
    * IF C-AMNTX NOT NUMERIC
    *   GENERATE ERROR-DETAIL
    *   MOVE ZERO TO C-AMNTX.
    IF C-AMNTSIGN EQUAL '-'
      COMPUTE AMT EQUAL C-AMNT * -1
    ELSE
      MOVE C-AMNT TO AMT.
    GENERATE REPORT-LINE.
    MOVE ZERO TO C-AMNT.
    GO TO 1000-LOOP.
3000-EOF.
    TERMINATE REPORT-DETAIL.
    CLOSE CARD-FILE PRINT-FILE
    STOP RUN.
```

```
IDENTIFICATION DIVISION.                                                4840002
     PROGRAM-ID. XXXXX                                                  4840003
     AUTHOR. XXXXX, XXXXX                                               4840004
     VERSION 2.                                                         4840005
     INSTALLATION.  XXXXXXXXXXX                                         4840006
     DATE-WRITTEN. APRIL 1984                                           4840007
ENVIRONMENT DIVISION.                                                   4840012
CONFIGURATION SECTION.                                                  4840013
SOURCE-COMPUTER. IBM-370.                                               4840014
OBJECT-COMPUTER. IBM-370.                                               4840015
INPUT-OUTPUT SECTION.                                                   4840016
FILE-CONTROL.                                                           4840017
     SELECT PRINT-FILE ASSIGN UT-P1L.                                   4840018
     SELECT CARD-FILE ASSIGN UT-CARDIN.                                 4840019
                                                                        4840020
DATA DIVISION.                                                          4840021
FILE SECTION.                                                           4840022
                                                                        4840023
FD   PRINT-FILE LABEL RECORDS ARE OMITTED                               4840024
     REPORT IS REPORT-DETAIL.                                           4840025
                                                                        4840026
FD   CARD-FILE LABEL RECORDS ARE OMITTED.                               4840027
01   CARD-REC          PIC X(80).                                       4840028
                                                                        4840029
WORKING-STORAGE SECTION.                                                4840030
                                                                        4840031
01   WRK-REC.                                                           4840032
     02 C-FILL PIC X(5).                                                4840033
     02 FILLER REDEFINES C-FILL.                                        4840034
        03 FILLER PIC X.                                                4840035
           88 DOLLAR-SIGN VALUE '$'.                                    4840036
        03 FILLER PIC X(4).                                             4840037
     02 C-DATE REDEFINES C-FILL PIC 9(5).                               4840038
     02 C-REF  PIC X(6).                                                4840039
     02 FILLER PIC X(5).                                                4840040
     02 C-ACTA PIC X(4).                                                4840041
     02 C-ACTB PIC X(3).                                                4840042
     02 C-ACTC PIC X(3).                                                4840043
     02 FILLER PIC X.                                                   4840044
     02 C-AMNTX  PIC X(11).                                             4840045
     02 C-AMNT REDEFINES C-AMNTX PIC 9(9)V99.                           4840046
     02 C-AMNTSIGN PIC X.                                               4840047
     02 FILLER PIC X(8).                                                4840048
     02 C-TYPE PIC X.                                                   4840049
     02 FILLER PIC X(12).                                               4840050
     02 C-VEND PIC X(5).                                                4840051
     02 C-DESC PIC X(15).                                               4840052
                                                                        4840053
01   AMT                    PIC S9(9)V99 VALUE ZERO.                    4840054
                                                                        4840055
REPORT SECTION.                                                         4840056
RD   REPORT-DETAIL                                                      4840057
     PAGE LIMIT IS 65 LINES                                             4840058
     HEADING 1                                                          4840059
     FIRST DETAIL 4.                                                    4840060
01   PAGE-HEADER TYPE IS PAGE HEADING.                                  4840061
     05  LINE NUMBER IS 1.                                             4840062
       10  COLUMN  02 PIC X(6) VALUE 'ED0283'.                         4840063
       10  COLUMN  30 PIC X(21) VALUE 'DISTRIBUTION KICKOUTS'.         4840064
       10  COLUMN  78 PIC X(4) VALUE 'DATE'.                           4840065
       10  COLUMN  83 PIC X(8) SOURCE CURRENT-DATE.                    4840066
```

