USC COCOMOII.1997

Reference Manual

University of Southern California

This manual is compatible with USC-COCOMOII.1997 version 0.

Printed in the United States of America.

Acknowledgments

Cocomo81,Version 1.0:
Principal Investigator - Dr. Ellis Horowitz
Student Designers, Testers and Programmers - Alfredo Arcilla, Joyce Balimbin, Gina Gaborno, Larry Klein, Robert Kosai, Deseree Moon, Jason Pan, Thomas Quayle, Isaiah Simmons, Scott Zechiel

Cocomo81, Version 1.1:
Principal Investigator - Dr. Ellis Horowitz
Student Designers, Testers and Programmers - Ing-Jer Huang

Cocomo81, Version 10.0:
Principal Investigator - Dr. Ellis Horowitz
Student Designers, Testers and Programmers - M. Susan Lane, Ping Luo, Lorna Zorman

Cocomo2.0, Version 2.0:
Principal Investigator - Dr. Ellis Horowitz
Student Designers, Testers and Programmers - Wiryadi Adidharma, Sen-Ho Chang, Shu-fen Cheng, Yu-Chuan Lin, Steve K. Luk, Shawne Robinson, Tuan Ton

Cocomo2.0 Version 2.0.5:
Principal Investigator - Dr. Ellis Horowitz
Student Designers, Testers and Programmers -
Thomas Majchrowski, Suppachai Sangtongkhamsuk, Lloyd Manglapus

CocomoII.1997.1
Principal Investigator - Dr. Ellis Horowitz
Student programmers: Jongmoon Baik, Jungwon Park, James Chou, Dana Flora-Adams, Sang Hyun, and Eunsook Bang

Note - This manual is sufficient for the X Windows, MS Windows 95, and Java versions of USC-COCOMOII.1997 programs. Though there are some user interface differences between these systems, the core component of all versions is identical.

Some of the material used in this manual has been taken from Software Engineering Economics, by Barry Boehm, Prentice-Hall, with permission of the author.

Chapter

1 Introduction

1.1 What is COCOMO?

COCOMO (COnstructive COst MOdel) is a screen-oriented, interactive-software package that assists in budgetary planning and schedule estimation of a software development project. Through the flexibility of COCOMO, a software project manager (or team leader) can develop a model (or multiple models) of projects in order to identify potential problems in resources, personnel, budgets, and schedules both before and while the potential software package is being developed.

The COCOMO software package is based upon the software cost and schedule estimation model: COnstructive COst MOdel version II (COCOMOII) which is the newly revised version of its original COnstructive COst MOdel (COCOMO) first published by Dr. Barry Boehm in his book Software Engineering Economics, Prentice-Hall (1981), and Ada COCOMO (1989) predecessors. The current model is described in [Boehm et al. 1995]

The primary objectives of the COCOMOII.1997 effort are:

- To develop a software cost and schedule estimation model tuned to the life cycle practices of the 1990's and 2000's.
- To develop software cost database and tool support capabilities for continuous model improvement.
- To provide a quantitative analytic framework, and set of tools and techniques for evaluating the effects of software technology improvements on software life cycle costs and schedules.

The full COCOMOII model includes three stages. Stage 1 supports estimation of prototyping or applications composition efforts. Stage 2 supports estimation in the Early Design stage of a project, when less is known about the project's cost drivers. Stage 3 supports estimation in the Post-Architecture stage of a project.

This version of USC COCOMOII implements stage 3 formulas to estimate the effort, schedule, and cost required to develop a software product. It also provides the breakdown of effort and schedule into software life-cycle phases and activities from the original COCOMO manual. These are still reasonably valid for waterfall model software projects, but need to be interpreted for non-waterfall projects.

1.1.1 Effort Estimation Equation

Estimate effort with:

$$PM = \prod_{i=1}^{17} (EM_i) \cdot A \cdot \left[\left(1 + \frac{BRAK}{100}\right) \cdot Size\right]^{\left(1.01 + 0.01 \sum_{j=1}^{5} SF_j\right)} + \left(\frac{ASLOC \cdot \left(\frac{AT}{100}\right)}{ATPROD}\right) \quad \textbf{EQ 1.}$$

where

$$Size = KNSLOC + \left[KASLOC \cdot \left(\frac{100 - AT}{100}\right) \cdot \frac{(AA + SU + 0.4 \cdot DM + 0.3 \cdot CM + 0.3 \cdot IM)}{100}\right]$$

$$B = 1.01 + 0.01 \sum_{j=1}^{5} SF$$

Symbol	Description
A	Constant, currently calibrated as 2.45
AA	Assessment and assimilation
ADAPT	Percentage of components adapted (represents the effort required in understanding software)
AT	Percentage of components that are automatically translated
ATPROD	Automatic translation productivity
BRAK	Breakage: Percentage of code thrown away due to requirements volatility
CM	Percentage of code modified
DM	Percentage of design modified
EM	Effort Multipliers: RELY, DATA, CPLX, RUSE, DOCU, TIME, STOR, PVOL, ACAP, PCAP, PCON, AEXP, PEXP, LTEX, TOOL, SITE
IM	Percentage of integration and test modified
KASLOC	Size of the adapted component expressed in thousands of adapted source lines of code
KNSLOC	Size of component expressed in thousands of new source lines of code
PM	Person Months of estimated effort
SF	Scale Factors: PREC, FLEX, RESL, TEAM, PMAT
SU	Software understanding (zero if DM = 0 and CM = 0)

1.1.2 Schedule Estimation Equation

Determine time to develop (TDEV) with an estimated effort, $\overline{PM}$, that excludes the effect of the SCED effort multiplier:

$$TDEV = \left[2.66 \times (\overline{PM})^{(0.33 + 0.2 \times (B - 1.01))} \right] \cdot \frac{\text{SCED\%}}{100}$$

where

$$B = 1.01 + 0.01 \sum_{j=1}^{5} SF_j$$

EQ 2.

Symbol	Description
$\overline{PM}$	Person Months of estimated effort from Early Design or Post-Architecture models (excluding the effect of the SCED effort multiplier).
SF	Scale Factors: PREC, FLEX, RESL, TEAM, PMAT
TDEV	Time to develop
SCED	Schedule
SCED%	The compression / expansion percentage in the SCED effort multiplier

1.1.3 Scale Factors

Equation 2 defines the exponent, *B*, used in Equation 1. Table 1.1 provides the rating levels for the COCOMOII scale drivers. The selection of scale drivers is based on the rationale that they are a significant source of exponential variation on a project's effort or productivity variation. Each scale driver has a range of rating levels, from Very Low to Extra High. Each rating level has a weight, *W*, and the specific value of the weight is called a scale factor. A project's scale factors, W_i, are summed across all of the factors, and used to determine a scale exponent, *B*, via the following formula:

$$B = 1.01 + 0.01 \times \sum_{j=1}^{5} SF_j \qquad \textit{EQ 3.}$$

For example, if scale factors with an Extra High rating are each assigned a weight of (0), then a 100 KSLOC project with Extra High ratings for all factors will have $SF_j = 0$, $B = 1.01$, and a relative effort $E = 100^{1.01} = 105$ *PM*. If scale factors with Very Low rating are each assigned a weight of (5), then a project with Very Low (5) ratings for all factors will have $SF_j = 5$, $B = 1.26$, and a relative effort $E = 331$ *PM*. This represents a large variation, but the increase involved in a one-unit change in one of the factors is only about 4.7%.

Scale Factors (SF_j)	Very Low	Low	Nominal	High	Very High	Extra High
PREC	thoroughly unprecedented	largely unprecedented	somewhat unprecedented	generally familiar	largely familiar	throughly familiar
FLEX	rigorous	occasional relaxation	some relaxation	general conformity	some conformity	general goals
RESL[a]	little (20%)	some (40%)	often (60%)	generally (75%)	mostly (90%)	full (100%)
TEAM	very difficult interactions	some difficult interactions	basically cooperative interactions	largely cooperative	highly cooperative	seamless interactions
PMAT	Weighted average of "Yes" answers to CMM Maturity Questionnaire					

Table 1.1: Scale Factors for COCOMO.II Early Design and Post-Architecture Models

a. % significant module interfaces specified,% significant risks eliminated.

1.1.4 Sizing Methods

SLOC: Lines of Code Counting Rules

In COCOMOII, the logical source statement has been chosen as the standard line of code. Defining a line of code is difficult due to conceptual differences involved in accounting for executable statements and data declarations in different languages. The goal is to measure the amount of intellectual work put into program development, but difficulties arise when trying to define consistent measures across different languages. Breakage due to requirements change also complicates sizing. To minimize these problems, the Software Engineering Institute (SEI) definition checklist for a logical source statement is used in defining the line of code measure. The Software Engineering Institute (SEI) has developed this checklist as part of a system of definition checklists, report forms and supplemental forms to support measurement definitions [Park 1992] [Goethert et al. 1992].

Figure 1-1 shows a portion of the definition checklist as it is being applied to support the development of the COCOMOII model. Each checkmark in the “Includes” column identifies a particular statement type or attribute included in the definition, and vice-versa for the excludes. Other sections in the definition clarify statement attributes for usage, delivery, functionality, replications and development status. There are also clarifications for language specific statements for ADA, C, C++, CMS-2, COBOL, FORTRAN, JOVIAL and Pascal. The complete checklist is in Appendix B.

Some changes were made to the line-of-code definition that depart from the default definition provided in [Park 1992]. These changes eliminate categories of software which are generally small sources of project effort. Not included in the definition are commercial-off-the-shelf software (COTS), government furnished software (GFS), other products, language support libraries and operating systems, or other commercial libraries. Code generated with source code generators is not included though measurements will be taken with and without generated code to support analysis.

The “COCOMOII line-of-code definition” can be calculated directly by the Amadeus automated metrics collection tool [Amadeus 1994] [Selby et al. 1991], which can be used to ensure uniformly collected data in the COCOMOII data collection and analysis project. We have developed a set of Amadeus measurement templates that support the COCOMOII data definitions for use by the organizations collecting data, in order to facilitate standard definitions and consistent data across participating sites.

To support further data analysis, Amadeus will automatically collect additional measures including total source lines, comments, executable statements, declarations, structure, component interfaces, nesting, and others. The tool will provide various size measures, including some of the object sizing metrics in [Chidamber and Kemerer 1994], and the COCOMO sizing formulation will adapt as further data is collected and analyzed.

Definition Checklist for Source Statements Counts

Definition name: Logical Source Statements Date:_____

(basic definition) Originator: COCOMO.II

Measurement unit:	Physical source lines						
	Logical source statements	✔					
Statement type	Definition	✔	Data Array			Includes	Excludes
When a line or statement contains more than one type, classify it as the type with the highest precedence.							
1 Executable	Order of precedence →				1	✔	
2 Nonexecutable							
3 Declarations					2	✔	
4 Compiler directives					3	✔	
5 Comments							
6 On their own lines					4		✔
7 On lines with source code					5		✔
8 Banners and non-blank spacers					6		✔
9 Blank (empty) comments					7		✔
10 Blank lines					8		✔
11							
12							
How produced	Definition	✔	Data array			Includes	Excludes
1 Programmed						✔	
2 Generated with source code generators							✔
3 Converted with automated translators						✔	
4 Copied or reused without change						✔	
5 Modified						✔	
6 Removed							✔
7							
8							
Origin	Definition	✔	Data array			Includes	Excludes
1 New work: no prior existence						✔	
2 Prior work: taken or adapted from							
3 A previous version, build, or release						✔	
4 Commercial, off-the-shelf software (COTS), other than libraries							✔
5 Government furnished software (GFS), other than reuse libraries							✔
6 Another product							✔
7 A vendor-supplied language support library (unmodified)							✔
8 A vendor-supplied operating system or utility (unmodified)							✔
9 A local or modified language support library or operating system							✔
10 Other commercial library							✔
11 A reuse library (software designed for reuse)						✔	
12 Other software component or library						✔	
13							
14							

Figure 1. Definition Checklist

FP: Counting with Unadjusted Function Points

The function point cost estimation approach is based on the amount of functionality in a software project and a set of individual project factors [Behrens 1983][Kunkler 1985][IFPUG 1994]. Function points are useful estimators since they are based on information that is available early in the project life cycle. A brief summary of function points and their calculation of COCOMOII is as follows.

Function points measure a software project by quantifying the information processing functionality associated with major external data input, output, or file types. Five user function types should be identified as defined in the Table2.

External Input (Inputs)	Count each unique user data or user control input type that (i) enters the external boundary of the software system being measured and (ii) adds or changes data in a logical internal file.
External Output (Outputs)	Count each unique user data or control output type that leaves the external boundary of the software system being measured.
Internal Logical File (Files)	Count each major logical group of user data or control information in the software system as a logical internal file type. Include each logical file (e.g., each logical group of data) that is generated, used, or maintained by the software system.
External Interface Files (Interfaces)	Files passed or shared between software systems should be counted as external interface file types within each system.
External Inquiry (Queries)	Count each unique input-output combination, where an input causes and generates an immediate output, as an external inquiry type.

Table 2: User Function Types

Each instance of these function types is then classified by complexity level. The complexity levels determine a set of weights, which are applied to their corresponding function counts to determine the Unadjusted Function Points quantity. This is the Function Point sizing metric used by COCOMII. The usual Function Point procedure involves assessing the degree of influence (DI) of fourteen application characteristics on the software project determined according to a rating scale of 0.0 to 0.05 for each characteristic. The 14 ratings are added together, and added to a base level of 0.65 to produce a general characteristics adjustment factor that ranges from 0.65 to 1.35.

Each of these fourteen characteristics, such as distributed functions, performance, and reusability, thus have a maximum of 5% contribution to estimated effort. This is inconsistent with COCOMO experience; thus COCOMO.II uses Unadjusted Function Points for sizing, and applies its reuse factors, cost driver effort multipliers, and exponent scale factors to this sizing quantity.

AAF: Adaptation Adjustment Factors

Adaptation of Existing Code

COCOMO is not only capable of estimating the cost and schedule for a development started from "scratch", but it is also able to estimate the cost and schedule for products that are built upon already existing code. Adaptation considerations have also been incorporated into COCOMO, where an estimate for KSLOC will be calculated. This value will be substituted in place of the SLOC found in the equations already discussed. This adaptation of code utilizes an additional set of equations that are used to calculate the final count on source instructions and related cost and schedule. These equations use the following values as components:

- Adapted Source Lines of Code (ASLOC). The number of source lines of code adapted from existing software used in developing the new product.
- Percent of Design Modification (DM). The percentage of the adapted software's design that received modification to fulfill the objectives and environment of the new product.
- Percent of Code Modification (CM). The percentage of the adapted software's code that receives modification to fulfill the objectives and environment of the new product.
- Percent of Integration Required for Modified Software (IM). The percentage of effort needed for integrating and testing of the adapted software in order to combine it into the new product.
- Percentage of reuse effort due to Software Understanding (SU).
- Percentage of reuse effort due to Assessment and Assimilation (AA).
- Programmer Unfamiliarity with Software (UNFM)

These components are brought together in Figure 1-6. The AAF is the adaptation adjustment factor. The AAF is the calculated degree to which the adapted software will affect overall development.

1.1.5 Effort Multipliers

There are a number of contributing factors to a project‘s delivery time and effort. Development productivity was found to be affected by additional factors that were found to fall under the headings: product attributes, platform attributes, personnel attributes, and project attributes.

Product attributes refer to the constraints and requirements placed upon the project to be developed. These included

- Required software reliability (RELY)
- Database size (DATA)
- Documentation match to life-cycle needs (DOCU)
- Product complexity (CPLX)

- Required Reusability (RUSE)

Platform attributes refer to the limitations placed upon development effort by the hardware and operating system being used to run the project. These limitations are listed below.

- Execution time constraint (TIME)
- Main storage constraint (STOR)
- Platform volatility (PVOL)

Personnel attributes refer to the level of skills that are possessed by the personnel. The skills in question are general professional ability, programming ability, experience with the development environment and familiarity with the project's domain. These skills are characterized below.

- Analyst capabilities (ACAP)
- Applications experience (AEXP)
- Programmer capabilities (PCAP)
- Platform experience (PEXP)
- Programming language experience (LEXP)
- Personnel Continuity (PCON)

Project attributes refer to the constraints and conditions under which project development takes place. The issues that affect development are:

- Use of software tools (TOOL)
- Multisite Development (SITE)

These 16 factors are incorporated into calculating an estimated effort and schedule. Each of the factors has associated with it up to six ratings. These ratings are *very low, low, nominal, high, very high,* and *extra high*. Each rating has a corresponding real number based upon the factor and the degree to which the factor can influence productivity. A rating less than 1 denotes a factor that can decrease the schedule and effort. A rating greater than 1 denotes a factor that extends the schedule or effort. Lastly, a rating equal to 1 does not extend nor decrease the schedule and effort (this rating is called *nominal*).

These 16 factors (or effort multipliers) are incorporated into the schedule and effort estimation formulas by multiplying them together. The numerical value of the i^{th} adjustment factor (there are 16 of them) is called EM_i and their product is called the adjustment factor or EAF. The actual effort, PM_{total} is the product of the nominal effort times the EAF (see Figure 1-7).

In addition to the 16 EAF factors there are two user defined factors named USR1 and USR2. Their initial values are all set to 1. They may be redefined by using the Parameters-User Defined EAF menu item.

A final effort multiplier, Required Development Schedule (SCED) is treated separately as it operates at the overall project level rather than potentially varying from module to module.

FIGURE 1-2 **Estimate Development Effort**

$$PMtotal = (SCED) \times PMnominal \times \prod_{i=1}^{18} EMi$$

1.2 Navigating COCOMO

This software is a stand-alone software system intended for a single user. The software is extremely user interactive in that it attempts to interface well with a user's needs, using extensive mouse interaction wherever possible.

