A REQUIREMENTS SPECIFICATION SOFTWARE COST ESTIMATION TOOL

by

GARY DAVID SCHNEIDER

B.S. Long Island University, 1974
M.B.A. Long Island University, 1975

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

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1986

Approved by:

[Signature]
Dr. D. A. Gustafson
## CONTENTS

1. Chapter One - Introduction .................................................. 1
   1.1 Overview ........................................................................ 1
   1.2 The Software Life Cycle .................................................. 1
   1.3 The Requirements Specification ......................................... 2
   1.4 Software Cost Estimation .................................................. 3
   1.5 Software Cost Models Used ............................................... 8
   1.6 Scope of the Implementation .............................................. 13

2. Chapter Two - Requirements for Implementation .................. 15
   2.1 Overview ........................................................................ 15
   2.2 Application of Boehm’s Constructive Cost Model ................. 16
   2.3 Application of Putnam’s System Sizing Technique ................. 20
   2.4 Application of DeMarco’s Specification Measure ................. 20
   2.5 Application of Britcher and Gaffney’s State Machine Measure ................................................................. 22
   2.6 Design Consideration Impacted By Requirements ............... 23

3. Chapter Three - Design of Cost Estimation ...................... 24
   3.1 Overview ........................................................................ 24
   3.2 Obtaining Interactive Input .............................................. 24
   3.3 Scanning the Specification Input File ................................... 25
   3.4 Calculation of the Cost Estimation ..................................... 25
   3.5 Generating the Report ...................................................... 27
   3.6 Archiving the Data ......................................................... 27

4. Chapter Four - Implementation of Cost Estimation ............ 35
   4.1 Overview ........................................................................ 35
   4.2 Invocation and Interactive Data Entry ................................. 35
   4.3 Processing the E-R-A Specification File ............................... 37
   4.4 Cost Computation and Report ........................................... 37
   4.5 The Cost Estimation Database .......................................... 39
   4.6 Evaluating The Cost Estimation Results .............................. 40

5. Chapter Five - Conclusions and Extensions ................... 43
   5.1 Extensions ....................................................................... 43

APPENDIX I - BNF SYNTAX FOR E-R-A SPECIFICATION ........ 45
APPENDIX II - SCE E-R-A SPECIFICATION ............................. 47
APPENDIX III - SCE KEY DATABASE SEGMENT LAYOUTS ........ 66
APPENDIX IV - SCE COMPUTATIONAL TABLES ......................... 70
APPENDIX V - SCE SOURCE CODE ......................................... 75
REFERENCES ....................................................................... 103
LIST OF FIGURES

Figure 3-1. The Software Cost Estimator Procedure Flow........ 29
Figure 3-2. The Software Cost Estimator Measure Flow........ 30
Figure 3-3. The Software Cost Estimator Front End........... 31
Figure 3-4A. The Software Cost Estimator Report Layout (Page 1)..................................................... 32
Figure 3-4B. The Software Cost Estimator Report Layout (Page 2).......................................................... 33
Figure 3-5. The Software Cost Estimator Database
Structure......................................................... 34
Figure 4-1. Cost Estimation Results........................................ 42
Figure III-1. CoCoMo Cost Driver Database Segment Layout..... 66
Figure III-2. CoCoMo Basic/Intermediate Database Segment
Layout.............................................................. 68
Figure III-4. General Project Database Segment Layout........... 69
Figure IV-1. Britcher & Gaffney Software Sizing Measure
Table............................................................... 70
Figure IV-2. CoCoMo Development Effort Table...................... 71
Figure IV-3. CoCoMo Multiplier & Exponent Table................ 72
Figure IV-4. Bang Size Correction Table............................. 73
Figure IV-5. Bang Functional Complexity Table...................... 74
1. Chapter One - Introduction

1.1 Overview

This report describes the application of a cost estimation tool based upon the software requirements specification. The SCE tool applies the methodologies of Boehm[1], DeMarco[2], and Britcher & Gaffney[3] to calculate a cost estimate from a requirements specification. The objective is to be able to utilize, during the early phases of the software life cycle, the requirements specification as an accurate and reliable predictor of development effort.