```
      05  LINE NUMBER IS PLUS 1.                                          4 8A0067
        10  COLUMN  78 PIC X(4) VALUE 'PAGE'.                             4 8A0068
        10  COLUMN  82 PIC Z(4) SOURCE PAGE-COUNTER.                      4 8A0069
      05  LINE NUMBER IS PLUS 1.                                          4 8A0070
        10  COLUMN 03 PIC X(4) VALUE 'DATE'.                              4 8A0071
        10  COLUMN 12 PIC X(6) VALUE 'REF NO'.                            4 8A0072
        10  COLUMN 23 PIC X(7) VALUE 'ACCT NO'.                           4 8A0073
        10  COLUMN 57 PIC X(3) VALUE 'AMT'.                               4 8A0074
        10  COLUMN 67 PIC X(4) VALUE 'TYPE'.                              4 8A0075
        10  COLUMN 74 PIC X(6) VALUE 'VENDOR'.                            4 8A0076
        10  COLUMN 85 PIC X(11) VALUE 'DESCRIPTION'.                      4 8A0077
      05  LINE NUMBER IS PLUS 1.                                          4 8A0078
        10 COLUMN 20 PIC X(10) VALUE SPACES.                             4 8A0079

  01  REPORT-LINE TYPE IS DETAIL.                                         4 8A0080
      05  LINE NUMBER IS PLUS 2.                                          4 8A0081
        10 COLUMN 02 PIC Z9B99B99 SOURCE C-DATE.                         4 8A0082
        10 COLUMN 04 PIC X VALUE '-'.                                    4 8A0083
        10 COLUMN 07 PIC X VALUE '-'.                                    4 8A0084
        10 COLUMN 12 PIC X(6) SOURCE C-REF.                              4 8A0085
        10 COLUMN 20 PIC X(4) SOURCE C-ACTA.                             4 8A0086
        10 COLUMN 24 PIC X VALUE '-'.                                    4 8A0087
        10 COLUMN 25 PIC X(3) SOURCE C-ACTB.                             4 8A0088
        10 COLUMN 28 PIC X VALUE '-'.                                    4 8A0089
        10 COLUMN 29 PIC X(3) SOURCE C-ACTC.                             4 8A0090
        10 COLUMN 51 PIC ZZZ,ZZZ,ZZ9.99- SOURCE AMT.                     4 8A0091
        10 COLUMN 68 PIC X SOURCE C-TYPE.                                4 8A0092
        10 COLUMN 74 PIC X(5) SOURCE C-VEND.                             4 8A0093
        10 COLUMN 83 PIC X(15) SOURCE C-DESC.                            4 8A0094
                                                                         4 8A0095
                                                                         4 8A0096
  01  ERROR-DETAIL TYPE IS DETAIL.                                        4 8A0097
      05  LINE NUMBER IS PLUS 3.                                          4 8A0098
        10 COLUMN 02 PIC X(9) VALUE '****ERROR'.                         4 8A0099
        10 COLUMN 10 PIC X(80) SOURCE WRK-REC.                           4 8A0100
                                                                         4 8A0101
  PROCEDURE DIVISION.                                                    4 8A0102
                                                                         4 8A0103
      OPEN OUTPUT PRINT-FILE                                             4 8A0104
           INPUT CARD-FILE.                                              4 8A0105
      INITIATE REPORT-DETAIL.                                            4 8A0106
                                                                         4 8A0107
  1000-LOOP.                                                             4 8A0108
      READ CARD-FILE INTO WRK-REC AT END GO TO 3000-EOF.                 4 8A0109
      IF DOLLAR-SIGN                                                     4 8A0110
         GO TO 1000-LOOP.                                                4 8A0111
      *                                                                  4 8A0112
      *                                                                  4 8A0000
      *                                                                  4 8A0001
      EXAMINE C-AMNTX REPLACING ALL SPACES BY ZERO.                      4 8A0002
      IF C-AMNTX NOT NUMERIC                                             4 8A0113
         GENERATE ERROR-DETAIL                                           4 8A0114
         MOVE ZERO TO C-AMNTX.                                           4 8A0115
      IF C-AMNTSIGN EQUAL '-'                                            4 8A0116
         COMPUTE AMT EQUAL C-AMNT * -1                                   4 8A0117
      ELSE                                                               4 8A0118
         MOVE C-AMNT TO AMT.                                             4 8A0119
      GENERATE REPORT-LINE.                                              4 8A0120
      MOVE ZERO TO C-AMNT.                                               4 8A0121
      GO TO 1000-LOOP.                                                   4 8A0122
  3000-EOF.                                                              4 8A0123
      TERMINATE REPORT-DETAIL.                                           4 8A0124
      CLOSE CARD-FILE PRINT-FILE                                         4 8A0125
      STOP RUN.                                                          4 8A0126
                                                                         4 8A0127
```

Appendix D. Result from Running Maintain