On the screen in Figure 1-3 is the CLEF (Component Level Estimation Form). This is where all of the entered information will be displayed. The top of the screen shows all of the subfunctions which the user may call. The choices appear in pop down menus according to the major headings of Project, Model, and Phase.

In order to efficiently use COCOMO, you must become familiar with the Component Level Estimating Form (CLEF). The different sections that are to be discussed have been given a corresponding number. These sections are given a descriptive label as a point of reference as well as a summary of their contents and functions

The sections found in Figure 1-3 and their descriptions are as follows:

1. **Main Menu bar** - This area contains the menu selection of the main functions of COCOMO. These selections are *File, Edit, Parameters, Calibrate, Phase Distribution* and *Help*. File, Edit, Parameters, Calibrate, and Phase Distribution are discussed in chapters 2, 3, 4, 5, and 6 respectively. Help is the selection used to receive on-line assistance with the available functions.

2. **Project Reference Names** - These fields denoted by this arrow contain the following elements: *Project File and Model File*.

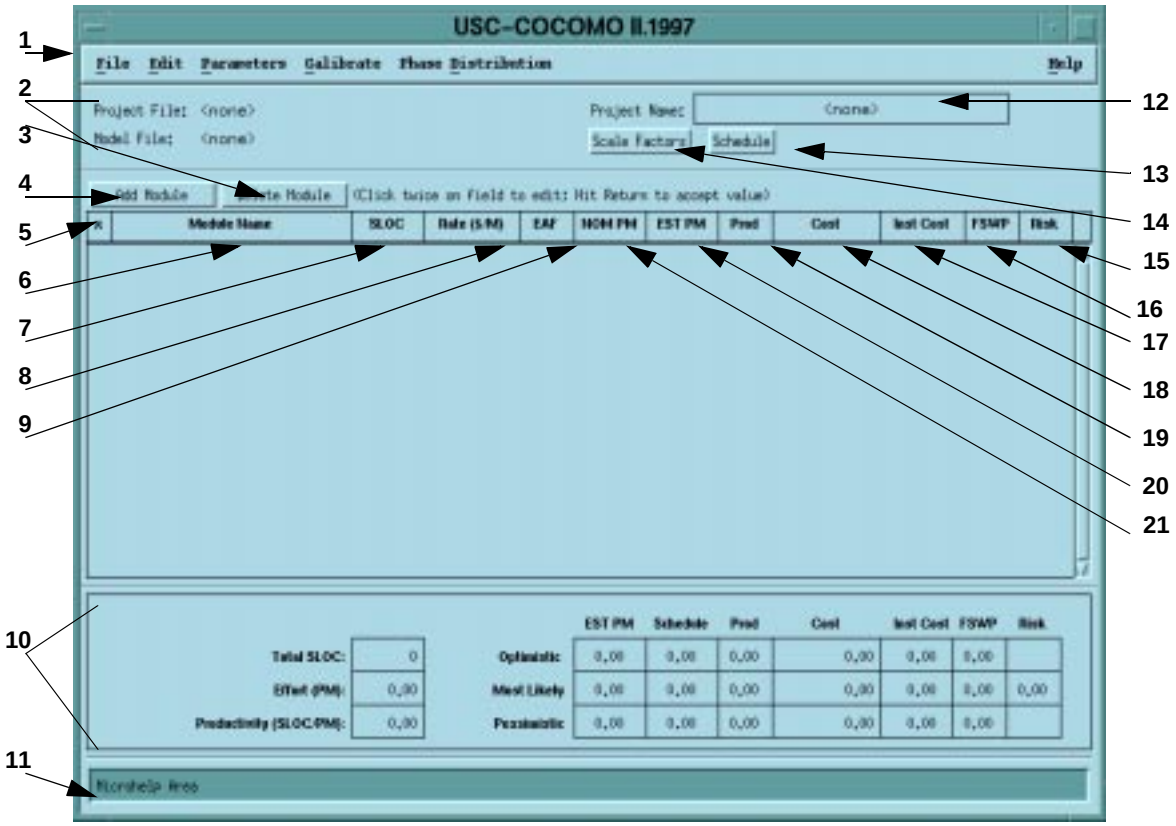
- Project File displays the name of the current project file loaded into COCOMO. The default name is "(none)" as found in Figure 1-3.

- Model File displays the name of the current model file that has been loaded into COCOMO. The default name is "(none)" as found in Figure 1-3.

3. **Delete Module** - This button is used to delete a module that has been marked with an "x" (see number 5).

4. **Add Module** - This button when depressed adds a new module to the project that is currently being worked upon by the user.

FIGURE 1-3 **COCOMO CLEF**



5. **x** - This column is reserved for identifying a module. Pressing upon this field for a given module will mark the desired module. Marking is denoted by an x that appears in this column. Only one module can be marked at a time. Modules are marked in order to perform module deletion, cutting, copying or pasting.

6. **Module Name Column** - This column is used to house the name of each module located in the Module Area. The module name can be changed by clicking twice on the desired module name box and entering the changes into the module name field. Upon completion of editing press "Return".

7. **Source Lines of Code (SLOC) Column** - This column is used to house the SLOC of each module located in the Module Area. The value for SLOC can be computed in one of three ways. One, the value can be entered directly in the SLOC field as shown in Figure 1-4. Two, by using the function point model as shown in Figure 1-5. Three, by using Adaptation Adjustment Factor as shown in Figure 1-6. Upon completion click on OK. There is a limit to the range of input. The inputted value for SLOC must be within the range 0 - 9,999,999.

Note - COCOMO is not calibrated for Total SLOC < 2000.

FIGURE 1-4 **SLOC Dialog Box - Source Lines of Code (SLOC)**

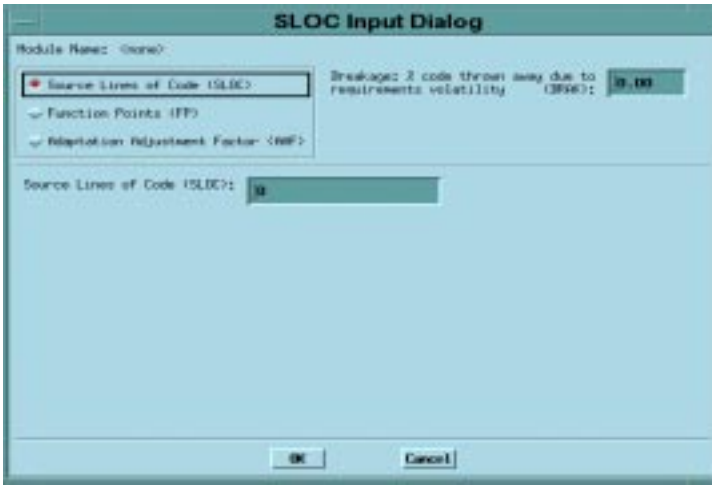


FIGURE 1-5 **SLOC Dialog Box - Function Points (FP)**

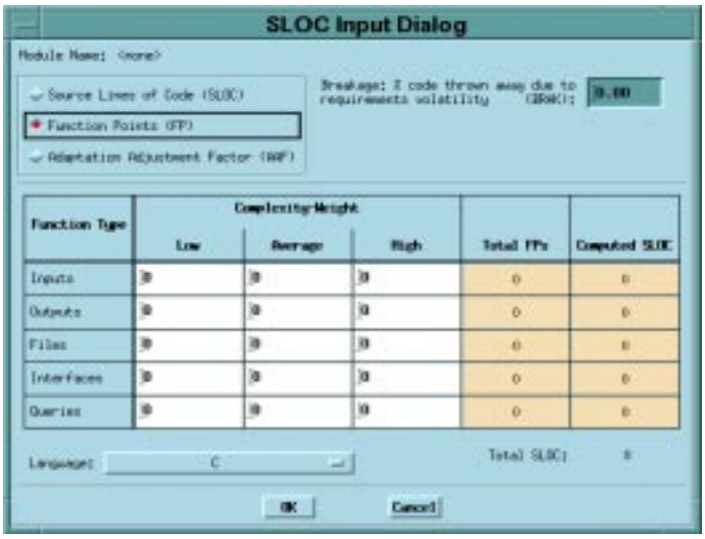
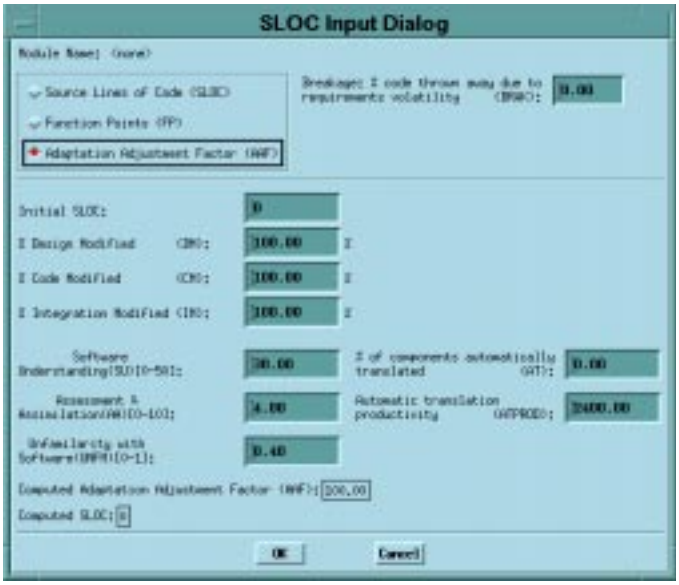


FIGURE 1-6 **SLOC Dialog Box - Adaptation Adjustment Factor (AAF)**



8. **Labor Rate Column** - This column contains the amount of money at which a developer working on a particular module would be paid per month. The labor rate can be

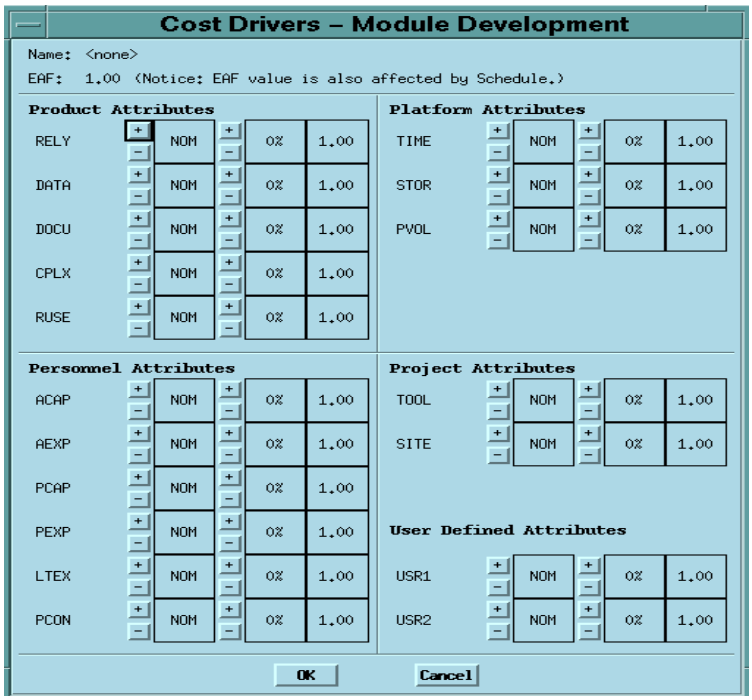
edited by clicking on the corresponding Labor Rate box and entering the new value via the edit area. The range on labor rate is between $0 and $99,999.

9. **Effort Adjustment Factor (EAF) Column** - This column displays the product of the cost drivers for each specific module. By clicking on this field a dialog box appears (see Figure 1-7). This box displays all of the cost drivers and their current ratings. The cost drivers are divided into the groupings: *Product, Platform, Personnel* and *Project*. The ratings for each multiplier can be changed by cycling through the available ratings using either the "+" or "-" buttons until the desired rating is displayed. As the cost driver ratings are changed the total product of the cost drivers is displayed in the upper right hand corner of the dialog box along with the module name.

10. **Totals Area** - This area houses the calculated results of all of the modules combined. Within this area is the total SLOC count for the module, the total nominal effort (PM), the total nominal productivity (SLOC/PM), the total estimated effort (EST PM), the total estimated productivity (Prod), the total estimated project cost, the estimated cost per instruction, the total estimated FSWP and the total estimated schedule for project completion (see each individual column for more information). The latter six quatities have not only a most likely estimate but also an optimistic estimate (no less than this, 90% of the time) and a pessimistic estimate (no greater than this, 90% of the time).

11. **Microhelp Window** - This window displays a short definition of the column headings clicked upon and also displays a short description of the result of the last function initiated by the user.

FIGURE 1-7 **EAF Dialog Box**



12. **Project Name** - This editable field displays the name of the currently displayed project. To edit the name click twice upon this field and proceed to edit name. Upon completion of editing press the "Return" key.

13. **Schedule Button -** This button displays the Schedule Dialog Box as shown in Figure 1-8.

14. **Scale Factor Button** - This button displays the Scale Factor Dialog Box as shown in Figure 1-9.

15. **Risk Column** - This column contains the Total risk level for each specific module. By clicking on this field a dialog box appears (see Figure 1-10). This box displays all of the risk levels for the chosen module. The total risk of a module is computed as:

total_risk=schedule_risk+product_risk+personnel_risk+process_risk+platform_risk+reuse_risk;

total risk of a module=total_risk/373.*100.;

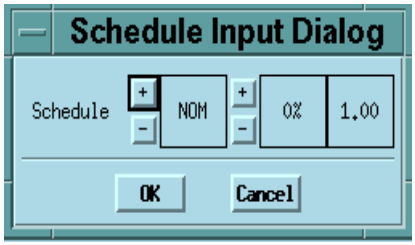
For the definitions of schedule risk, product risk, platform risk, personnel risk, process risk, and reuse risk, see [Madachy 1997].

16. **Full-time SoftWare Personnel (FSWP) Column** - This column houses the calculated most likely estimate for the number of full-time developers that would be needed to complete a module in the estimated development time.

17. **Instruction Cost Column** - This column contains the calculated most likely cost per instruction. This number is calculated from Cost/SLOC in each module.

18. **Cost Column** - This column contains the calculated most likely estimate of the development cost for a particular module.

FIGURE 1-8 **Schedule Dialog Box**



19. **Productivity (PROD) Column** - This column contains the calculated result of the module's individual SLOC divided by the the module's most likely effort estimate.

20 **Estimated Person-Month (EST PM) Column** - This column holds the module's most likely effort estimate obtained from multiplying Effort Adjustment Factor (EAF) by Nominal Person Month (NOM PM).

21. **Nominal Person-Month (NOM PM) Column** - This column holds the module's most likely effort estimate without incorporating the Effort Adjustment Factors (EAF).

FIGURE 1-9 **Scale Factor Dialog Box**

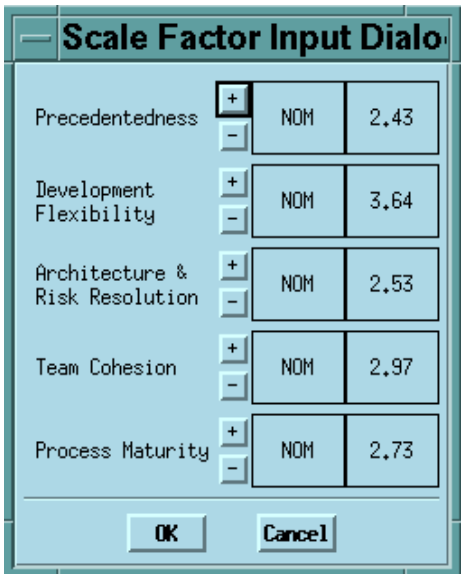
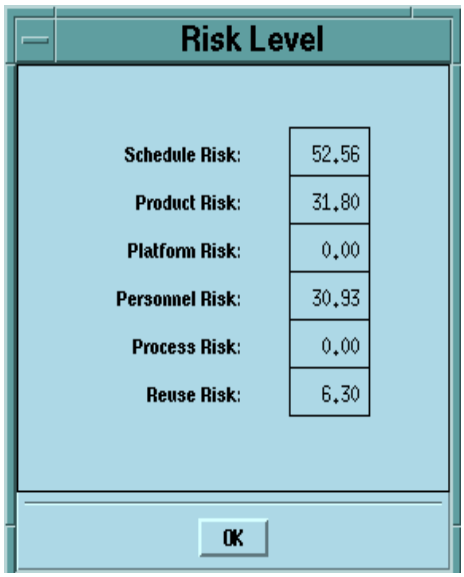


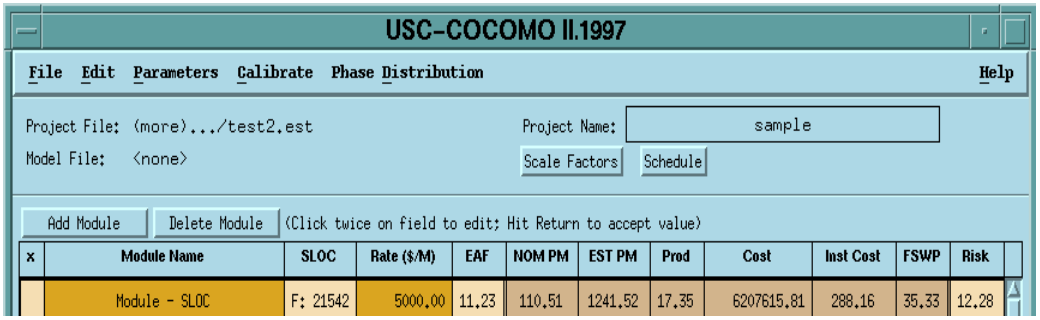
FIGURE 1-10 **Risk Level Dialog Box**



1.3 Begin Using COCOMO

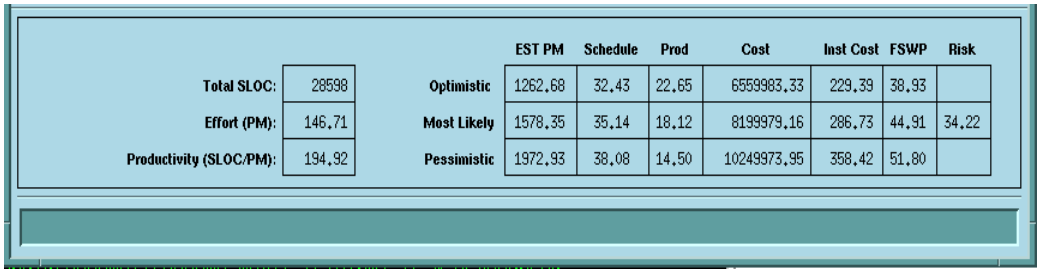
To begin entering a new module, click on the "Add Module" button. At this point, a new module will appear in the CLEF with all values set to their respective defaults. Double click upon the module name field in order to give the new module a name. Upon typing the module name press "Return.". A value for SLOC and Labor rate may also be given by clicking on the respective field and editing appropriately (see Figure 1-11).