1.2 The Software Life Cycle

The model for the software development lifecycle is composed of seven sequential phases. These are Requirements Analysis, Preliminary Design, Detailed Design, Implementation, System Integration and Testing, Acceptance Testing, and Maintenance & Operation. It is the Requirements Analysis phase that will be of significant consideration for this paper.

Traditionally, these phases represent distinct differences of activity. However, McGarry[4] at NASA's Software Engineering Laboratory, claims that activities characteristically performed in one phase can be performed in other phases as well. For example the refinement of requirements analysis which represent the bulk of activity at the beginning of the project, will continue at a lower level throughout the development life cycle. This life cycle phase overlap indicates that activities are continuously evolving and
that the requirements specification is a dynamic instrument. The
distinction is of importance for cost estimation and illustrates
how estimates may need to be recomputed many times during the
development cycle.

1.3 The Requirements Specification

The software development process originates with a perceived
"problem" and a series of abstract solutions. These solutions are
defined during the requirements phase of the software life cycle
and play an important role for the remaining development phases.
Often requirements are neglected by management during project
development for fear of incurring costs and delays. The
requirements specification serves as a vehicle for communication
and refinement. This is accomplished with the client and developer
"iterating" requirements until a complete and precise requirements
document is produced. The requirements specification becomes an
early indicator or predictor of project development. Some
additional goals of the requirements specifications include being
understandable, modifiable, precise, unambiguous, internally
consistent, and complete.

Numerous requirements specification procedures have been developed,
but it is the Entity-Relationship-Attribute form that will be used
for this tool. The E-R-A form is an information flow approach that
describes the process components through a series of "entities".
The entities establish a relationship to other entities and are
described by "attributes". The attributes supply the
characteristics and domain information for each entity. Examples
of entity descriptors include 'inputs', 'outputs', 'activities', and 'type'. The E-R-A form describes a machine-processable specification whose language describes objects that relate to objects in later phases of the software life cycle[4]. The style borrows heavily from Heninger's[5] AE7E Aircraft model for a disciplined approach used to document software requirements. The model forms a foundation for describing a system in terms of external stimuli and externally visible behavior.

1.4 Software Cost Estimation

As projects become increasingly larger and more complex, cost estimation becomes more important in planning schedules and resource allocations. It is clear that the cost of software development has increased to the point that software represents the largest component of total system cost. Being able to accurately predict resource requirements would be a tremendous aid for management.

Historically, cost estimation involves two distinct steps that are usually represented in a single model. First, estimating the amount of work to be done and second, estimating the amount of effort needed to do the work. Then from these estimates a schedule is developed for performing the actual work.

1.4.1 Cost Estimation Techniques  Boeing[1] describes numerous software cost estimation techniques in use today. They generally fall into an algorithmic or non-algorithmic groupings which are described in detail below:
1.4.1.1 Algorithmic Cost Models The most common quantitative form of measurement determines its result from one or more algorithms combined with a series of supporting cost drivers. They are categorized as being linear, multiplicative, analytic, tabular, and composite.

- Linear Models - Originated during the mid-60's, a linear cost estimation model takes the general form of:

\[
\text{Effort} = a_0 + a_1 x_1 + \ldots + a_n x_n
\]

where
\[
\begin{align*}
    x_1, \ldots, x_n & \quad \text{are the cost driver variables} \\
    a_0, \ldots, a_n & \quad \text{are the set of coefficients that correlate to a set of observed data points}
\end{align*}
\]

The resulting effort value is then multiplied by a constant labor cost factor. The SDC[7] model using a significant sample of data points and cost drivers is an example of this approach. The limitation of the linear model is not being able to account for the large number of non-linear interactions that normally takes place during development.

- Multiplicative Models - Similar to the linear form, the multiplicative cost estimate takes the form of:

\[
\text{Effort} = a_0 \cdot a_1 \cdot a_2 \cdot \ldots \cdot a_n
\]

where
\[
\begin{align*}
    x_1, \ldots, x_n & \quad \text{are the cost driver variables} \\
    a_0, \ldots, a_n & \quad \text{are the set of coefficients that correlate to a set of observed data points}
\end{align*}
\]

IBM's Walston & Felix[8] and Doty Associates[9] are the most...
noted models of this type. There stability lies in making sure that the cost driver variables are independently chosen. However, the models support only a limited number of cost driver multipliers (usually 0 or 1) that produce estimates that change only in large steps.