```
ANALYSIS FOR: COBOL.1 COBOL.2

LIST OF MISSING DIVISIONS FOR  COBOL.1

END OF LIST

LIST OF MISSING DIVISIONS FOR  COBOL.2

END OF LIST
-----------------------------------------------------

OVERALL ANALYSIS OF STATEMENTS


NUMBER OF LINES OF COMMENTS :  10
    IDENTIFICATION DIVISION :  7
    SPACING PURPOSES        :  0
    USEFUL COMMENTS         :  3

NUMBER OF ENVIRONMENT STATEMENTS  :  7
          CONFIGURATION SECTION   :  1
          SOURCE-COMPUTER         :  1
          OBJECT-COMPUTER         :  1
          COMPUTER SPECIFICATION  :  0
          SPECIAL NAMES           :  0
          SPECIAL NAME ASSIGNMENT :  0
          INPUT-OUTPUT SECTION    :  1
          FILE-CONTROL            :  1
          SELECT                  :  2

NUMBER OF DECLARATIONS  :  85
          SECTIONS      :  2
          FD            :  0
          DECLARATIONS  :  61
          VALUE CLAUSES :  19
          REDEFINES     :  3
          RENAMES       :  0

NUMBER OF ASSIGNMENTS :  21
 *** note that the above total includes VALUES CLAUSES ***
          MOVE          :  1
          ADD           :  0
          SUBTRACT      :  0
          MULTIPLY      :  0
          DIVIDE        :  0
          COMPUTE       :  1

NUMBER OF CONDITIONALS  :  4
          IF            :  2
          ELSE          :  1
          ON            :  0
          ON SIZE ERROR :  0
          AT END        :  1

NUMBER OF BRANCHINGS :  4
```

```
          CALL        :  0
          PERFORM     :  0
          GO TO       :  3
          NEXT        :  0
          EXIT        :  0
          STOP        :  1

NUMBER OF INPUT/OUTPUT :  3
          DELETE      :  0
          DISPLAY     :  0
          OPEN        :  1
          CLOSE       :  1
          READ        :  1
          REWRITE     :  0
          WRITE       :  0
          ACCEPT      :  0

NUMBER OF LABELS :  2

NUMBER OF OTHER STATEMENTS :  1
          COPY              :  0
          ALTER             :  0
          TRANSFORM         :  0
          EXAMINE           :  1
          INSPECT           :  0
          SEARCH            :  0
          SORT              :  0
          SET               :  0
          CICS EXEC         :  0
          GOBACK            :  0

************************************************

TOTAL NUMBER OF ADDED SECTIONS IS: 1
TOTAL NUMBER OF ALTERS: 7
TOTAL NUMBER OF DELETIONS: 2

************************************************


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

ANALYSIS OF STATEMENTS ALTERED


NUMBER OF LINES OF COMMENTS :  5
    IDENTIFICATION DIVISION :  2
    SPACING PURPOSES        :  0
    USEFUL COMMENTS         :  3