FIGURE 1-11 **Create Sample Module and give values to SLOC and Labor Rate**



NOTE - In order to change any of the editable fields, just click on the desired field twice and begin editing the field. Upon completing editing, either hit the "Return" key, or click on OK. All of the final results can be found at the bottom of the CLEF in the Totals area (see Figure 1-12).

FIGURE 1-12 **Totals area after calculations have been completed**



1.4 Running COCOMO: Windows, Sparc or Java

Currently there are three implementations of COCOMO, a Windows 95 version, a Sun Microsystems Sparcstation version and a Java version.

To download any of these versions, you shold connect to:

http://sunset.usc.edu/COCOMOII/Cocomo.html

or

ftp://ftp.usc.edu/pub/soft_engineering/COCOMOII/

To run java cocomo (Netscape Navigator3.0 or Microsoft Internet Explorer3.0 or higher), you should connect to:

http://sunset.usc.edu/Ongoing_Research/topics.html

and click on Run Java Cocomo.

Chapter

2

File Menu

The COCOMO file types include: project file, model file, report file and calibration file. The first three are discussed here. The last is discussed in Chapter 5.

The project file in COCOMO stores a project's data, which include project name, project scale factors, project schedule constraint, module name, SLOC, labor rate, effort adjustment factors (EAF), and COCOMO related calculation results. The COCOMO system gives all project files an ".est" extension.

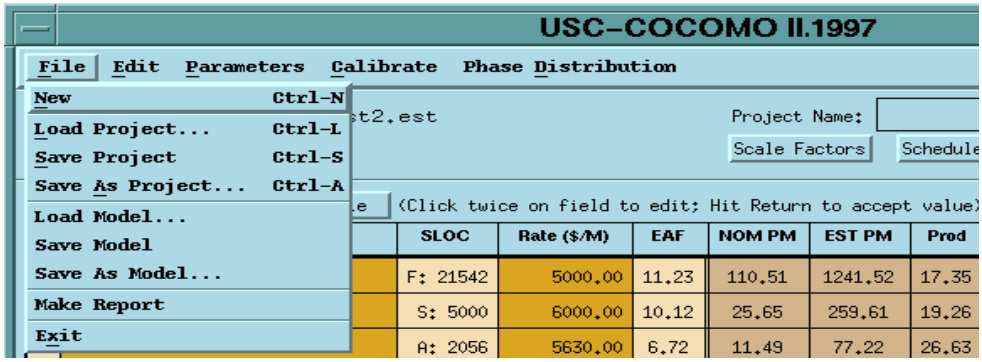
Regarding the model file, as we mentioned in chapter one, COCOMO incorporates 16 predictor factors, or cost driver attributes, which are grouped into four categories: software product attributes, platform attributes, personnel attributes, and project attributes. Each of these cost drivers attributes determines a multiplying factor which estimates the effect of the attribute in software development effort. There are also two user defined EAF factors plus the project-level required development schedule EAF factor. Besides these cost drivers, COCOMO also has scale factors. These multiplying factors and effort estimating equations constitue the model of a project. As we said previously, COCOMO has assigned default values and equations for the annually calibrated default model. Each time a COCOMO project is created, its effort estimate is based on the default model. COCOMO provides flexibility in changing the values of multiplying factors and effort estimating equation, or schedule estimating equation. An adjusted model can no longer be considered default values, and therefore it will be lost if COCOMO is exited without saving it in a model file. Upon saving this model file, these altered values can be applied to another project by loading the saved model file. The COCOMO system gives all model files a ".mod" extension.

The report file is a summary report of the COCOMO project. This report contains all edited and calculated values of a project. These files are given a ".rpt" extension.

The File menu option will enable you to create, retrieve, save, or print COCOMO files.

To select the File menu and its options, click on File with the mouse. The File menu will appear as Figure 2-1.

FIGURE 2-1 **File Menu**



2.1 New

The New option creates a new project file in the COCOMO working window, replacing any previous project file in the working window.

To Create a New Working File

1. Choose New from the File menu with mouse.

The working window will now be clear; the previous project file in the working window has been removed.

Note: New can be selected anytime; however, if the previous project file or model file has been modified, a warning dialog box will appear and requests confirmation. (as seen in Figure 2-2)

FIGURE 2-2 **Warning Dialog Box**

2. If the modifications on the previous file are not to be saved, choose Yes, otherwise choose No. If the No is selected, a File Save dialog will appear. (see Project Save and Model Save respectively)

2.2 Project Load

The Project Load option is used to retrieve a project file as well as loading it on the working window.

To Retrieve or Load a Project File

1. Choose Project Load from the File menu with the mouse.

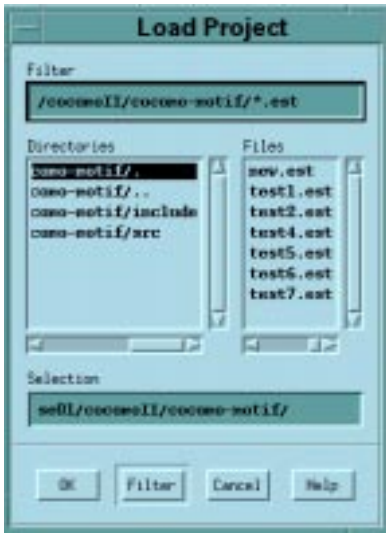
2. If a previous project file has been modified in the working window, the dialog box as in Figure 2-3 will appear.

FIGURE 2-3 **Warning Dialog Box**

3. If the previous project file is to be saved, choose Yes, then a File Save dialog box will appear. (see Project Save). If the modified file is not to be saved, choose No.

4. The Project Load dialog box will appear as seen in Figure 2-4.

FIGURE 2-4 **Project Load Dialog Box**



The file name of a COCOMO project has a default format with ".est" as an extension.

With this window, the desired project file can be selected from the Files scroll list for loading. If the desired project file does not exist in the scroll list, it is necessary to choose an appropriate directory.

5. Choose desired directory for file loading

Look at the filter input box. The path found in this box represents the path to be searched for loading a project file. This path will be changed after each directory change. To select the desired directory, click the appropriate directory choice from the Directories scroll list, then click the "Filter" button. At this point, the files located under this directory will be listed in the Files scroll list.

6. When the desired file is shown on the Files list, click it, and click the "OK" button to initiate project loading.

7. After a project file is loaded, its file name will be displayed on the PROJECT FILE field at upper left corner on the working window, and all modules and related items will be displayed in the CLEF area. If the number of modules are beyond the window scope, the scroll bar can be used to look at all items.

2.3 Project Save

The Project Save option is used to store the results of the current COCOMO project as a file with ".est" extension.

To Store the Results of Current Project

1. Choose Project Save from the File menu with the mouse. If the current project is loaded from a previously stored project file, the Project Save will overwrite the same project file with the current project.

2. If the current project is a new one, i.e., being created by the New command, the Project Save dialog box will appear, as seen in Figure 2-5.

FIGURE 2-5 **Project Save Dialog Box**

3. Look at the Files scroll window. If the file saving is to update (overwrite) a existing project file, the desired filename should be found in the Files scroll list. If the filename can not be found from current list, change the directory from the Directories scroll list until the desired filename is being shown. When the desired filename is on the list, click it.

4. If the file saving is to store a new project file, choose the desired directory, then type in a new filename.

5. After the desired filename is selected or inputted, click the OK button to initiate project saving.

2.4 Project Save As

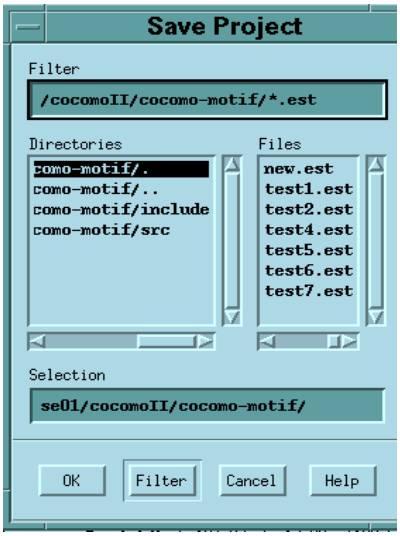
The Project Save As option is to store the current project as a COCOMO project file, which has a file name different from current file.

To Store Current Project With different File Name

1. Choose Project Save As from the File menu with the mouse.

2. The Project Save dialog box will appear, as seen in Figure 2-6.

FIGURE 2-6 **Project Save Dialog Box**



3. Look at the Files scroll window. If the file saving is to update (overwrite) a existing project file, the desired filename should be found in the Files scroll list. If the filename can not be found from current list, change the directory from the Directories scroll list until the desired filename is being shown. When the desired filename is on the list, click it.

4. If the file saving is to store a new project file, choose the desired directory, then type in a new filename in the SELECTION box.

5. After the desired filename is selected or inputted, click the OK button to initiate project saving. After a project file is saved, the project file name will be displayed on the PROJECT FILE field at the upper left corner of the working window.

2.5 Model Load

The Model Load command is used when a specific model, in which the values of multiplying factors and scale factors are different from the COCOMO default model, is to be applied to the current project.

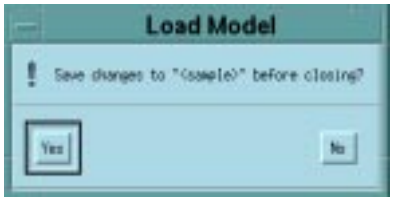
The Model Load option is used to retrieve a model file as well as loading it for the current project.

To Retrieve or Load a Model File

1. Choose Model Load from the File menu.

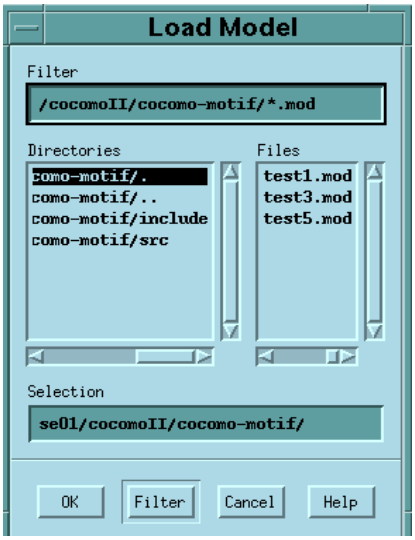
2. If a previous model has been modified in the current project, the following dialog box will appear.

FIGURE 2-7 **Warning Dialog Box**



3. If the previous model file is to be saved, choose Yes, then a Model Save dialog box will appear. (see Model Save). If the modified model is not to be saved, choose No.

FIGURE 2-8 **Model Load Dialog Box**



4. The Model Load dialog box will appear as seen in Figure 2-8.

The file name of a COCOMO model has a default format with ".mod" as an extension. With this window, the desired model file can be selected from the Files scroll list for loading. If the desired model file does not exist in the scroll list, look for it in the other directories.

5. Choose desired directory for file loading

Look at the filter input box. The path found in this box represents the path to be searched for loading a model file. This path will be changed after each directory change. To select the desired directory, click the appropriate directory choice from the Directories scroll list, then click the "Filter" button. At this point, the files located under this directory will be listed in the Files scroll list.

6. When the desired file is shown on the Files list, click it, and click the "OK" button to initiate model loading.

7. After a model file is loaded, its file name will be displayed on the MODEL FILE field at upper left corner on the working window, and the related costs of current project will be recalculated and shown on the working window.

2.6 Model Save

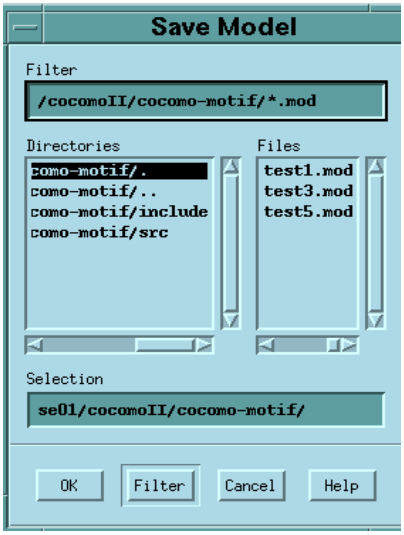
The Model Save option is used to store the results of the current COCOMO model as a file with ".mod" extension.

To Store the Results of Current Model

1. Choose Model Save from the File menu. If the current model is loaded from a previously stored model file, the Model Save will overwrite the same model file with the current model.

2. If the current model is a new one, the Model Save dialog box will appear, as seen in Figure 2-9.

FIGURE 2-9 **Model Save Dialog Box**



3. Look at the Files scroll window. If the file saving is to update (overwrite) a existing model file, the desired filename should be found in the Files scroll list. If the filename can not be found from current list, change the directory from the Directories scroll list until the desired filename is being shown. When the desired filename is on the list, click it.

4. If the file saving is to store a new model file, choose the desired directory, then type in the filename.

5. After the desired filename is selected or inputted, click the OK button to initiate model saving.

2.7 Model Save As

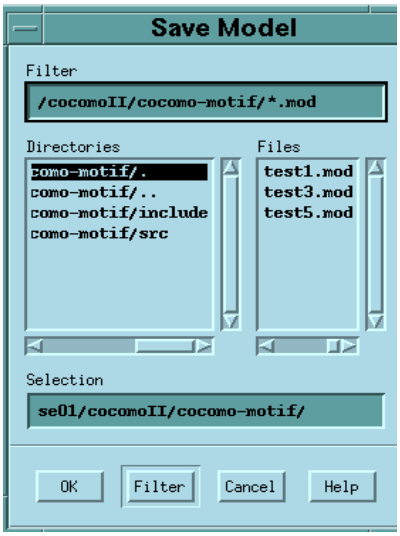
The Model Save As option is to store the current model as a COCOMO model file, which has a file name different from current model.

To Store Current Model With different File Name

1. Choose Model Save As from the File menu.

2. The Model Save dialog box will appear, as seen in Figure 2-10.

FIGURE 2-10 **Model Save Dialog Box**



3. Look at the Files scroll window. If the file saving is to update (overwrite) a existing model file, the desired filename should be found in the Files scroll list. If the filename can not be found from current list, change the directory from the Directories scroll list until the desired filename is being shown. When the desired filename is on the list, click it.

4. If the file saving is to store a new model file, choose the desired directory, then type in the filename in the SELECTION box.

5. After the desired filename is selected or inputted, click the OK button to initiate model saving. After a model file is saved, the project file name will be displayed on the MODEL FILE field at the upper left corner of the working window.

2.8 Make Report

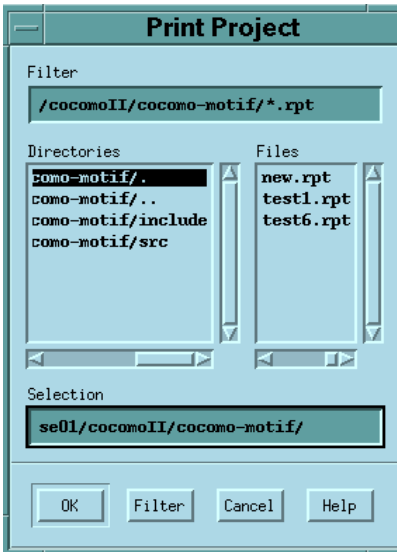
The Make Report option creates a COCOMO report in the form of a text file for printing.

To Create Project Report

1. Choose Make Report from the File menu.

2. The Make Report dialog box will appear, as seen in Figure 2-11.

FIGURE 2-11 **Make Report Dialog Box**



3. Look at the Files scroll window. If the file saving is to update (overwrite) a existing report file, the desired filename should be found in the Files scroll list. If the filename can not be found from current list, change the directory from the Directories scroll list until the desired filename is shown. When the desired filename is on the list, click it.

4. If the file saving is to store a new report file, choose the desired directory, then type in the filename.

5. Choose desired directory for file saving: Look at the filter input box. The path found in this box represents the directory where the report file is going to be saved. This path will be changed after each directory change. To change the directory, click the appropriate directory choice from the Directories scroll list, then click the "Filter" button.

6. After the desired filename is selected or inputted, click the OK button to initiate report file saving.

7. To print a COCOMO project report, execute the local commands for your system in order to send the file for printing.

2.9 Exit

The Exit option leaves the COCOMO system.