- Analytic Models - Using the general mathematical approach, the analytic model takes the form of:

\[ \text{Effort} = f(x_1, \ldots, x_n) \]

where
\[ x_1, \ldots, x_n \text{ are cost driver variables} \]
\[ f \text{ is a non-linear or multiplicative mathematical function} \]

Putnam's[10] use of the Raleigh distribution for project personnel level versus time is an example of the analytic approach. The model is criticized for the limited number of cost variables that are defined and may generate unstable results.

- Tabular Models - These models utilize tables to equate cost driver variables to software development effort. Thus the stability of the models are dependent on the number of cost driver or elements used in the table. Examples of tabular models are Aron[11], Wolverten[12], and Boeing[13].

- Composite Models - These incorporate a combination of linear, multiplicative, analytic, and tabular functions to estimate software effort as a function of cost driver variables. The RCA PRICE-S[14] model, and the Putnam[15] SLIM model are examples of commercially available products that are of
composite design. The CoCoMo\(^1\) model used for this tool and
described in the next section is also of composite design, and
unlike the others, is non-proprietary.

1.4.1.2 Non-Algorithmic Cost Models

- **Analogy** - Reasoning by analogy with one or more completed
  projects. The similarities and differences between the new
  project and the completed project are used to estimate the new
  project cost. The advantage of estimation by analogy is that
  the estimate is based on experiences of past projects. These
  experiences can be evaluated for use in developing future
  projects. The disadvantages of analogy is the difficulty in
  assessing how representative the previous project is in
  relation to the new project.

- **Expert Judgement** - This method involves consulting one or more
  experts and obtaining an expert consensus. Here the
  experiences and judgements are used to distinguish the
  differences between past and future projects. The expert can
  process factors that may enhance or inhibit development costs.
  The problems in using expert judgement hinge on objectivity
  and in being able to apply the judgement from week to week as
  specifications change.

- **Top-Down** - This provides an overall estimate for the project
  by evaluating the global properties of the software product.
  Then the estimate is split up among the various sub
  components. The advantage of using top-down is being able to
  focus on development cost at the system level. This ensures
that functions such as quality assurance, integration of the product, and configuration management are not overlooked. The major disadvantage of top-down estimation is that it does not identify low level problems that could increase costs and may overlook components that need to be developed.

- **Bottom-up** - Each component of the project is computed separately, then the results are totaled to produce an overall estimate for the job. The strengths and weaknesses of bottom-up estimating are complimentary to top-down estimating. Here the bottom-up estimate will identify costs in developing individual components and will often overlook many of the system level costs. While the bottom-up estimate involves more effort, it does help cost out the system to the person(s) responsible for each component.

- **Software Size Estimation** - A means of determining the size of the software product early in the life cycle using design or requirements specification elements. DeMarco[1] and Britcher & Gaffney[4] are examples of this technique and are discussed in the next section.

The conclusion in classifying cost estimation techniques is that each method has its own strengths and weaknesses for determining project development costs. In addition, the particular strengths and weaknesses are usually complimentary so that the best method of estimation can be expressed as a combination of techniques[4].
1.4.1.3 Issues Most cost estimation activities are usually performed during design and implementation phases which is long after resources and commitments have been made for development. Being able to produce accurate cost estimates during the requirements stage of software development can add valuable insight for determining product feasibility and resource allocation. Correlating these results to traditional costing techniques can allow algorithmic cost models such as CoCoMo to be applied early in development.

Each of these various methods share certain similarities in that they require some historical knowledge of other software projects. This is supported by Mohanty[17] who compared several cost model equations using data for a hypothetical software system. The wide variation in cost estimates was suggested as environmental. He concluded that any software cost estimation technique needs to be calibrated to the particular environment being used. An historical database, therefore, serves a necessary element for recording software cost and project information.

1.5 Software Cost Models Used

This section will focus on specific cost estimation techniques selected for the implementation of this tool. The discussion includes techniques by DeMarco, Boehm, Putnam, and Britcher & Gaffney.