NUMBER OF ENVIRONMENT STATEMENTS  :  2
          CONFIGURATION SECTION   :  0
          SOURCE-COMPUTER         :  0
          OBJECT-COMPUTER         :  0
          COMPUTER SPECIFICATION  :  0
          SPECIAL NAMES           :  0
          SPECIAL NAME ASSIGNMENT :  0
          INPUT-OUTPUT SECTION    :  0
          FILE-CONTROL            :  0
          SELECT                  :  2
```

```
NUMBER OF DECLARATIONS  :  0
          SECTIONS      :  0
          FD            :  0
          DECLARATIONS  :  0
          VALUE CLAUSES :  0
          REDEFINES     :  0
          RENAMES       :  0

NUMBER OF ASSIGNMENTS :  0
 *** note that the above total includes VALUES CLAUSES ***
          MOVE          :  0
          ADD           :  0
          SUBTRACT      :  0
          MULTIPLY      :  0
          DIVIDE        :  0
          COMPUTE       :  0

NUMBER OF CONDITIONALS  :  0
          IF            :  0
          ELSE          :  0
          ON            :  0
          ON SIZE ERROR :  0
          AT END        :  0

NUMBER OF BRANCHINGS :  0
          CALL       :  0
          PERFORM    :  0
          GO TO      :  0
          NEXT       :  0
          EXIT       :  0
          STOP       :  0

NUMBER OF INPUT/OUTPUT :  0
          DELETE       :  0
          DISPLAY      :  0
          OPEN         :  0
          CLOSE        :  0
          READ         :  0
          REWRITE      :  0
          WRITE        :  0
          ACCEPT       :  0

NUMBER OF LABELS :  0

NUMBER OF OTHER STATEMENTS :  0
          COPY             :  0
          ALTER            :  0
          TRANSFORM        :  0
          EXAMINE          :  0
          INSPECT          :  0
          SEARCH           :  0
          SORT             :  0
          SET              :  0
          CICS EXEC        :  0
          GOBACK           :  0
```

ANALYSIS OF STATEMENTS DELETED

```
NUMBER OF LINES OF COMMENTS :  2
    IDENTIFICATION DIVISION :  2
    SPACING PURPOSES        :  0
    USEFUL COMMENTS         :  0

NUMBER OF ENVIRONMENT STATEMENTS  :  0
          CONFIGURATION SECTION   :  0
          SOURCE-COMPUTER         :  0
          OBJECT-COMPUTER         :  0
          COMPUTER SPECIFICATION  :  0
          SPECIAL NAMES           :  0
          SPECIAL NAME ASSIGNMENT :  0
          INPUT-OUTPUT SECTION    :  0
          FILE-CONTROL            :  0
          SELECT                  :  0

NUMBER OF DECLARATIONS  :  0
          SECTIONS      :  0
          FD            :  0
          DECLARATIONS  :  0
          VALUE CLAUSES :  0
          REDEFINES     :  0
          RENAMES       :  0

NUMBER OF ASSIGNMENTS :  0
 *** note that the above total includes VALUES CLAUSES ***
          MOVE          :  0
          ADD           :  0
          SUBTRACT      :  0
          MULTIPLY      :  0
          DIVIDE        :  0
          COMPUTE       :  0

NUMBER OF CONDITIONALS  :  0
          IF            :  0
          ELSE          :  0
          ON            :  0
          ON SIZE ERROR :  0
          AT END        :  0

NUMBER OF BRANCHINGS :  0
          CALL       :  0
          PERFORM    :  0
          GO TO      :  0
          NEXT       :  0
          EXIT       :  0
          STOP       :  0

NUMBER OF INPUT/OUTPUT :  0
          DELETE       :  0
          DISPLAY      :  0
          OPEN         :  0
          CLOSE        :  0
          READ         :  0
          REWRITE      :  0
          WRITE        :  0
          ACCEPT       :  0