To Leave COCOMO System

1. Choose Exit from the File menu with the mouse.

2. This causes your system to terminate the cocomo program.

Chapter

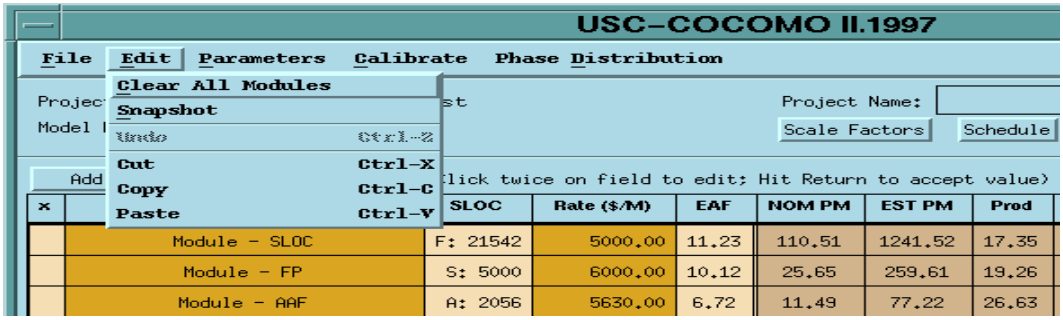
3

Edit Menu

The Edit Menu option supplies several useful commands which will enable you to establish a project more conveniently. With the Edit Menu options, the entering of modules will be easier.

To select the Edit menu and its options, click on Edit with the mouse, then the Edit menu will appear as Figure 3-1.

FIGURE 3-1 **Edit Menu**



3.1 Clear

The Clear option erases all modules of the current project on the working window.

To Erase All Modules of Current Project

1. Choose Clear from the Edit menu.

During the execution of the Clear command, if some changes have occurred on the currently viewed project and have not been saved, the warning dialog box will appear as Figure 3.2.

2. If you really want to clear, click Yes. If not, click No.

3. After Clear, all modules of current project will disappear.

FIGURE 3-2 **Warning Dialog Box**

3.2 Snapshot

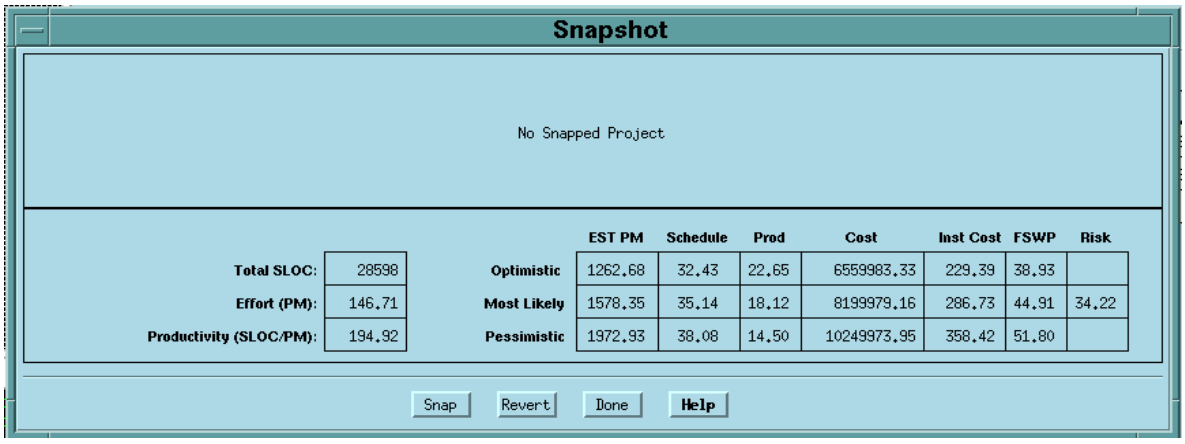
The Snapshot option enables users to compare the effort estimation change for a project so that he/she can decide to apply the change or not. This function makes COCOMO more convenient and powerful for software project decision analyses.

The Snapshot command stores the current set of modules, effort adjustment factors and all other data associated with a project. At a later time this data can be restored.

To Compare the Overall Change of a Project

1. Choose Snapshot from the Edit menu. The Snapshot dialog box will initially appear as Figure 3-3.

FIGURE 3-3 **Snapshot Dialog Box-1**

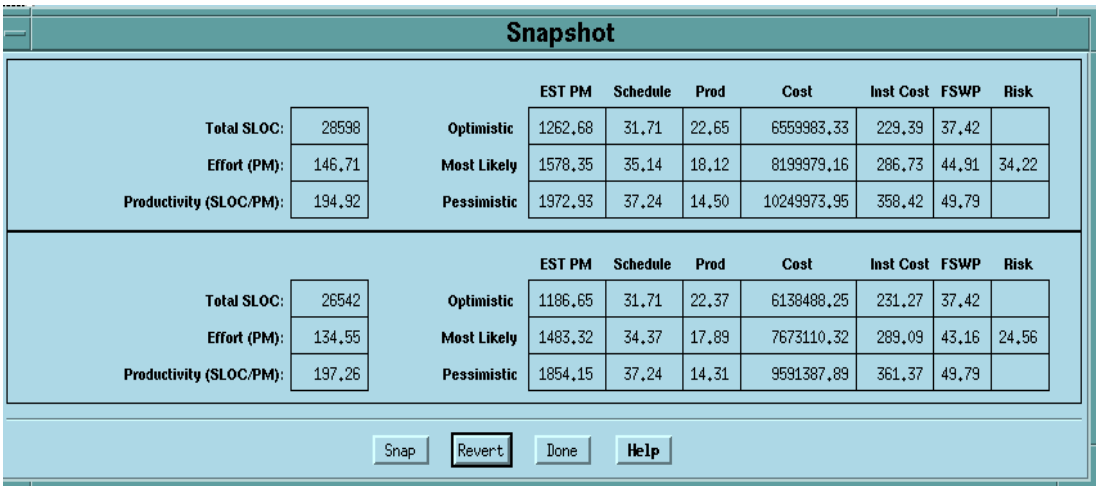


In the dialog box, the lower section represents the current results for the project. The upper section is previously snapped results. The current project can be snapped by clicking upon the Snap button. After completing this action the upper and lower section of the Snapshot window will contain identical information. At this point changes can be made to the current project values after clicking upon the Done button.

2. Upon completing the modification of the project values, a comparison can be made between the previously snapped project and the modified project by clicking again upon the Snapshot option in the Edit menu.

3. Now the values in the upper part of the window will likely be different from the current values, in the lower part. To restore the upper values, click on Revert. The two sets of values are interchanged.

FIGURE 3-4 **Snapshot Dialog Box-2**



4. When finished, click the Done button.

3.3 Undo

The Undo option retracts the previous cut or paste done on a module.

To Retract Previous Cut/Paste for a Module

1. Choose Undo from the Edit menu with the mouse.

2. The changed module will go back to its previous status.

3.4 Cut

The Cut option copies a module into the cut buffer and removes it from the current project. The cut module can be used for Paste.

To Cut a Module and Remove It From the CLEF

1. Check the module which is to be cut. The Check boxes for modules are located in the leftmost column of the CLEF area. Place the mouse in the box just to the left of the module name, and click.

2. Choose Cut from the Edit menu with the mouse.

3. The cut module disappears.

3.5 Copy

The Copy option copies a module. The copied module can be used for Paste.

To Copy a Module

1. Check the module which is to be copied. The Check boxes for modules are located in the leftmost column of the CLEF area.

2. Choose Copy from the Edit menu with the mouse.

3. The cross sign in the check box disappears.

3.6 Paste

The Paste option pastes a previously copied or cut module in the CLEF.

To Paste a Previously Copied or Cut Module

1. Check the module above which the previously copied or cut module is to be pasted. The Check boxes for modules are located in the leftmost column of CLEF area.

2. Choose Paste from the Edit menu with the mouse.

3. The pasted module appears at the checked position, and the modules lower than it were pushed one row down.

4. If there is no module checked, the Paste will attach the previously copied or cut module at the end.

Chapter

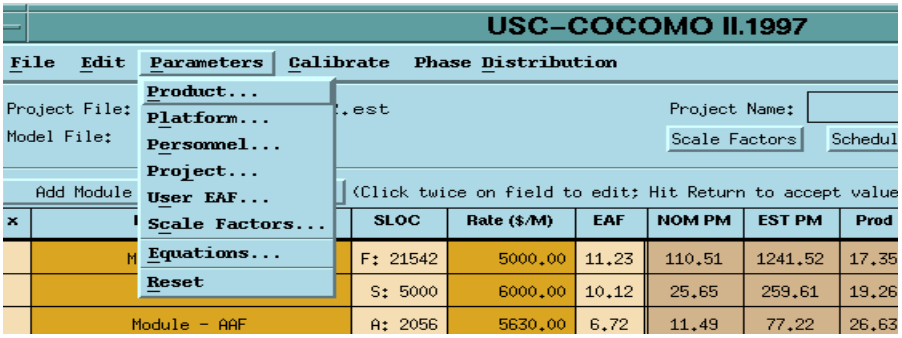
4

Parameters Menu

The Parameters menu option will enable you to look at, or change the values of effort adjustment factors, scale factors and effort/schedule estimating equations factors for the current project.

To choose the Parameters menu and its options, click on Calibrate with the mouse. The Parameters menu will appear as Figure 4-1.

FIGURE 4-1 **Parameters Menu**

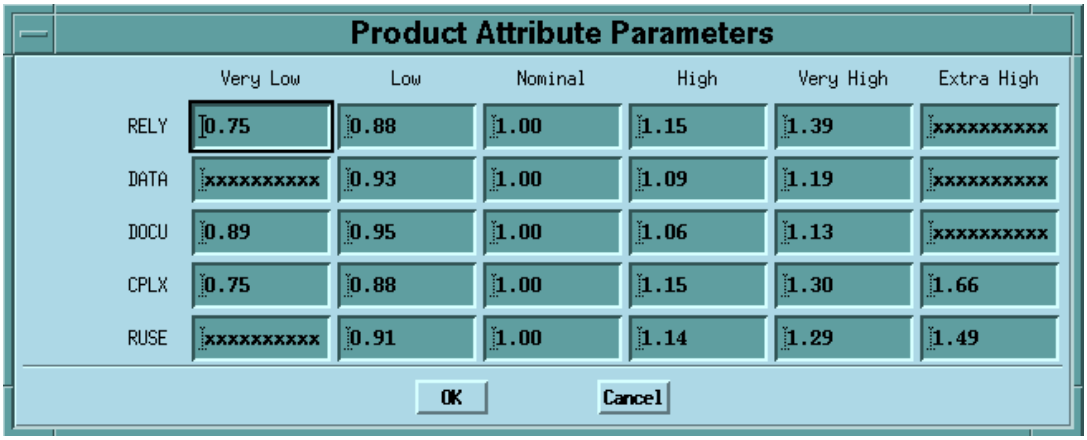


4.1 Product

The Product option displays five cost drivers: RELY, DATA, DOCU, CPLX, and RUSE and their corresponding ratings and multiplier values.

Select Product from the Parameters menu with the mouse. The Product Dialog Box will appear as Figure 4-2.

FIGURE 4-2 **Product Dialog Box**



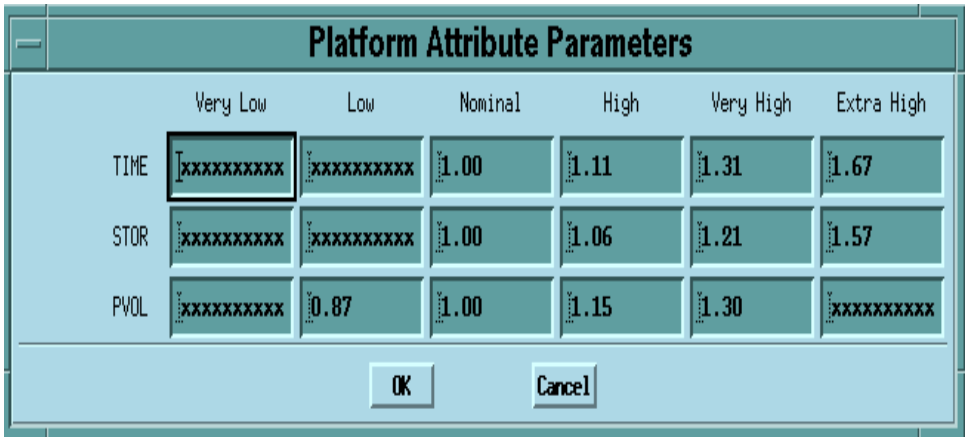
To modify these values, go straight to those edit boxes and type new values. When finished with the modification, click the OK button

4.2 Platform

The Platform option displays three cost drivers: TIME, STOR and PVOL, and their corresponding ratings and multiplier values.

Select Platform from the Parameters menu with the mouse. The Platform Dialog Box will appear as Figure 4-3.

FIGURE 4-3 **Platform Dialog Box**



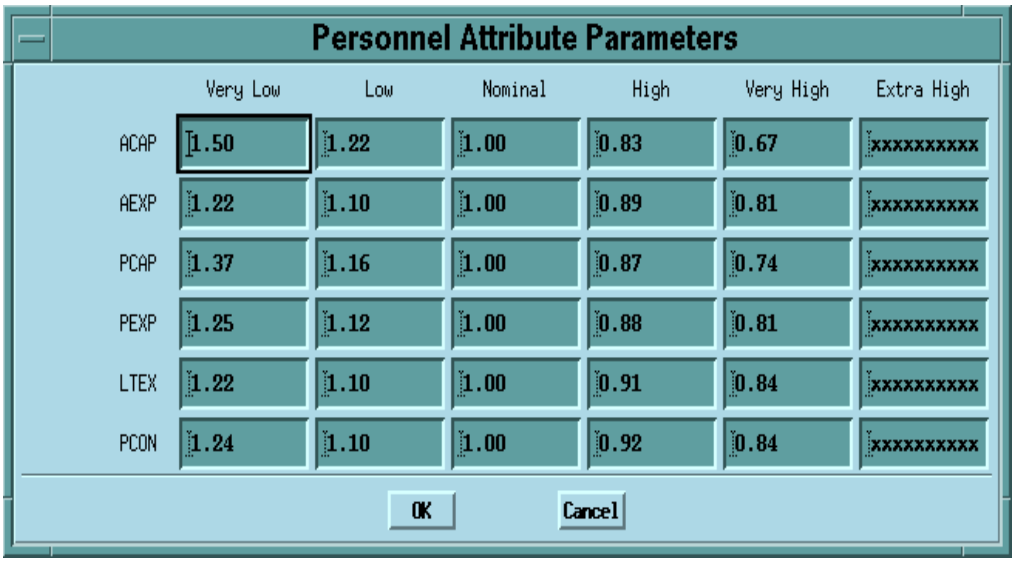
To modify these values, go straight to those edit boxes and type new values. When finished with the modification, click the OK button.

4.3 Personnel

The Personnel option displays six cost drivers: ACAP, AEXP, PCAP, PEXP, LEXP, and PCON and their corresponding ratings and multiplier values.

Select Personnel from the Parameters menu with the mouse. The Personnel Dialog Box will appear as Figure 4-4

FIGURE 4-4 **Personnel Dialog Box**



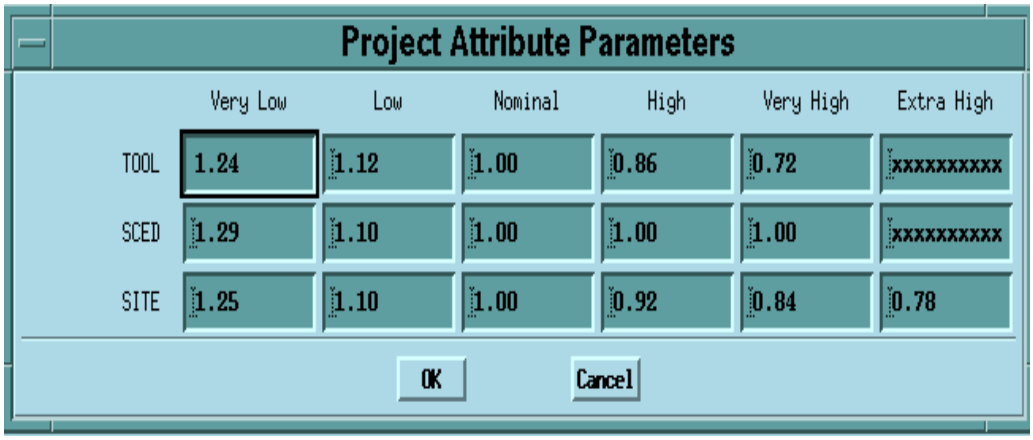
To modify these values, go straight to those edit boxes and type new values. When finished with the modification, click the OK button.

4.4 Project

The Project option displays three cost drivers: TOOL, SCED, and SITE and their corresponding ratings and multiplier values.

Select Project from the Parameters menu with the mouse. The Project Dialog Box will appear as Figure 4-5.

FIGURE 4-5 **Project Dialog Box**



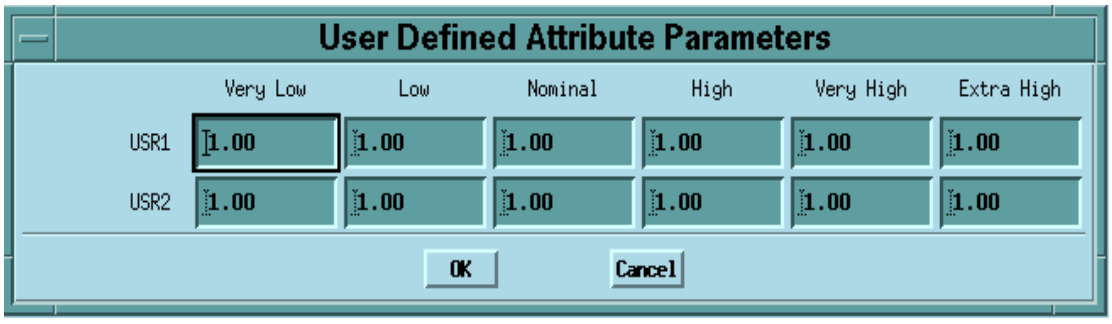
To modify these values, go straight to those edit boxes and type new values. When finished with the modification, click the OK button.