1.5.1 SCE and Requirements Specifications Boehm's[18] paper describes a fundamental limitation of software cost estimation techniques and the limitation in which estimates can be made. It
is exhibited by the accuracy or uncertainty in which software cost estimates can be made expressed as a function of the software life cycle. The level of uncertainty is greatest at the beginning of the software life cycle and progressively decreases during later phases. This decrease in uncertainty is natural as fundamental design issues become resolved. However, it is at the earliest stages of project development, prior to any actual expenditure or commitment of resources, that the need to obtain reasonable cost estimates are most critical.

Therefore, to facilitate accurate and timely software cost estimation, it is necessary to identify the requirements to be satisfied and the software to be developed as early as possible and in a more quantitative fashion. Usually, cost estimation methodologies use source lines of code as a measure of product size. Clearly, it is desirable to focus on information available in the early, conceptual stages of the software life cycle. Using quantitative measures of requirements as input to the SCE process, a measurement of value can be identified for both the user and developer. This value can be extended to establish a relationship between specification decomposition and product size in source instructions[19]

1.5.2 The CoCoMo Model CoCoMo (CONstructive COst Modeling)\(^1\) was developed by Barry Boehm of TRW, Inc. and is based on the algorithmic relationship between product size and production effort. The results are further refined by the application of a number of cost drivers. CoCoMo was developed as a hierarchical series of three models and built under the experiences of TRW
models. The models are called basic, intermediate and detailed CoCoMo.

Basic CoCoMo was intended to provide a quick, rough order estimate useful during first cut exercises.

Intermediate CoCoMo takes into account additional factors relating to the particular project. These personnel and environment adjustment factors allow costing to take place at the module level and will increase the accuracy of the estimate. The intermediate CoCoMo becomes more important as the project details become further refined.

Detailed CoCoMo is the most finely-tuned version of the model. It is designed to allow the effects of cost drivers to vary from phase to phase. Detailed CoCoMo will not be represented in the scope of this paper.

CoCoMo was selected over some of the other available models on software cost estimation for the following reasons:

- Being developed at TRW, CoCoMo is based on the historical data of 63 TRW projects of various sizes. For the given development environment, the estimates are said to be within 20% of the actual estimates 68% of the time. While not overwhelming, the statistics prove a good track record compared to other models.

- CoCoMo provides most of the adjustment factors that were used on popular earlier models.
- 11 -

- The three CoCoMo versions (basic, intermediate, detailed) provide the full range of estimation accuracy for a given project.

- CoCoMo is well documented and makes carefully stated assumptions for its use.

1.5.3 Putnam's System Sizing Methodology Most comprehensive resource models expect as input an estimate of size in terms of lines of code. Source statements are used mostly because they are easy to relate to and serve to quantify the system in an acceptable way. However, source statement estimates are often derived in an intuitive way and introduce a high level of uncertainty. Putnam[11] describes the need to minimize this level of uncertainty. This is particularly significant during the early system definition or specification stage of project development where a high level of intuitive estimate is required on behalf of management. As development continues into the life cycle and more is learned about the system, the degree of uncertainty is reduced. Thus, to overcome the estimation risks during early phase development, more than one estimate should be made. Putnam recommends three sizing estimates prior to any development of the system begins:

- The smallest possible size
- The most likely size
- The largest possible size

With these three estimates, statistical methods can be used to develop a formula for expected number of delivered source
instructions (DSI). This sizing technique affords management a wider range of estimating flexibility for a particular project.

1.5.4 DeMarco's Specification Measure One of the major problems in using algorithmic cost models is providing sound sizing estimates usually in the form of source or object instructions to be developed. As this is usually difficult to determine in advance, DeMarco establishes a formula for determining the size of a software product early in the software life cycle through requirements and design specification elements[7]. DeMarco's paradigm model constructs sizing estimates on the properties of software specifications and designs. These include number of functional primitives, data elements, input elements, output elements, states, transitions between states, relations, modules, data tokens, and control tokens.

The quantitative approach used is to provide a composite measure made up of countable characteristics of the requirements specification. By this, DeMarco refers to as "Bang", a function measure describing an implementation-independent indicator used as an early, strong predictor of effort. DeMarco describes this effort as "a weight of usable function specified for delivery". Of course, "Bang" is significant only when correlated to the working environment and is continually recomputed as the specification model is revised.