NUMBER OF LABELS :  0

NUMBER OF OTHER STATEMENTS :  0
          COPY             :  0
```

```
          ALTER              :  0
          TRANSFORM          :  0
          EXAMINE            :  0
          INSPECT            :  0
          SEARCH             :  0
          SORT               :  0
          SET                :  0
          CICS EXEC          :  0
          GOBACK             :  0


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

ANALYSIS OF STATEMENTS ADDED


TOTAL NUMBER OF ADDED STATEMENTS: 3


NUMBER OF LINES OF COMMENTS :  3
    IDENTIFICATION DIVISION :  0
    SPACING PURPOSES        :  3
    USEFUL COMMENTS         :  0

NUMBER OF ENVIRONMENT STATEMENTS  :  0
          CONFIGURATION SECTION   :  0
          SOURCE-COMPUTER         :  0
          OBJECT-COMPUTER         :  0
          COMPUTER SPECIFICATION  :  0
          SPECIAL NAMES           :  0
          SPECIAL NAME ASSIGNMENT :  0
          INPUT-OUTPUT SECTION    :  0
          FILE-CONTROL            :  0
          SELECT                  :  0

NUMBER OF DECLARATIONS  :  0
          SECTIONS      :  0
          FD            :  0
          DECLARATIONS  :  0
          VALUE CLAUSES :  0
          REDEFINES     :  0
          RENAMES       :  0

NUMBER OF ASSIGNMENTS :  0
 *** note that the above total includes VALUES CLAUSES ***
          MOVE        :  0
          ADD         :  0
          SUBTRACT    :  0
          MULTIPLY    :  0
          DIVIDE      :  0
          COMPUTE     :  0

NUMBER OF CONDITIONALS  :  0
          IF            :  0
          ELSE          :  0
          ON            :  0
          ON SIZE ERROR :  0
          AT END        :  0

NUMBER OF BRANCHINGS :  0
          CALL       :  0
```

```
          PERFORM     :  0
          GO TO       :  0
          NEXT        :  0
          EXIT        :  0
          STOP        :  0

NUMBER OF INPUT/OUTPUT :  0
          DELETE      :  0
          DISPLAY     :  0
          OPEN        :  0
          CLOSE       :  0
          READ        :  0
          REWRITE     :  0
          WRITE       :  0
          ACCEPT      :  0

NUMBER OF LABELS :  0

NUMBER OF OTHER STATEMENTS :  0
          COPY        :  0
          ALTER       :  0
          TRANSFORM   :  0
          EXAMINE     :  0
          INSPECT     :  0
          SEARCH      :  0
          SORT        :  0
          SET         :  0
          CICS EXEC   :  0
          GOBACK      :  0

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

ANALYSIS OF ALTERED STATEMENTS


ORIGINAL LINE  < Comment                VERSION 1.

ALTERED TO

NEWLINE  > Comment                VERSION 2.

ORIGINAL LINE  <            SELECT PRINT-FILE ASSIGN SYS003-UR-1403-S.      Env
ORIGINAL LINE  <            SELECT CARD-FILE ASSIGN SYS004-UR-2520R-S.      Env

ALTERED TO

NEWLINE  >          SELECT PRINT-FILE ASSIGN UT-P1L.                        Env
NEWLINE  >          SELECT CARD-FILE ASSIGN UT-CARDIN.                      Env


ORIGINAL LINE  <            * IF C-AMNTX NOT NUMERIC
ORIGINAL LINE  <                 *   GENERATE ERROR-DETAIL
ORIGINAL LINE  <                 *   MOVE ZERO TO C-AMNTX.