4.5 User Defined EAF

The User Defined EAF option displays two cost driver: USR1 and USR2, and their corresponding ratings and multipliers.

Select User EAF from the Parameters menu with the mouse. The User EAF Dialog Box will appear as Figure 4-6.

FIGURE 4-6 **User Defined EAF Dialog Box**



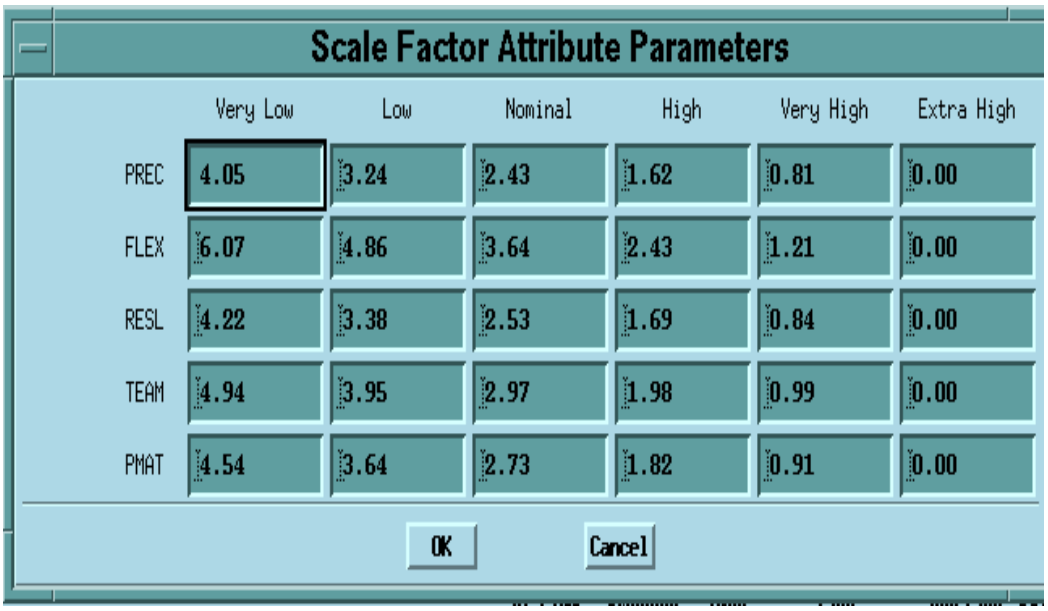
To modify these values, go straight to those edit boxes and type new values. When finished with the modification, click the OK button.

4.6 Scale Factors

The Scale Factors option displays five development attributes: PREC, FLEX, RESL, TEAM and PMAT, and their corresponding ratings and values.

Select Scale Factors from the Parameters menu with the mouse. The Scale Factor Dialog Box will appear as Figure 4-7.

FIGURE 4-7 **Scale Factors Dialog Box**



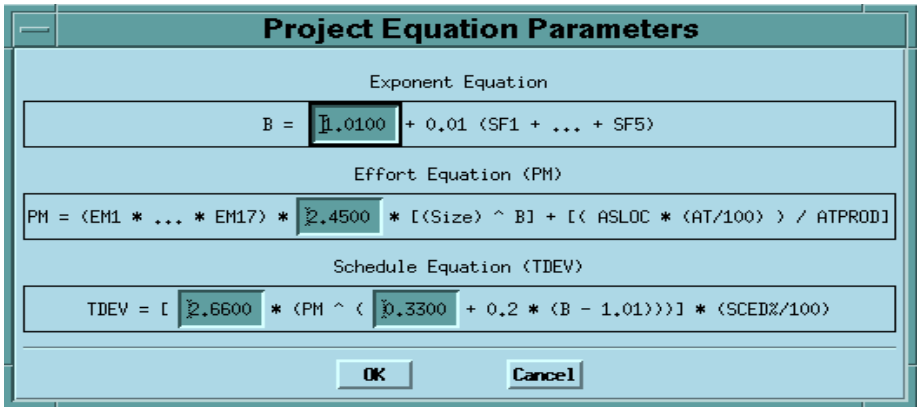
To modify these values, go straight to those edit boxes and type new values. When finished with the modification, click OK button.

4.7 Equation

The Equation option displays effort and schedule equations.

Select Equation from the Parameters menu with the mouse. The Equation Dialog Box will appear as Figure 4-8.

FIGURE 4-8 **Equation Dialog Box**



To modify these values, go straight to those edit boxes and type new values. When finished with the modification, click the OK button.

4.8 Reset

The Reset option resets the values of multiplying factors and effort/schedule estimating equations of the current project back to the COCOMO default values.

Select Reset from the Parameters menu with mouse. The command will be executed directly, and there is no warning message for users. After the RESET, the values of all multiplying factors and effort estimating equations of current project will be changed to the COCOMO default values

Chapter

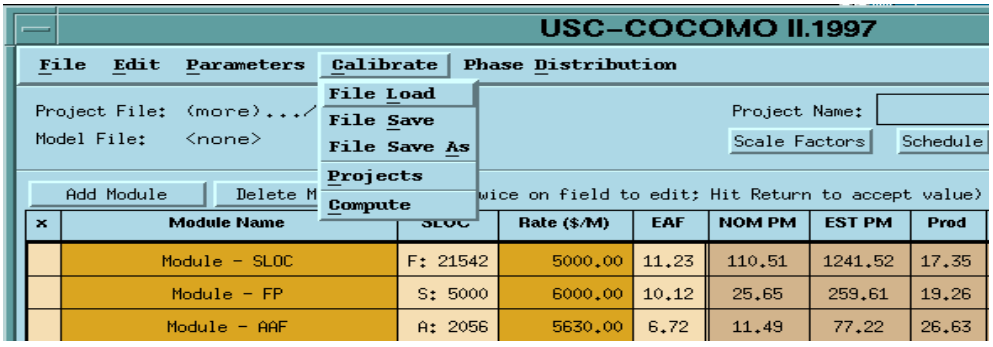
5

Calibrate Menu

COCOMOII now has the ability to archive your own software project data. Using this data, COCOMOII will compute various coefficients and exponents involved in the effort and schedule equations. This will make your COCOMOII estimates even more reliable.

Each software project to be archived is described as a complete COCOMOII project. It may include multiple modules, each with their own SLOC estimate and EAF factors. In addition, a software project consists of a name, date/time, actual effort and actual schedule. The actual effort and actual schedule must be supplied by the COCOMOII user. Entering revised values for effort and schedule are always possible. Effort is given in units of person/months. Schedule is given in units of months.

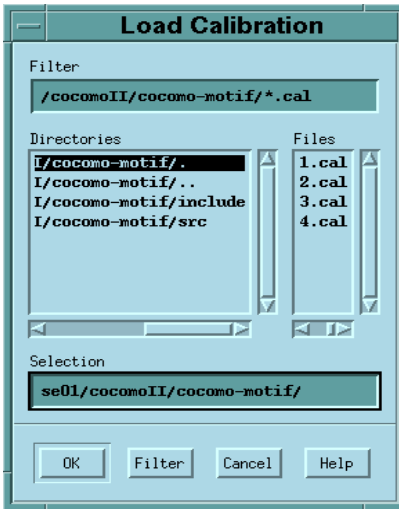
FIGURE 5-1 **Calibrate Menu**



5.1 File Load

The Calibrate File Load command is used when a specific set of calibration data is desired. Calibration files are named with *.cal as the suffix.

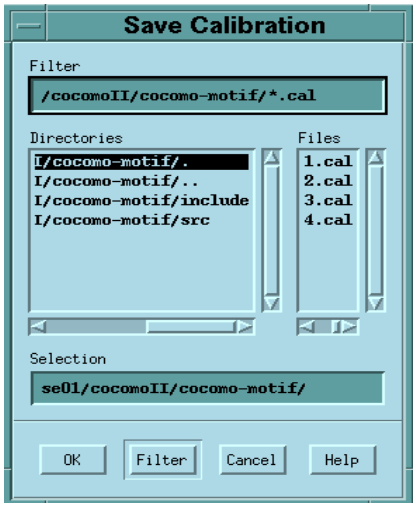
FIGURE 5-2 **Load Calibration Dialog Box**



5.2 File Save

The Calibrate File Save command saves the current calibration data in the file whose name was previously identified using File Save As.

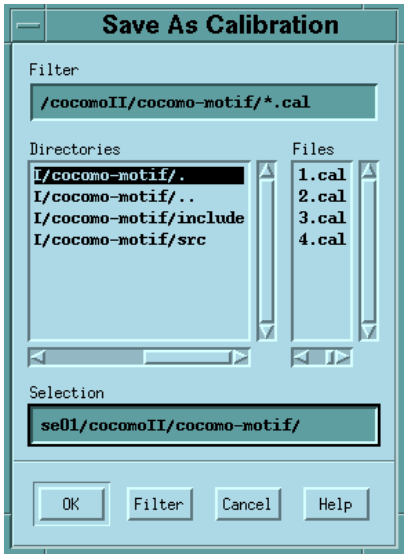
FIGURE 5-3 **Save Calibration Dialog Box**



5.3 File Save As

The Calibrate File Save As is to store the current calibration data as a *.cal file which has a different file name from the current file. This command works precisely the same as the File Save As for *.est and *.mod files

FIGURE 5-4 **Save As Calibration Dialog Box**



5.4 Projects

A windows appears (shown in FIGURE 5-5) which displays the archived project data.

- To remove the window, click on Cancel.

- To delete an existing entry, first place an x at the leftmost end of the row and click on Delete. a warning box appears as shown in FIGURE 5-6.

- To display the entire set of values for an archived project, click on Display. Since the display of an archived project eliminates the display of any existing CLEF data, a warning message appears as shown in FIGURE 5-7.

- To insert a new archived project from the CLEF, click on Insert

FIGURE 5-5 **Projects Dialog Box**

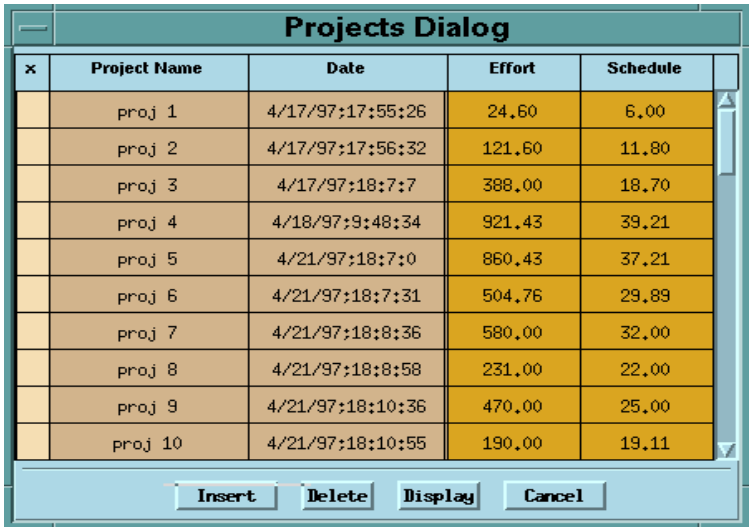
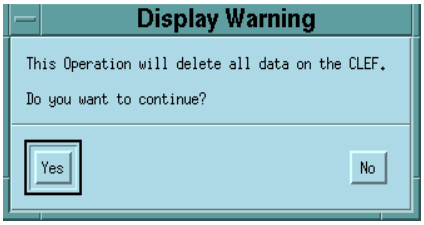


FIGURE 5-6 **Delete Warning Dialog**

FIGURE 5-7 **Display Warning Dialog**



5.5 Compute

This command takes all of the data that has been archived and uses it to compute new constant and exponent values for the effort equation and similarly for the schedule equation. There are two options to calibrate equation parameters. One is the Constant Term and the other one is Development Mode. Those two options are explained below in detail. They are displayed in this window and compared to the values currently used by COCOMOII. To get COCOMOII to use these values, click on Accept.

5.5.1 Calibrating the Constant Term

COCOMOII calibrates only constant values in the Constant Term. The coefficients are calibrated using the formula in pp525-526 of the book, “Software Engineering Economics, Prentice Hall 1981". To get the most appropriate constant c, for the man-month equation:

(EQ 1)

$$MM = c(KDSI)^{1.05}\Pi(EM)$$

After inserting the actual effort, we wish to solve the constant c in the following linear equations

(EQ 2)

$$MM_1 = c(KDSI_1)^{1.05}\Pi_1$$
$$MM_2 = c(KDSI_2)^{1.05}\Pi_2$$
$$\vdots$$
$$MM_n = c(KDSI_n)^{1.05}\Pi_n$$

From the above equation, we can derives the sum of the squares of the linear equations which minimizes the residual errors:

(EQ 3)

$$S = \sum_{i=1}^{n} [c(KDSI_i)^{1.05}\Pi_i - MM_i]^2$$

By setting the derivative dS/dc equal to zero, we can derive the most appropriate constant for the man-month equation.

(EQ 4)

$$0 = dS/dc = 2\sum_{i=1}^{n} c(KDSI_i)^{1.05}\Pi_i - \sum_{i=1}^{n} MM_i(KDSI_i)^{1.05}\Pi_i$$

$$\bar{c} = \frac{\sum_{i=1}^{n} MM_i(KDSI_i)^{1.05}\Pi_i}{\sum_{i=1}^{n} ((KDSI_i)^{1.05}\Pi_i)^2}$$

The constant of the schedule equation is calibrated in the same way using the above method.

5.5.2 Calibrating the Software Development Mode

The new coefficients and exponents for the equations are calibrated using the formula in pp526-528 of the book, “Software Engineering Economics, Prentice Hall 1981". Those parameters are computed as:

(EQ 5)

$$MM = c(KDSI)^b\Pi(EM)$$

For COCOMO’s effort equation, a similar least square method is used to calibrate the coefficient and exponent. The first step is to rearrange the above man-month equation by taking natural logarithm.

(EQ 6)

$$\log c + b\log KDSI = \log(MM/\Pi)$$

If there are completed a number of projects P_1, P_2,....., P_n with $KDSI_1$, $KDSI_2$, ..., $KDSI_n$ and actual efforts MM_1, MM_2,.....,MM_n,

(EQ 7)

$$\begin{aligned}
\log c + b\log(KDSI)_1 &= \log(MM/\Pi)_1 \\
\log c + b\log(KDSI)_2 &= \log(MM/\Pi)_2 \\
\log c + b\log(KDSI)_3 &= \log(MM/\Pi)_3 \\
&\vdots \\
\log c + b\log(KDSI)_n &= \log(MM/\Pi)_n
\end{aligned}$$

In order to determine the optimal values log $\bar{c}$ and $\bar{b}$, the following equation is used

(EQ 8)

$$\begin{aligned}
a_0\log\bar{c} + a_1\bar{b} &= d_0 \\
a_1\log\bar{c} + a_2\bar{b} &= d_1
\end{aligned}$$

Where a_0, a_1, a_2, d_0, and d_1 are calculated as:

(EQ 9)

$$a_1 = \sum_{i=1}^{n} \log(KDSI)_i$$

$$a_2 = \sum_{i=1}^{n} [\log(KDSI)_i]^2$$

$$d_0 = \sum_{i=1}^{n} \log(MM/\Pi)_i$$

$$d_1 = \sum_{n} \log(MM/\Pi)_i \log(KDSI)_i$$

The resulting coefficient and exponent for man-month equation are calculated as:

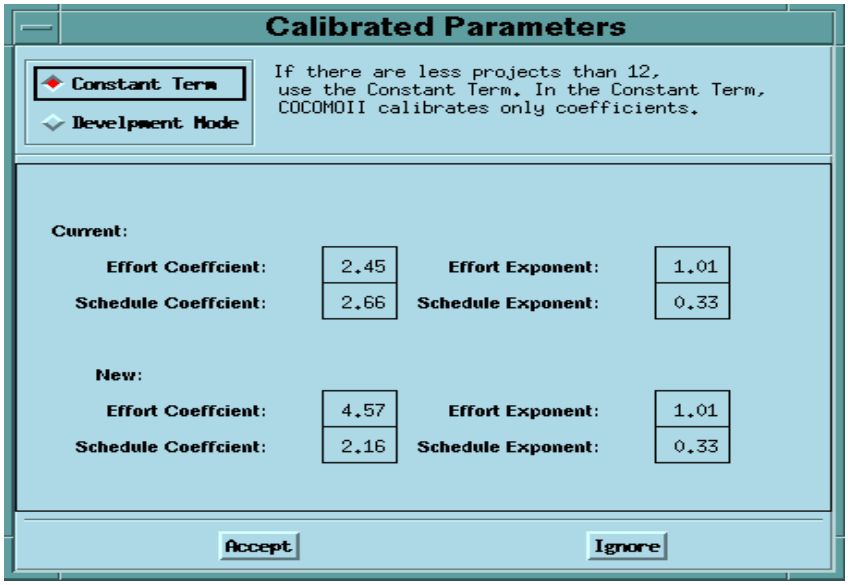
(EQ 10)

$$\log\bar{c} = \frac{a_2 d_0 - a_1 d_1}{a_0 a_2 - a_1^{\,2}}$$

$$\bar{b} = \frac{a_0 d_1 - a_1 d_0}{a_0 a_2 - a_1^{\,2}}$$

The calibration of coefficient and exponent for the schedule equation are computed in the same way using the above method.

FIGURE 5-8 **Compute Dialog Box**



Chapter

6

Phase Distribution

The Phase Distribution is one of the menu selections in the menu bar that can be accessed by either clicking upon Phase Distribution in the main menu. Its function is to display a breakdown of the software effort and schedule into the phases of the development cycle. These phases are plans & requirements, design, programming and integration & test. These phases are described as follows:

Plan & Requirements - In this phase, a statement for the required functions, interfaces and performance is created. These expectations are used to define the capabilities of the software product as expressed by representatives of all interested parties.

Design - In this phase, a hardware/software architecture, control structure and data structure for the product are defined. A draft of the users' manual and test plans are also created during this phase.

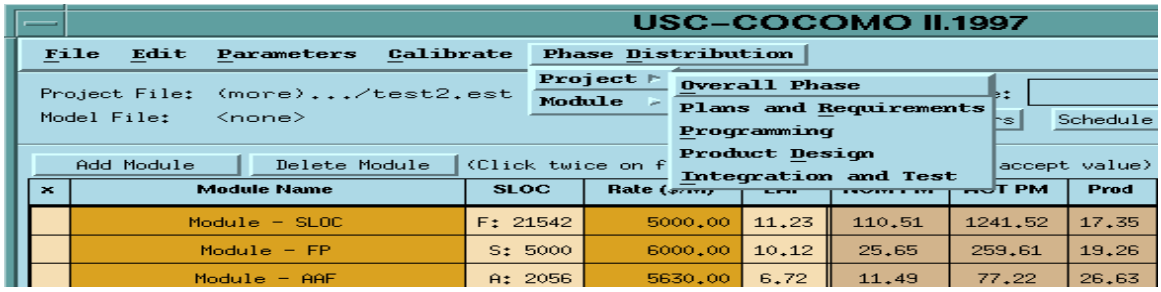
Programming - In this phase, the design of the previous phase is implemented in the creation of complete sets of software components.

Integration & Test - In this phase, the various software components are brought together in order to achieve a properly functioning software product composed of loosely coupled modules. The requirements as defined in the first phase are used to determine the fitness of the delivered product.

The phase distribution menu has two selections: project phase distribution and module phase distribution. The project phase distribution allows the user to view the development phases for the entire project all together or individually. The module phase distribution allows the user to view the development phases for a particular module either all together or individually. These two variations of phase distribution are discussed further in this chapter under sections 6.1 and 6.2 in this chapter.

NOTE: These phase distribution estimates are retained from the COCOMO81 model, which assumed a waterfall (sequentially phased) process model. If your project's process model is not a waterfall, these phase distributions are either inapplicable or need to be reinterpreted.

FIGURE 6-1 **Phase Distribution Sub-menu**



6.1 Project Phase Distribution

In order to view the phase distribution of an entire project, the user can click on the Project Phase Distribution button under the Phase Distribution menu (see FIGURE 6-1). Four formats for viewing will appear in another menu: overall phase, plan & requirements, programming, and integration & test. Each of these menu selections will be discussed in sections 6.1.1 - 6.1.4, respectively. The phase distribution of plan & requirements, programming and integration & test are broken down into sub-phases. These phases include: requirements analysis, product design, programming, test planning, verification & validation, project office, CM/QA, and manuals. For each of these sub-phases the percentage of the phase, the estimated effort, the estimated schedule, and the estimated FSWP is displayed. A description of each of these sub-phases follows:

Requirements analysis: Determination, specification review and update of software functional, performance, interface, and verification requirements.

Product Design: Determination, specification, review and update of hardware-software architecture, program design, and database design.

Programming: Detailed design, code, unit test, and integration of individual computer program components. Includes programming personnel planning, tool acquisitions, database development, component level documentation, and intermediate level programming management.

Test Planning: Specification, review, and update of product test and acceptance test plans. Acquisition of associated test drivers, test tools, and test data.

Verification & Validation(V&V): Performance of independent requirements validation, design V&V, product test, and acceptance test. Acquisition of requirements and design V&V tools. "Are we building the product right?" and "are we building the right product?"

Project Office Functions: Project level management functions. Includes project level planning and control, contract and subcontract management, and customer interface.

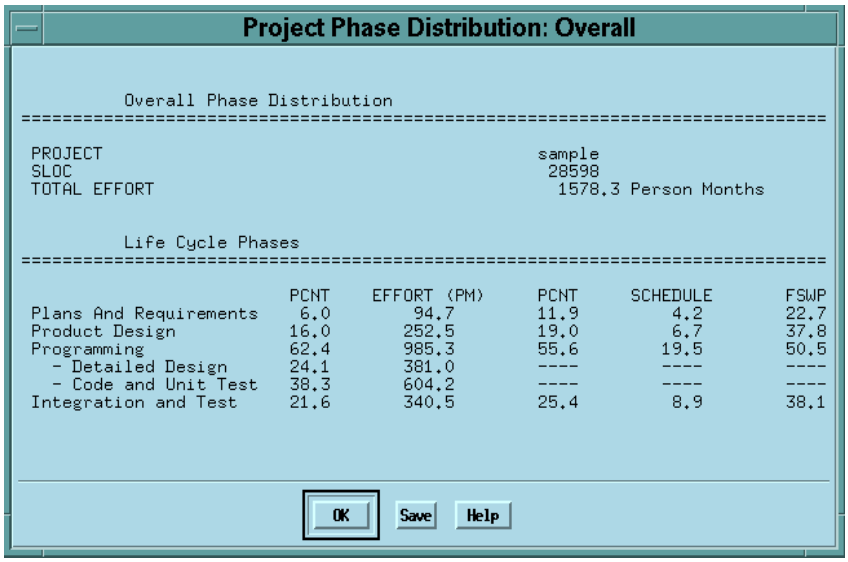
Configuration Management and Quality Assurance (CM/QA): Configuration management includes product identification, change control, status accounting, operation of program support library, development and monitoring of end item acceptance plan. Quality assurance includes development and monitoring of project standards, and technical audits of software products and processes.

Manuals: Development and update of users' manuals, operators' manuals and maintenance manuals.

6.1.1 Overall Project Phase Distribution

The overall phase distribution allows the user to view an entire project's estimated effort, schedule and number of personnel needed for phase completion. Upon clicking on "Overall Phase," a window will be displayed showing the phase breakdown of the current project in COCOMO (see FIGURE 6-2). This window displays the project name, project SLOC, and the total estimated effort for the project. Looking at FIGURE 6-1, this information can be seen in the upper left corner of the window.

FIGURE 6-2 **Phase Distribution window displaying a sample project's overall phase distribution**



In addition, each phase of the project's development cycle is represented by the estimated effort, the estimated schedule and the estimated number of personnel needed for phase completion. Again looking at FIGURE 6-2, the information has been separated into columns. The first column displays the phase name. The second column displays the percentage that the corresponding phase takes in the estimated effort. The third column displays the estimated effort for each phase. The fourth column displays the percentage of the estimated schedule that is dedicated to the corresponding phase‘s

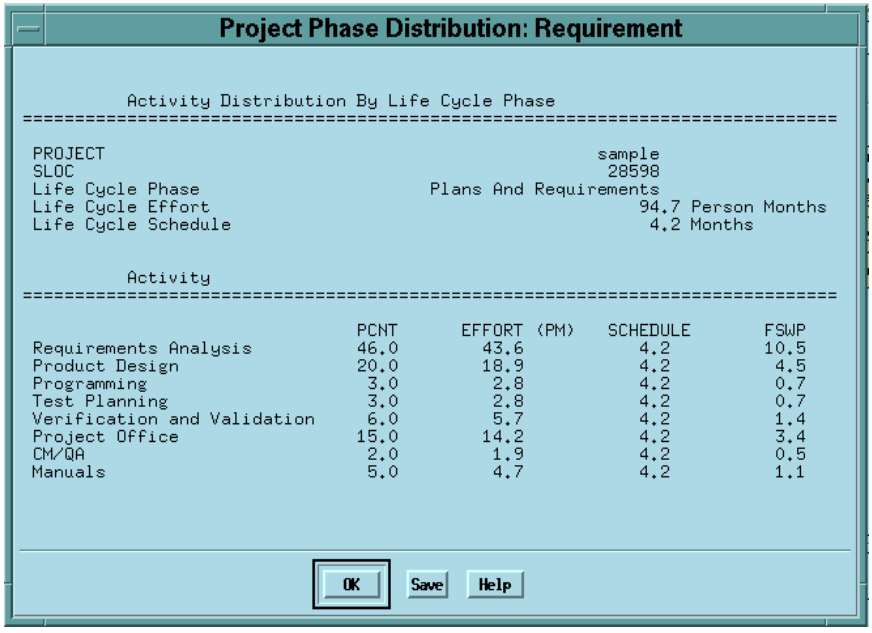
completion. The fifth column displays the estimated schedule for phase completion. And the last column displays the estimated number of personnel needed for phase completion (FSWP).

Note: The programming phase has been broken down into two additional phases: "Detailed Design" and "Code and Unit Test." The detailed design is a follow-up to the product design phase. In this sub phase, those points developed in the product design are elaborated to a point necessary to breakdown agreed functions into units necessary for coding. The code and unit test sub-phases house the actual coding effort of the individual units of code. The testing of these units (upon completion) is also encompassed within this sub phase.

6.1.2 Plans and Requirements Project Phase Distribution

The plans and requirements phase distribution allows the user to view the components of this particular phase. When the Plans and Requirements distribution is chosen from the Project Phase distribution menu, the window shown in FIGURE 6-3 is displayed. This window displays the following information: project name, the total project SLOC, the total estimated project effort, the total estimated project schedule. In addition the window displays the estimated effort for the activities of requirements analysis, product design, programming, test planning, verification & validation, project office, CM/QA, and manuals. These activity estimates are accompanied with a percentage of the phase effort that they encompass, the estimated effort, schedule and FSWP for the activities' completion as shown in FIGURE 6-3. To exit from this window click the OK button.

FIGURE 6-3 **Plans and Requirements Phase window for the overall project**

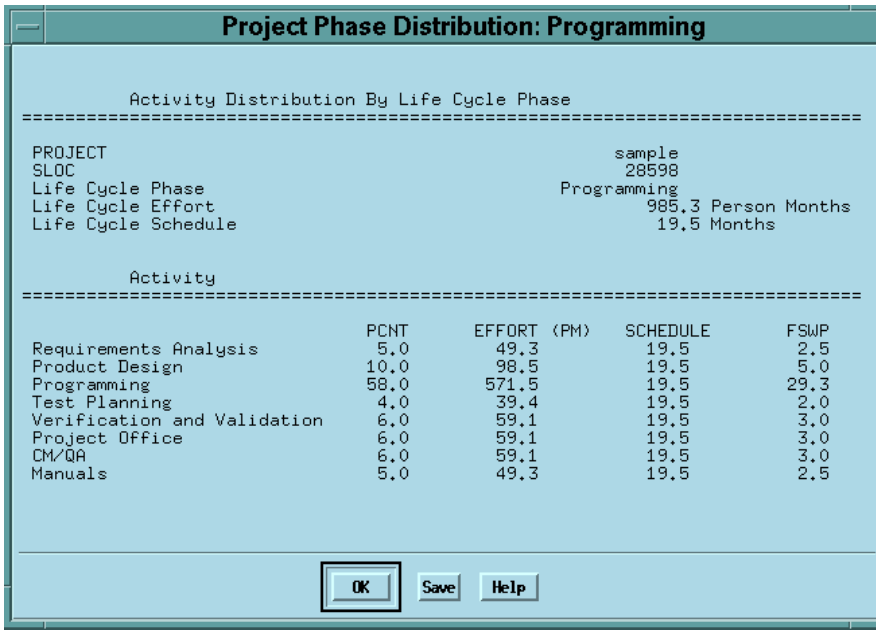


6.1.3 Programming Project Phase

The programming phase distribution allows the user to view the components of this particular phase. When the Programming distribution is chosen from the Project Phase distribution menu,

the window shown in FIGURE 6-4 is displayed. This window displays the following information: project name, the total project SLOC, the total estimated project effort, the total estimated project schedule. In addition the window displays the estimated effort for the activities of requirements analysis, product design, programming, test planning, verification & validation, project office, CM/QA, and manuals. These activities are accompanied with a percentage of the phase effort that they encompass, the estimated effort, schedule and FSWP for the activities' completion as shown in FIGURE 6-4. To exit from this window click the OK button.

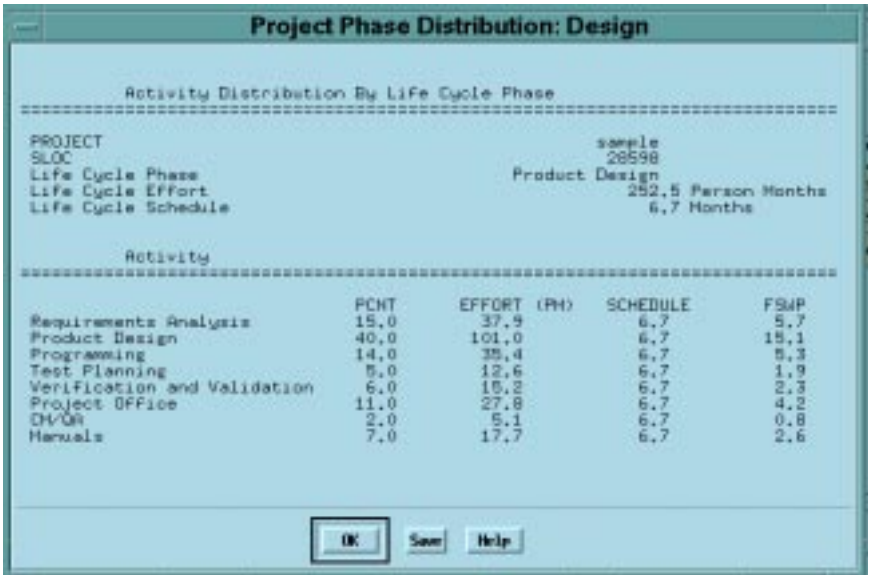
FIGURE 6-4 **Programming Phase window for the overall project**



6.1.4 Product Design Phase

The product design phase distribution allows the user to view the components of this particular phase. When the Product Design distribution is chosen from the Project Phase distribution menu, the window shown in FIGURE 6-5 is displayed. This window displays the following information: project name, the total project SLOC, the total estimated project effort, the total estimated project schedule. In addition the window displays the estimated effort for the activities of requirements analysis, product design, programming, test planning, verification & validation, project office, CM/QA, and manuals. These activity estimates are accompanied with a percentage of the phase effort that they encompass, the estimated effort, schedule and FSWP for the activities' completion as shown in FIGURE 6-5. To exit from this window click the OK button.

FIGURE 6-5 **Product Design window for the overall project**



6.1.5 Integration and Test Project Phase

The integration & test phase distribution allows the user to view the components of this particular phase. When the Integration and Test distribution is chosen from the Project Phase distribution menu, the window shown in FIGURE 6-6 is displayed. This window displays the following information: project name, the total project SLOC, the total estimated project effort, the total estimated project schedule. In addition the window displays the estimated effort for the activities of requirements analysis, product design, programming, test planning, verification & validation, project office, CM/QA, and manuals. These activity estimates are accompanied with a percentage of the phase effort that they encompass, the estimated effort, schedule and FSWP for the activities' completion as shown in FIGURE 6-6. To exit from this window click the OK button.

FIGURE 6-6 **Integration and Test window for the overall project**

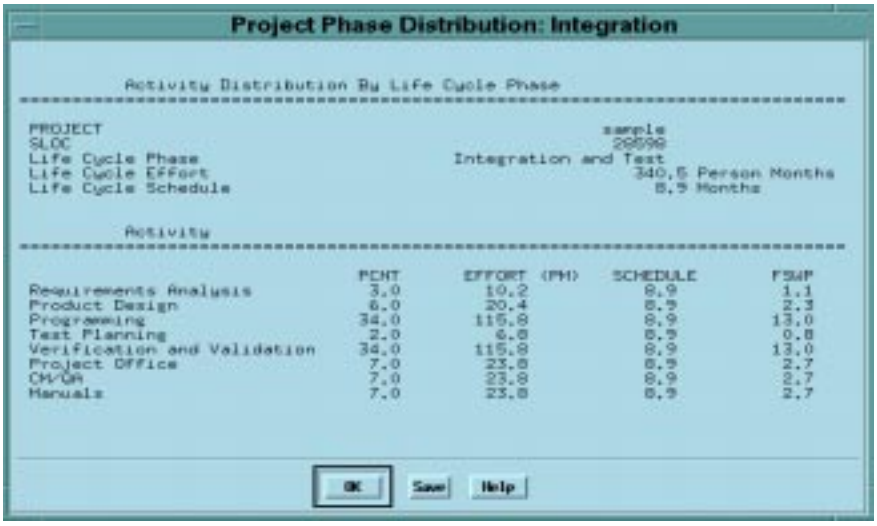
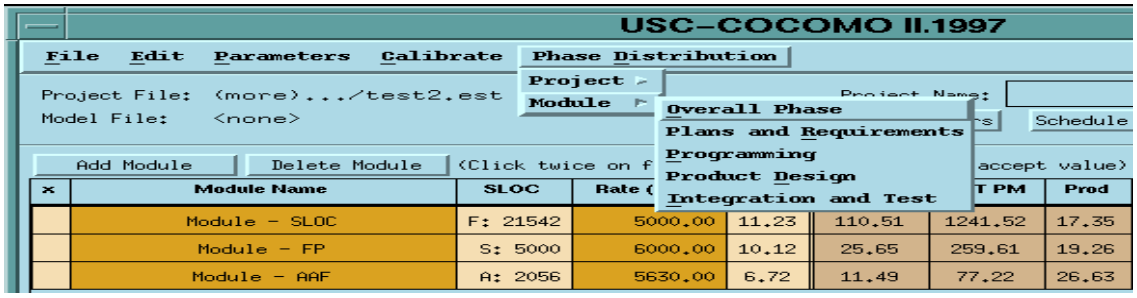


FIGURE 6-7 **Phase Distribution Module Sub-menu**



6.2 Module Phase Distribution

Four formats for viewing will appear in another menu: overall phase, plan & requirements, programming, and integration & test (see FIGURE 6-7). Each of these menu selections will be discussed in sections 6.2.1 - 6.2.4, respectively. The phase distribution of plan & requirements, programming and integration & test are broken down into activities. These activities include: requirements analysis, product design, programming, test planning, verification & validation, Module office, CM/QA, and manuals. For each of these activities, the percentage of the phase, the estimated effort, the estimated schedule, and the estimated FSWP is displayed. A description of each of these activities follows:

Requirements analysis: Determination, specification review and update of software functional, performance, interface, and verification requirements.