ALTERED TO

NEWLINE  > IF C-AMNTX NOT NUMERIC GENERATE ERROR-DETAIL
NEWLINE  >      MOVE ZERO TO C-AMNTX.
------------------------------------------------------------

LIST THE DELETED STATEMENTS

  Comment               *  PROGRAOM WAS WRITTEN FROM  PRG JEX 10-8-84 MVS
```

```
Comment                 * CONVERSION, MODIFIED SELECT CLAUSE
-----------------------------------------------------------

LIST THE ADDED STATEMENTS

                        *
                        *
                        *
```

Appendix E. The Shell Program Classify

```
#
#     The input file is the result from executing the shell
#     program Maintain.   It generates 4 temporary files.
#     "Alter1" stores old altered statements and "alter2" stores
#     new altered statements.  Deleted statemens are put into "delfile"
#     and added statements in "addfile".
#
#     The output lists six types of maintenance from 3 files.  The six types
#     are Correction, Adaption, Retrenchment, Retrieving, Pretty printing,
#     and Documentation.   The puls and minus signs in pretty printing and
#     documentation stand for the increasing or decreasing or the numbers.
#
echo  ANALYZING FOR : $1 > $1.out
echo  ------------------------------------ > out
#
#  insert special characters to original file
#
awk '
BEGIN { line0 = 0 }
  /LIST THE ADDED STATEMENTS/       { print "#%"$" >> "copyfile" }
  /LIST THE DELETED STATEMENTS/     { print "#%"$" >> "copyfile" }
                                    { print $0     >> "copyfile" }
END { print "#%"$" >> "copyfile" } ' $1

#  line1: old numbers of altered statement;     line2: new numbers of altered
#  noline1: old numbers of altered statements in a block
#  noblock: numbers of block being altered
#  noalter: numbers of block counted as big changed in size
awk '
BEGIN  { line1=0; line2=0; noline1=0; noline2=0; flag =0; noblock=0; noalter=0}
  /ANALYSIS OF ALTERED STATEMENTS/, /\#\%\"\$/  {
       if ($3 == "<")
            line++
            #  store old altered statements to alter1 file
            i = 4
            while (i <= NF)
              { print $i >> "alter1"; i++ }
            #
            if (flag == 0)
              { noline1++ }
            # compute the block which changes rapidly in size
            if (flag == 1)
              { if ((noline1 > 10) && (noline2 > 10))
                   { if (noline1 > noline2)
                        { div1 = noline1 / noline2 }
                     else
                        { div1 = noline2 / noline1 }
                     if ( div1 > 2)
                        { noalter++ }
                   }
                else
                   { if (noline1 > noline2)
                        { div2 = noline1 / noline2
                          dif2 = noline1 - noline2
                        }
                     else
                        { div2 = noline2 / noline1
                          dif2 = noline2 - noline1
                        }
                     if ((div2 > 5) || (dif2 > 5))
                        { noalter++ }
```

```
                # reset to 0 after done a block
                flag = 0
                noline1 = 1
                noline2 = 0
              }
          }
        if ($2 == ">")
          { line2++
            # store new altered statements to alter2 file
            j = 3
            while  (j <= NF)
               { print $j >>  "alter2"; j++ }
            if (flag == 0)
               { noline2 = 0; flag = 1 }
            if (flag == 1)
               { noline2++ }
          }
        if (($1 == "ALTERED") && ($2 == "TO") && (NF == 2))
          { noblock++ }
    }
 #  store deleted statements to delfile
 /LIST THE DELETED STATEMENTS/, /\#\$\*\$/ {
       if (($3=="DELETED")||($0=="#$*$")||(NF==0)||(substr($0,1,3)=="---"))
           {  }
       else
           { print $0 >> "delfile" }
     }
# store added statements to addfile
 /LIST THE ADDED STATEMENTS/, /\#\$\*\$/ {
       if (($3=="ADDED")||($0=="#$*$")||(NF==0)||(substr($0,1,3)=="---"))
           {  }
       else
           { print $0 >> "addfile" }
     }
END { if (noblock != 0)
         {  print "\n (( Altered ))"    >> "out"
            print "          number of original line : " line1 >> "out"
            print "          number of new line      : " line2  >> "out"
            print "          number of block altered : " noblock >> "out"
            if  (noalter > 5)
                { print "       <Adaptive>" >> "out"}
            else
                { print "       <Corrective>" >> "out" }
         } } '  copyfile
#
#
if  (( test -f alter1) && (test -f alter2))
then