Product Design: Determination, specification, review and update of hardware-software architecture, program design, and database design.

Programming: Detailed design, code, unit test, and integration of individual computer program components. Includes programming personnel planning, tool acquisitions, database development, component level documentation, and intermediate level programming management.

Test Planning: Specification, review, and update of product test and acceptance test plans. Acquisition of associated test drivers, test tools, and test data.

Verification & Validation(V&V): Performance of independent requirements validation, design V&V, product test, and acceptance test. Acquisition of requirements and design V&V tools. "Are we building the product right?" and "are we building the right product?"

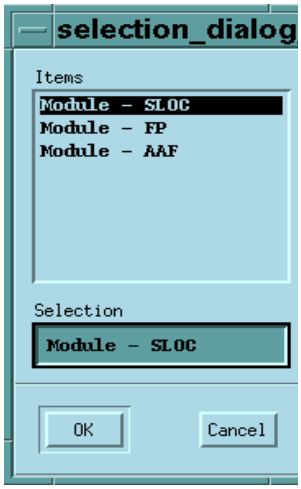
Module Office Functions: Module level management functions. Includes Module level planning and control, contract and subcontract management, and customer interface.

Configuration Management and Quality Assurance (CM/QA): Configuration management includes product identification, change control, status accounting, operation of program support library, development and monitoring of end item acceptance plan. Quality assurance includes development and monitoring of Module standards, and technical audits of software products and processes.

Manuals: Development and update of users' manuals, operators' manuals and maintenance manuals.

In order to view the phase distribution of an entire Module, the user can click on the Module Phase Distribution button under the Phase Distribution menu. When choosing any of the views of phase distribution, you will be confronted with a module selection window (see FIGURE 6-8). At this point, you may choose which module is to be viewed by clicking on the desired module name, which will be highlighted after the click. Click the OK button in order to initiate phase distribution of the chosen module.

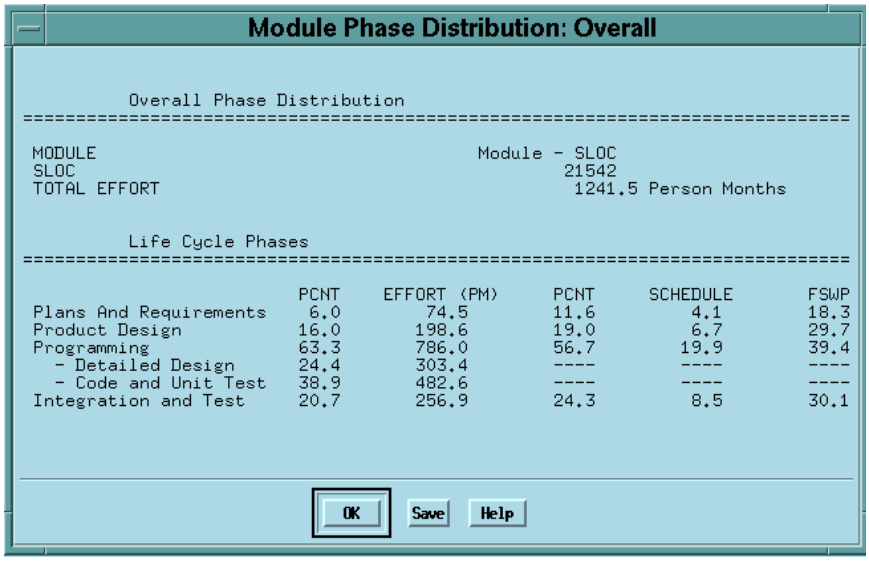
FIGURE 6-8 **Module selection window**



6.2.1 Overall Module Phase Distribution

The overall phase distribution allows the user to view an entire Module's estimated effort, schedule and number of personnel needed for phase completion. Upon clicking on "Overall Phase," a window will be displayed showing the phase breakdown four formats for viewing will appear in another menu: overall phase, plan & requirements, programming, and integration & test (see FIGURE 6-9). To exit from this window click the OK button.

FIGURE 6-9 **Phase Distribution window displaying a sample Module's overall phase distribution**



In addition, each phase of the Module's development cycle is represented by the estimated effort, the estimated schedule and the estimated number of personnel needed for phase completion. Again looking at FIGURE 6-9, the information has been separated into columns. The first column displays the phase name. The second column displays the percentage that the corresponding phase takes in the estimated effort. The third column displays the estimated effort for each phase. The fourth column displays the percentage of the estimated schedule that is dedicated to the corresponding phase‘s completion. The fifth column displays the estimated schedule for phase completion. And the last column displays the estimated number of personnel needed for phase completion (FSWP).

Note: The programming phase has been broken down into two additional phases: "Detailed Design" and "Code and Unit Test." The detailed design is a follow-up to the product design phase. In this sub phase, those points developed in the product design are elaborated to a point necessary to breakdown agreed functions into units necessary for coding. The code and unit test sub phase houses the actually coding effort of the individual units of code. The testing of these units (upon completion) is also encompassed within this sub phase.

6.2.2 Plans and Requirements Module Phase Distribution

The plans and requirements phase distribution allows the user to view the components of this particular phase. When the Plans and Requirements distribution is chosen from the Module Phase distribution menu, the window shown in FIGURE 6-10 is displayed. This window displays the following information: Module name, the total Module SLOC, the total estimated Module effort, the total estimated Module schedule. In addition the window displays the activities requirements analysis, product design, programming, test planning, verification & validation, Module office, CM/QA, and manuals. These activity estimates are accompanied with a percentage of the phase effort that they encompass, the estimated effort, schedule and FSWP for the activities' completion as shown in FIGURE 6-10. To exit from this window click the OK button.

6.2.3 Programming Module Phase

The programming phase distribution allows the user to view the components of this particular phase. When the Programming distribution is chosen from the Module Phase distribution menu, the window shown in FIGURE 6-11 is displayed. This window displays the following information: Module name, the total Module SLOC, the total estimated Module effort, the total estimated Module schedule. In addition the window displays the activities' requirements analysis, product design, programming, test planning, verification & validation, Module office, CM/QA, and manuals. These activity estimates are accompanied with a percentage of the phase effort that they encompass, the estimated effort, schedule and FSWP for the activities' completion as shown in FIGURE 6-11. To exit from this window click the OK button.

FIGURE 6-10 **Plans and Requirements Phase window for the overall Module**

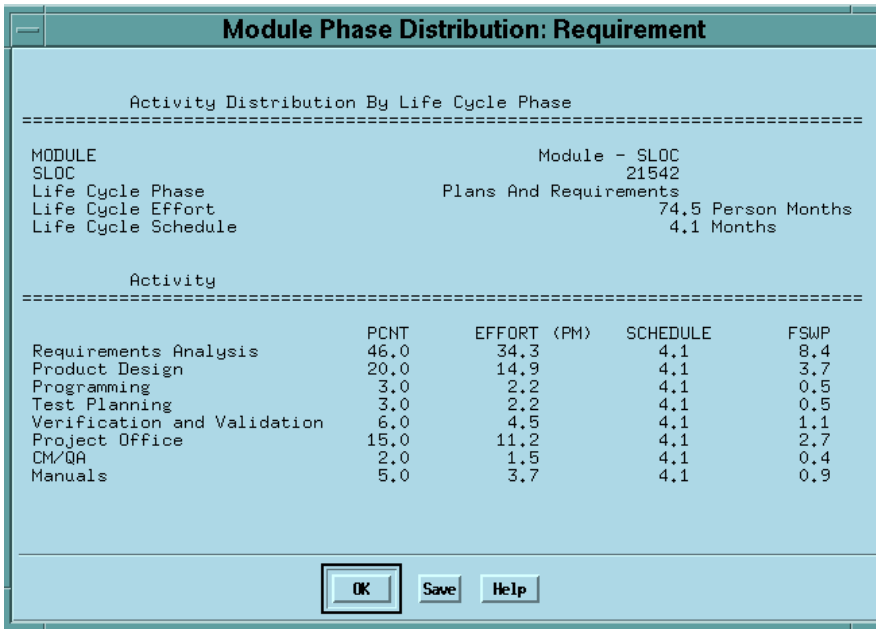
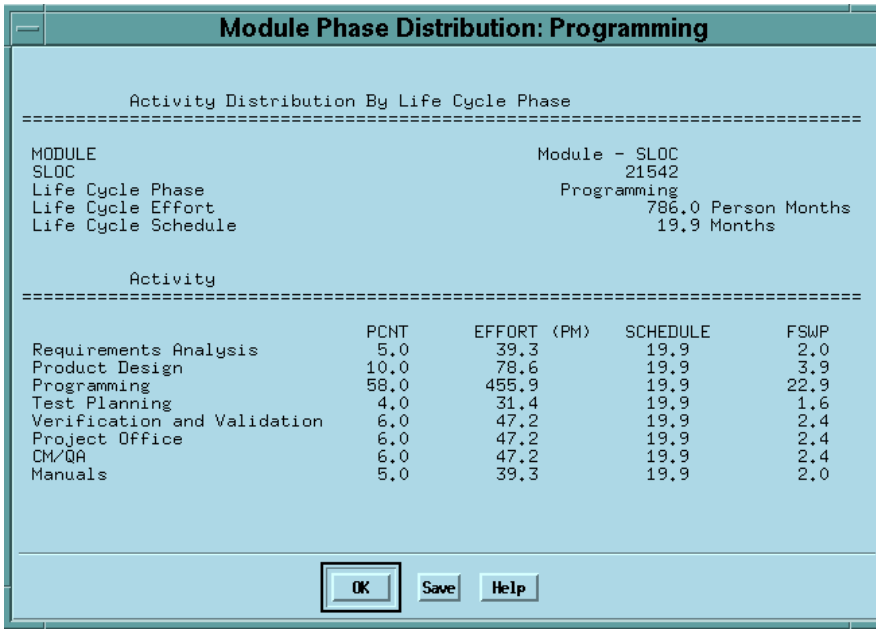


FIGURE 6-11 **Programming Phase window for the overall Module**



6.2.4 Product Design Phase

The product design phase distribution allows the user to view the components of this particular phase. When the Product Design distribution is chosen from the Module Phase distribution menu, the window shown in FIGURE 6-12 is displayed. This window displays the following information: Module name, the total Module SLOC, the total estimated Module effort, the total estimated Module schedule. In addition the window displays the activities requirements analysis, product design, programming, test planning, verification & validation, Module office, CM/QA, and manuals. These activity estimates are accompanied with a percentage of the phase effort that they encompass, the estimated effort, schedule and FSWP for the activities' completion as shown in FIGURE 6-12. To exit from this window click the OK button.

6.2.5 Integration and Test Module Phase

The integration & test phase distribution allows the user to view the components of this particular phase. When the Integration and Test distribution is chosen from the Module Phase distribution menu, the window shown in FIGURE 6-13 is displayed. This window displays the following information: Module name, the total Module SLOC, the total estimated Module effort, the total estimated Module schedule. In addition the window displays the activities requirements analysis, product design, programming, test planning, verification & validation, Module office, CM/QA, and manuals. These activity estimates are accompanied with a percentage of the phase effort that they encompass, the estimated effort, schedule and FSWP for the activities' completion as shown in FIGURE 6-13. To exit from this window click the OK button.

FIGURE 6-12 **Product Design window for the overall Module**

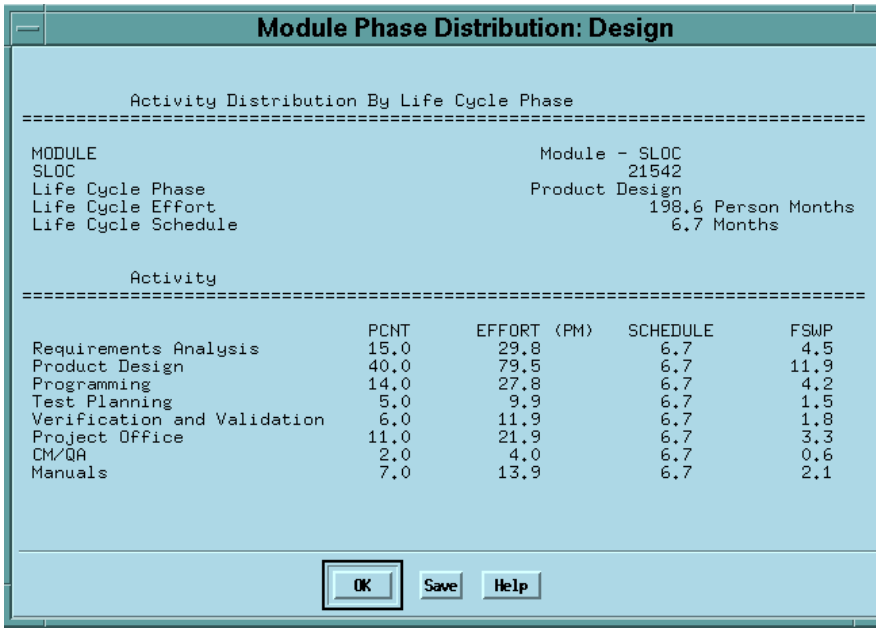
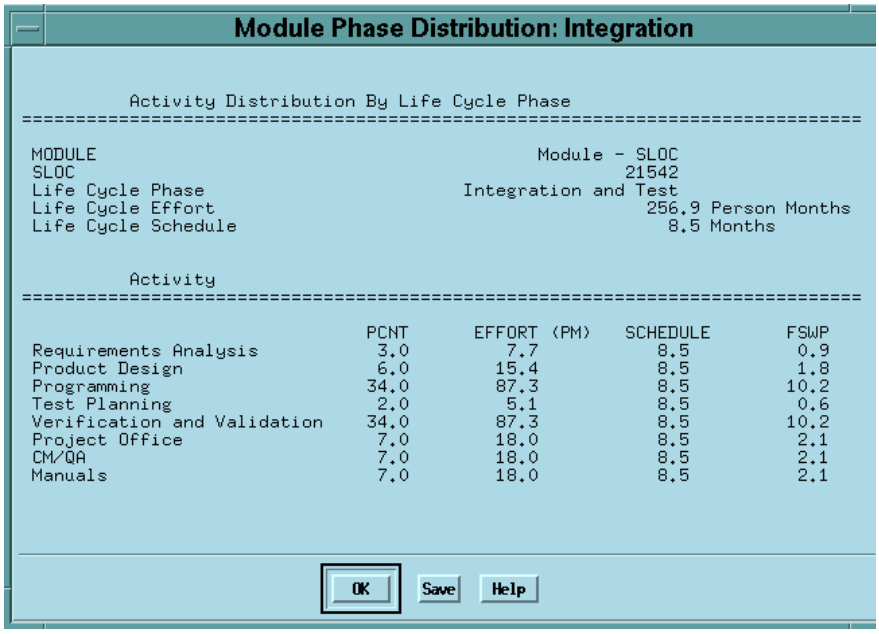


FIGURE 6-13 **Integration & Test window for the overall Module**



Chapter

7

References

Amadeus (1994), Amadeus Measurement System User's Guide, Version 2.3a, Amadeus Software Research, Inc., Irvine, California, July 1994.

Behrens, C. (1983), "Measuring the Productivity of Computer Systems Development Activities with Function Points," *IEEE Transactions on Software Engineering*, November 1983.

Boehm, B. (1981), *Software Engineering Economics*, Prentice Hall.

Boehm, B. and W. Royce (1989), "Ada COCOMO and the Ada Process Model," *Proceedings, Fifth COCOMO Users' Group Meeting*, Software Engineering Institute, Pittsburgh, PA, November 1989.

Boehm et al. (1995), "Cost Models for future Software Life Cycle Process: COCOMO 2.0", Annals of Software Engineering Special Volume on Software Process and Product Measurement, J.D Arther and S.M. Henry, Eds., J.C. Baltzer AG, Science Publishers, Amsterdam, The Netherlands, Vol 1, pp. 45 - 60.

Chidamber, S. and C. Kemerer (1994), "A Metrics Suite for Object Oriented Design,", *IEEE Transactions on Software Engineering*, (to appear 1994).

Goethert, W., E. Bailey, M. Busby (1992), "Software Effort and Schedule Measurement: A Framework for Counting Staff Hours and Reporting Schedule Information." CMU/SEI-92-TR-21, Software Engineering Institute, Pittsburgh, PA.

IEPUG (1994), *IFPUG Function Point Counting Practices: Manual Release 4.0*, International Function Point Users' Group, Westerville, OH.

Kunkler, J. (1985), "A Cooperative Industry Study on Software Development/Maintenance Productivity," Xerox Corporation, Xerox Square --- XRX2 52A, Rochester, NY 14644, Third Report, March 1985.

Madachy, J. Raymond (1997), "Heuristic Risk Assessment Using Cost Factors,", *IEEE Software*, May/June 1997, pp. 51-59.

Park R. (1992), "Software Size Measurement: A Framework for Counting Source Statements." CMU/SEI-92-TR-20, Software Engineering Institute, Pittsburgh, PA.

Selby, R., A. Porter, D. Schimidt and J. Berney (1991), "Metric-Driven Analysis and Feedback systems for Enabling Empirically Guided Software Development," Proceedings of the Thirteenth International Conference on Software Engineering (ICSE 13), Austin, TX, May 13-16, 1991, pp. 288-298.

Appendix

A

Accelerator Keys

	Sparc	Windows
File	F Meta+F	Alt+F
New	N Ctrl+N	N Ctrl+N
Load Project	L Ctrl+L	L Ctrl+L
Save Project	S Ctrl+S	S Ctrl+S
Save As Project	A Ctrl+A	A Ctrl+A
Load Model		O
Save Model		V
Save As Model		E
Make Report		R
Exit	X	X
View	No	Alt+V
Edit	E Meta+E	Alt+E
Add Module	No	A Ctrl+A
Clear All Modules	C	L
Snapshot	S	S
Undo	Ctrl+Z	U Ctrl+Z
Cut	Ctrl+X	T Ctrl+X
Copy	Ctrl+C	C Ctrl+C
Paste	Ctrl+V	P Ctrl+V

	Sparc	Windows
Parameters	P Meta+P	Alt+R
Product	P	P
Platform	L	L
Personnel	E	N
Project	J	J
User EAF	S	U
Scale Factor	C	S
Equation	E	E
Reset	R	R
Calibrate	C Meta+C	Alt+C
File Load	L	L
File Save	S	S
File Save As	A	A
Project	P	P
Compute	C	C
Phase	D Meta+P	Alt+P
Project		P
Overall Phase	O	O
Plans & Requirement	R	R
Programming	P	P
Product Design	D	D
Integration & Test	I	I
Module		M
Overall Phase	O	O
Plans & Requirement	R	R
Programming	P	P

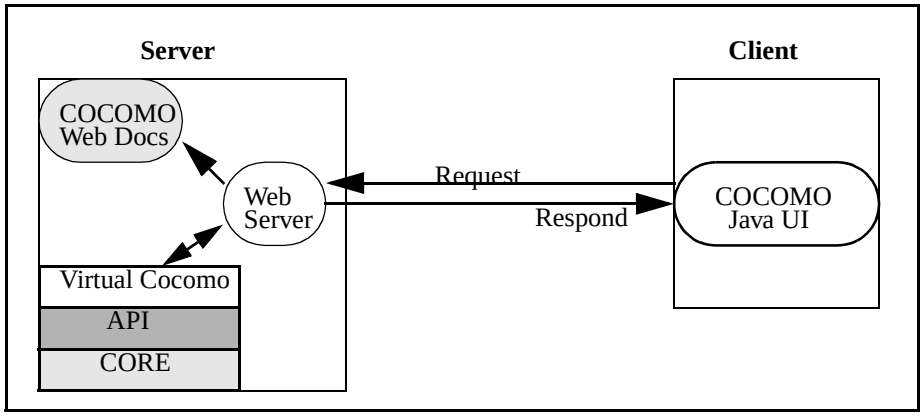
	Sparc	Windows
Product Design	D	D
Integration & Test	I	I
Help	H Meta+H	Alt+H
On Application	A	No
On Version	V	No
COCOMOII User's Manual	No	C
Using Help	No	U
About USC-COCOMOII	No	A

Appendix

B

Java COCOMOII Protocol

Java COCOMO has two components, a Java Applet user interface and a "virtual" COCOMOII as shown below. The former executes on the client side and the latter executes on the server side.



Once a user activates the virtual COCOMOII by accessing the cocomo home page, a TCP/IP connection is established. All commands are generated on the Client side according to the user's action. Here is an example of the sequence of protocol messages that are exchanged when the user:

1. Starts Java Cocomo

2. Adds a new module

3. Sets its SLOC to 10000

4. Sets the RELY EAF to High

5. Saves the result

1. Starts Java Cocomo

When a user visits Java COCOMOII home page, all Java classes are loaded from the server to the client. Then the client side sends the virtual server name, the client's IP and port number for the exchange of commands and responses between the server and client.

../cgi-bin/run-cocomo-server1?128.125.3.9+6211

../cgi-bin/run-cocomo-server1 starts the virtual COCOMOII server. 128.125.3.9 is the client's IP address. 6211 is the port number for the client.

Once a TCP/IP connection is established, the client side generates the following commands for the initialization and sends them to the server one by one.

1. COCOMO_DO_INIT

Command: 101!#%0@$^

101 is the numerical equivalent of COCOMO_DO_INIT. The 3 characters !#% act as a CD (COCOMO_DELIMINATER). 0 is the number of arguments. The last 3 characters @$^ specify the end of a command. After the server receives this command, the server executes the initialization process then sends a response to the client.

All commands consist of three parts. The first part specifies the numerical equivalent of a command. The second part contains the number of arguments. If the number of arguments is not zero, the third part contains the list of arguments.

Response: 0!#%0@$^

The first 0 represents the error status of the command (0: Success, 1: General Error, 2: Format Error). The second 0 is the number of arguments. All responses have the same format. If the number of arguments is not zero, the list of arguments is shown after the second part.

2. COCOMO_GET_MOD_EDSI_FP_LANG

Command: 327!#%0@$^

Response:

0!#%28!#%ADA!#%AI_SHELL!#%APL!#%ASM_BASIC!#%ASM_MACRO!#%BASIC_ANSI
!#%BASIC_COMP!#%BASIC_INTR!#%C!#%COBOL_ANSI_85!#%FIRST_GENERATION!#
%FORTH!#%FORTRAN_77!#%FOURTH_GENERATION!#%FIFTH_GENERATION!#%HIG
H_LEVEL!#%LISP!#%MODULA_2!#%OBJECT_ORIENTED!#%PASCAL!#%PROCEDUR-
AL!#%PROGRAM_GENERATOR!#%PRO-
LOG!#%QUERY_LANGUAGE!#%REPORT_GENERATOR!#%SECOND_GENERATION!#%S
PREADSHEET!#%THIRD_GENERATION@$^

This response contains the error status, number of arguments, and list of arguments. It has 28 arguments which are the list of languages used by EDSI input in the Function Points.

3. COCOMO_GET_CAL_PRJ_NUM

Command: 308!#%0@$^

This command is used to get the number of available projects used by the projects menu on Calibrate pulldown menu.

Response: 0!#%1!#%0@$^

As explained above, the first 0 is the error status, which means that the requested process is successfully done on the server side. 1 is the number of arguments. 0 is the number of projects for the calibration of the equation parameters.

4. COCOMO_GET_PRJ_NAME

Command: 303!#%0@$^

This command is sent to get the project's name displayed on the CLEF.

Response: 0!#%1!#%<none>@$^

The string <none> is the default name of a new project.

5. COCOMO_GET_PROJECT_TOTALS

Command: 333!#%0@$^

Response:

0!#%24!#%0!#%0.00@$^

This response contains total optimistic, most likely, and pessimistic PM, Schedule, Productivity, Cost, Instruction cost, FSWP, and Risk which are to be displayed on the total area of the CLEF.

2. Add a New Module

1. COCOMO_DO_ADD_MOD

Command: 119!#%0@$^

This command requests that the server add a new module to the module linked list of the project.

Response: 0!#%0@$^

2. COCOMO_GET_MOD_LINE

Command: 322!#%1!#%0@$^

This command contains the module index which is used to search the specified module (Modules are indexed as 0, 1, 2, ...).

Response:

0!#%11!#%<none>!#%S:0!#%0.00!#%1.00!#%0.00!#%0.00!#%0.00!#%0.00!#%0.00!#%0.00!#%0.00@$^

This response contains 11 arguments. They are Module name, ESDI, Rate, EAF, NOM PM, EST PM, Productivity, Cost, Instruction Cost, FSWP and Total Risk level for the specified module.

3. Set the SLOC of a Module to 10000

1. COCOMO_GET_MOD_EDSI_BRAK

Command: 324!#%1!#%0@$^

This command is used to get the Breakage of the SLOC for the specified module. 0 is the index of the module.

Respond: 0!#%1!#%0.00@$^

After the server gets the paramter of the Breakage for the specified module, this respond message is sent to the client. 0.00 is the Breakage for the SLOC of the module.

2. COCOMO_GET_MOD_EDSI_FLAG

Command: 325!#%1!#%0@$^

As described in Chapter 1, there are three ways of the input for EDSI (Source Lines of Code, Function Points, and Adaptation Ajustment Factor). This command gets the flag to set the EDSI screen.

Response: 0!#%1!#%1@$^

The second 1 is a flag for EDSI. the flag is set to one of the following options.

1: Source Lines of Code

2: Function Points

3: Adaptation Adjustment Factor

3. COCOMO_GET_MOD_EDSI_SLOC

Command: 326!#%1!#%0@$^

0 is the module index

Respond: 0!#%1!#%0@$^

This response has one argument which is the SLOC for the module(0).

After the client and sever exchange the above messages, the EDSI dialog appears. Java COCOMOII waits until the user inputs the values of Breakage and SLOC. When Ok button is clicked after he or she inputs the values, the following commands and reponses are exchanged.

4. COCOMO_SET_MOD_EDSI_FLAG

Command: 217!#%2!#%0!#%1@$^

This command sets the EDSI flag to one of SLOC, FP, and Adaptation. It contains two arguments, module index(0) and EDSI flag(1).

Response: 0!#%0@$^

5. COCOMO_SET_MOD_EDSI_BRAK

Command: 216!#%2!#%0!#%0.00@$^

This command requests that the server set the Breakege of the specified module(0) to 0.00.

Response: 0!#%0@$^

6. COCOMO_SET_MOD_EDSI_SLOC

Command: 218!#%2!#%0!#%10000@$^

This command requests that the server set the SLOC of the specified module(0) to 10000.

Response: 0!#%0@$^

7. COCOMO_GET_MOD_LINE

Command: 322!#%1!#%0@$^

Response:

0!#%11!#%<none>!#%S:10000!#%0.00!#%1.00!#%34.85!#%34.85!#%286.97!#%0.00!#%0.00!#%3.67!#%0.00@$^

These command and response are described above.

8. COCOMO_GET_PROJECT_TOTALS

Command: 333!#%0@$^

Response:

0!#%24!#%10000!#%34.85!#%286.97!#%34.85!#%286.97!#%0.00!#%0.00!#%3.67!#%0.00!#%9.50!#%27.88!#%229.57!#%0.00!#%0.00!#%3.18!#%0.00!#%8.77!#%43.56!#%358.71!#%0.00!#%0.00!#%4.23!#%0.00!#%10.30@$^

These command and response are described above.

4. Set the RELY EAF to High

1. COCOMO_GET_MOD_EAF

Command: 331!#%1!#%0@$^

Response:

0!#%38!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0@$^

This reponse contains 38 arguments. The first 19 arguments are EAF multipliers. They are set to one of these values (0: Very Low, 1: Low, 2: Nominal, 3: High, 4: Very High). The rest 19 arguments are interpolation rate. They are set to one of these values (0: 0%, 1: 25%, 2: 50%, 3: 75%). Those arguments are displayed on the EAF dialog.

2. COCOMO_DO_CALC_EAF

Command:

126!#%38!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0@$^

This command requests that the server calculate the current EAF value. This command is generated whenever any button is pressed on the EAF dialog.

Response: 0!#%1!#%1.00@$^

3. COCOMO_DO_CALC_EAF

Command:

126!#%38!#%3!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0@$^

After RELY button is clicked, this command is sent to have the server calcurate the current EAF. The first argument is set to Very High.

Response: 0!#%1!#%1.15@$^

This response contains the calcurated current EAF value for the module.

When OK button is clicked after RELY EAF value is changed, the following commands are executed sequetially.

4. COCOMO_SET_MOD_EAF

Command:

222!#%39!#%0!#%3!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%2!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0!#%0@$^

All EAF mutipliers and interpolation rates are sent to the server so that it set the current values on EAF dialog for the module which is specified in the first argument(0).

Response: 0!#%0@$^

5. COCOMO_DO_CAL_RISK

Command: 130!#%1!#%0@$^

This command requests that the server calcurate the risk level for the module which is displayed on the last column of the module line on the CLEF.

Response: 0!#%2!#%0!#%

6. COCOMO_GET_MOD_LINE

Command: 322!#%1!#%0@$^

Response:

0!#%11!#%<none>!#%S:10000!#%0.00!#%1.15!#%34.85!#%40.07!#%249.54!#%0.00!#%0.00!#%4.01!#%0.00@$^

These command and response are described above.

7. COCOMO_GET_PROJECT_TOTALS

Command: 333!#%0@$^

Response:

0!#%24!#%10000!#%34.85!#%286.97!#%40.07!#%249.54!#%0.00!#%0.00!#%4.01!#%0.00!#%9.99!#%32.06!#%199.63!#%0.00!#%0.00!#%3.48!#%0.00!#%9.22!#%50.09!#%311.92!#%0.00!#%0.00!#%4.63!#%0.00!#%10.82@$^

These command and response are described above.

5. Save the Result

When the user clicks the Save button on File pulldown menu, the Applet requests that the user enter his/her user_id and passwd. If he/she does not have an account, the server creates a new acccount by user's clicking New button. Otherwise, the server sends available project file names in the usr's account.

1. COCOMO_DO_CREATE_USR_ACCT

Command: 102!#%2!#%Test!#%test@$^

This command requests that the server create a new account for the user. the user_id(Test) and passwd(test) is saved in the file(pawdfile).

Response: 0!#%0@$^

2. COCOMO_SET_PRJ_NAME

Command: 201!#%1!#%<none>@$^

<none> is the default project name. This command requests that the sever set the project name as the default.

Response: 0!#%0@$^

3. COCOMO_SET_MOD_NAME

Command: 215!#%2!#%0!#%<none>@$^

This command requests that the server set all module's names. If the module name is not changed, the defualt name, <none> is sent.

Response: 0!#%0@$^

4. COCOMO_SET_MOD_RATE

Command: 221!#%2!#%0!#%0.00@$^

This command requests that the server set each module's rate.

Response: 0!#%0@$^

5. COCOMO_GET_PROJECT_TOTALS

Command: 333!#%0@$^

Response:

0!#%24!#%10000!#%34.85!#%286.97!#%40.07!#%249.54!#%0.00!#%0.00!#%4.01!#%0.00!#%9.99!#%32.06!#%199.63!#%0.00!#%0.00!#%3.48!#%0.00!#%9.22!#%50.09!#%311.92!#%0.00!#%0.00!#%4.63!#%0.00!#%10.82@$^

These command and response are described above.

6. COCOMO_GET_PRJ_AVAILABLE

Command: 301!#%0@$^

This command requests that the server send the available project files' names in the account.

Response: 0!#%0@$^

When OK button is clicked after the user inputs the name of the project file, the client sends this command.

7. COCOMO_SAVE_AS_PROJECT

Command: 106!#%1!#%test@$^

This command request that the server save the project in the specified file(test.est).

Response: 0!#%0@$^

Below is the complete set of protocol commands used by virtual COCOMOII. Extensions to Java COCOMOII are possible by using this protocol.

1. Request Message

Request =

Request-Header CD Request-Argument CEL

Request-Header =

(DO | SET | GET) CD #_of_Args

Request-Argument =

do-args | get-args | set-args

do-args =

INIT | NEW | COMPUTE | ACCEPT | RESET | ADD

| CREATE CD user_name CD passwd

| CHECK CD user_name CD passwd

| LOAD CD <proj_name|model_name|calib_name>

| SAVE CD <proj_name|model_name|calib|name>

| DELETE CD <mod_index|proj_index>

| CALCULATE (CD <arg>)*

| (COPY|PASTE) CD mod_index

set-args =

PROJ_NAME CD proj_name

| PROJ_EFFORT CD proj_index CD proj_effort

| PROJ_SCHEDULE CD proj_index CD proj_schedule

| SCALE_FACTOR (CD <arg>)*

| SCED CD sced_val CD inter_sced

|(PRODUCT|PLATFORM|PERSONNEL|PROJECT|UDF)(CD<dirver_value> CD <inter_value>)*

| SCALEFACTOR (CD <dirver_value> CD <inter_value>)*

| EQUATION CD effort-entry-value CD schedule-entry-value

| MOD_LINE CD mod_index (CD <arg>)*

| MOD_EDSI_BRAK CD mod_index CD brak

| MOD_EDSI_FLAG CD mod_index CD flag

| MOD_EDSI_FP (CD <arg>)*

| MOD_RATE CD mod_index CD rate

| MOD_EAF (CD <arg>)*

get-args =

PROJECTS | PRJ_MOD_NUM | MODELS | CALIBRATIONS | SCALEFACTOR

| EQUATION |PROJ_REPORT | PROJ_TOTAL | HELPS

| RISK_LEVEL CD mod_index

| (MOD_LINE | MOD_EAF | MOD_EDSI_FLAG) CD mod_index

| (MOD_FP_LANG | MOD_EDSI_AAF | MOD_EDSI_FP) CD mod_index

| PROJ_LINE CD proj_index

|(MOD_OVERALL| MOD_PLAN_RPT|MOD_DESIGN_RPT|MOD_INTE_RPT) CD mod_index

|(PRJ_OVERALL|PRJ_PLAN_RPT |PRJ_DESIGN_RPT |PRJ_INTE_RPT) CD proj_index

| (CAL_NEME| CAL_DATE| CAL_EFF| CAL_SCED) CD proj_index

2. Response Message

Response =

Response-Header Response-Argument

Response-Header =

Error_Status CD No_of_Args

Error_Stauts =

0 | 1 | 2

0: Success

1: General Error

2: Format Error

Response-Argument =

PROJ_NAME| MOD_LINE| PROJ_TOTAL| MOD_EAF| MOD_EDSI_FLAG| MOD_FP_LANG

| MOD_RISK| EQUATION| CAL_NAME| CAL_PROJ_LINE| DEL_MOD| DEL_PROJ

| PROJECTS | MODELS | CALIBRATIONS | OPEN_PROJ| PARAMETERS

3. Miscellaneous

CD(COCOMO_DELIMINATER) = "!#%"

CEL(COCOMO_END_LINE) = "@$^"