    diff alter1 alter2 > difference
    sed  's/\,/ /g
          s/a/ a /g
          s/c/ c /g
          s/d/ d /g'  difference > result
    awk '
    BEGIN { NoAdd = 0; NoDel = 0; NoRetrench = 0; NoRetrieve = 0;
            alterfrom = 0; alterto = 0;
            DelDocument = 0; AddDocument = 0; DelPrint = 0;  AddPrint = 0
            aflag = 0; cflag = 0; dflag = 0; c1flag =0; c2flag =0  }
```

```
    { if ($2 == "a")
        { aflag = 1; cflag = 0; dflag = 0 }
      if (($2 == "c") || ($3 == "c"))
        { cflag = 1; c1flag = 0; c2flag = 0; c3flag = 0; aflag = 0; dflag = 0}
      if (($2 == "d") || ($3 == "d"))
        { dflag = 1; aflag = 0; cflag = 0 }

      if ( aflag == 1 )
        { if ($1 == ">")
            { if ($2 == "*")
                { NoRetrench++ }
              else
                { NoAdd++ } } }
      if ( cflag == 1 )
        { if ($0 == "< *")
            { if (c1flag == 1)
                { DelPrint++ }
              if (c1flag == 0 )
                { c1flag = 1 }
              prestar = 1 }
          if (($1 == "<") && ($2 != "*") && (c1flag == 1))
            { c1flag = 0; DelDocument++ ; prestar = 0  }
          if (($0 == "---") && (c1flag == 1) && (prestar == 1))
            { NoRetrieve++ }
          if (($0 == "---") && (c1flag == 1) && (prestar == 0))
            { DelPrint++ }
          if ($0 == "> *")
            { if (c2flag == 1)
                { AddPrint++
                  if (c3flag == 0)
                    { AddPrint++; c3flag = 1 }  }
              if (c2flag == 0)
                { c2flag = 1 }  }
          if (($1 == ">") && ($2 != "*") && (c2flag == 1))
            { c2flag = 0;  AddDocument++ }   }
          if (($1 == "<") && ($2 != "*") && (c1flag == 0))
            { alterfrom++ }
          if (($1 == ">") && ($2 != "*") && (c2flag == 0))
            { alterto++ }
      if ( dflag == 1 )
        { if ($1 == "<")
            { if ($2 == "*")
                { NoRetrieve++ }
              else
                { NoDel++ }  }  }
    }
  END { if ( NoRetrench > 0 )
          { print "       <Retrenchment> : number = " NoRetrench >> "out" }
        if ( NoRetrieve > 0 )
          { print "       <Retrieving>   : number = " NoRetrieve >> "out" }
        if (AddDocument > 0)
          { print "       <Documentation+> : number = " AddDocument >> "out"}
        if (DelDocument > 0)
          { print "       <Documentation-> : number = " DelDocument >> "out"}
        if (AddPrint > 0)
          { print "       <Pretty Printing+> : number = " AddPrint >> "out"}
        if (DelPrint > 0)
          { print "       <Pretty Printing-> : number = " DelPrint >> "out"}
      } ' result
  rm alter1  alter2  result  difference
fi
```

```
#
#  if there exist deleted statements
if  ( test -f delfile )
then
    awk '
    BEGIN { NoComment = 0; NoDlrtn = 0; DelDoc = 0; DelDoc = 0}
      { if ($1 == "Comment")
           [ NoComment++ ]
        if ($1 == "Dlrtn")
           [ NoDlrtn++ ]
        if (($0 == "*") && (NF == 1))
           [ DelPnt++ ]
        if (($1 == "*") && (NF > 1))
           [ DelDoc++ ]
      }
    END { { print "\n (( Deleted )) "  >> "out" }
          {print "       The total number of deleted statemens is " NR >>"out"}
        if (NoComment > 0)
           { print "          Comment is deleted.  number: " NoComment >> "out"}
        if (NoDlrtn > 0)
           { print "          Declaration is deleted.  number: " NoDlrtn >> "out"}
        if (DelPnt > 0)
           { print "       <Pretty printing-> : number " DelPnt >> "out"}
        if (DelDoc > 0)
           { print "       <Documentation-> : number " DelDoc >> "out"}
        NoOther = NR - Nocomment - NoDlrtn - DelPnt - DelDoc
        if (NoOther > 10)
           { print "       <Adaptive> : number " NoOther >> "out"}
        if ((NoOther <= 10) && (NoOther > 0))
           { print "       <Corrective> : number " NoOther>> "out"}
      } ' delfile
     rm delfile
fi
#
#  if there  exists added statements
if  ( test -f addfile )
then
   awk '
   BEGIN { NoComment = 0; NoDlrtn = 0; AddDoc = 0; AddPnt = 0 }
     { if ($1 == "Comment")
          { NoComment++ }
       if ($1 == "Dlrtn")
          { NoDlrtn++ }
       if (($1 == "*") && (NF == 1))
          { AddPnt++ }
       if (($1 == "*") && (NF > 1))
          { AddDoc++  }
     }
   END {{ print "\n (( Added )) " >> "out" }
        { print "       The total number of added statements is " NR >> "out" }
        if (NoComment > 0)
           { print "          Comment is added.  number: " NoComment >> "out" }
        if (NoDlrtn > 0)
           { print "          Declaration is added.  number: " NoDlrtn >> "out"}
        if (AddPnt > 0)
           { print "       <Pretty Printing+> :  number " AddPnt >> "out"}
        if (AddDoc > 0)
           { print "       <Documentation+> :  number " AddDoc >> "out"}
        NoOther = NR - NoComment - NoDlrtn - AddPnt - AddDoc
        if ( NoOther > 10)
           { print "       <Adaptive> : number " NoOther >> "out"}
        if (( NoOther <= 10) && ( NoOther > 0))
           { print "       <Corrective> : number " NoOther >> "out"}
        } ' addfile
fi
```

Appendix F. Result from Running Classify

```
ANALYZING FOR : COBOL.1.listing
---------------------------------------

(( Altered ))
        number of original line : 7
        number of new line      : 6
        number of block altered : 3
     <Corrective>
     <Retrieving>   : number = 3

(( Deleted ))
     The total number of deleted statemens is 2
        Comment is deleted.  number: 2
     <Corrective> : number 2

(( Added ))
     The total number of added statements is 3
     <Pretty Printing+> :  number 3
```

ANALYZING CHANGES IN COBOL PROGRAMS DURING MAINTENANCE

by

IE-HONG LIN

B.S., NATIONAL CHANG KUNG UNIVERSITY
Tainan, Taiwan 1980

M.S., KANSAS STATE UNIVERSITY
Manhattan, Kansas 1985

AN ABSTRACT OF A THESIS

submitted in partial fulfillment of the

requirement for the degree

MASTER OF SCIENCE

Department of Computing and Information Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1988

Abstract

Software maintenance has become the most expensive phase. To maintain software, managers need methods to monitor the process in order to predict where changes will occur. Knowing the types of maintenance help managers in managing the maintenance.

The study presents a method to classify types of maintenance. The work focuses on analyzing COBOL programs and classifying different types of maintenance. The shell program Maintain was written as a tool to analyze two sequential versions on a program. Program set A, from a Kansas company, was first introduced to analyze. Six types of maintenance were identified from the results. They are corrective, adaptive, retrenchment, retrieving, pretty printing, and documentation. The classification rules were then converted into the second shell program Classify. Program set B, from data processing environment, was finally verified with the program Maintain and Classify to test the results.

The presented method is successfully in classifying types of maintenance from empirical data that changes between two versions of a program. In particular, the method allows managers to identify types of maintenance that have been done and evaluate the effort by means of the classification rules.