1.5.5 Britcher and Gaffney State Machine Measure The requirements specification can be expressed as a state machine representation. A state machine consists of state data encapsulated by transition
functions. These functions describe transformations that use external input and state values to produce external output and changes to the state data. Britcher and Gaffney[7] demonstrate a correlation between the number of variables defined in a state machine design representation and the product size in source instructions. The "State Machine" consists of two principal parts, the "transition function" which gives rise to the actual code and the "state data" which is the memory of the program. Variables in this case are defined as the unique data required by the state machine's transition function. The measure bridges the gap for deriving an estimated amount of function (in SLOC) the system is to provide early in the design. The developed code-size estimation formulas can be re-applied as the design for the system evolves.

1.6 Scope of the Implementation

The implementation merges the concepts of Boehm, Putnam, Britcher & Gaffney, and DeMarco to construct a Cost Estimation paradigm based on a software requirements specification. The specification incorporates an E-R-A format using established standards and includes additional attribute definitions. The tool performs interactively to generate basic and intermediate CoCoMo results using Boehm's model modified with concepts by Putnam. In addition, Britcher & Gaffney's measure is used with DeMarco's Bang formulations to quantify a SLOC estimate. This result is also inputted to CoCoMo with all generated results formatted as a report format. An historical perspective covering many projects and time frames is maintained through a database application. A keystone to this effort attempts to correlate the highly refined CoCoMo model
normally used in later stages of the development process with the
general sizing 'Bang' formulations during the specification phase.
It is the Britcher & Gaffney measure that is used to bridge these
two approaches.

While I believe useful results can be obtained through this
approach, there are several caveats to remember. As much of the
literature points out, "there is no royal road to software sizing"[1]. In other words, there is no magic formula that could
be used with absolute certainty in every application. It is
understood that any approach needs to be customized for a given
environment and its use be rigorously applied to a number of
projects. When applied in this manner, software cost estimation
methodologies (particularly the one proposed in this paper) should
provide satisfying results.
2. Chapter Two - Requirements for Implementation

2.1 Overview

The approach used in this application is to merge the algorithmic methodology of Boehm's CoCoMo model with the sizing estimates of DeMarco's paradigm model for software specification. Bridging these two diverse methodologies are Britcher & Gaffney's State Machine measure that uses the output of the DeMarco model as input. It then produces values to be used as input to Boehm's model. For comparison, the requirements specification is also applied to the state machine measure.

Using a project requirements specification as input, its form is of the Entity-Relationship-Attribute construct. The E-R-A representation is a textual description of the software product decomposed to a functional level. The E-R-A grammar is exemplified by the design of the cost estimation tool presented in this paper (Appendix II). Cost measurement will be performed at the E-R-A activity/periodic function level and then totaled for the project. There are two areas of cost specific input, one area for overall project data and the other for relating it to the requirements specification. For the project data, the input of development mode and development effort multiplier is used for CoCoMo estimates. For the specification input, there are two additional attribute definitions required for each E-R-A activity/periodic function. These are the complexity function weighting attribute used for 'Bang' estimation and the estimated lines of source attribute composed of three values in the form of smallest / most likely /
largest. The estimated SLOC is totaled on a project level and then applied directly to produce Basic and Intermediate CoCoMo formulations. The 'Bang' measure produces two levels of applications. First, it stands alone to represent a fundamental early sizing estimate. Second, it is used with the Britcher & Gaffney formula to produce an additional SLOC estimate. This too, is applied to the CoCoMo formulations. Finally, the E-R-A specification is used with the Britcher & Gaffney formula to also produce a SLOC estimate.

The E-R-A specification document is parsed by the tool to scan for the appropriate attribute definitions. Other input data needed for the costing applications are obtained through the interactive session or derived from supporting tables. The output of the tool is produced in report form to comparatively present the results of Basic/Intermediate CoCoMo and 'Bang'. This includes Programmer-Month effort, development schedule, productivity, and average staffing of personnel. A UNIX database tracks each invocation of the tool to produce cost estimates. Management can then evaluate the progress anytime during the projects development or compare the current results with that of previous projects.

2.2 Application of Boehm's Constructive Cost Model

All three CoCoMo models assume two basic cost relationships. First, between the size of the software being developed and total development effort. Second, between elapsed time and development effort. The relationships are in the form of:
\[ b \\
PM = a(KDSI) \\
\]

where PM is development effort in Programmer-Months, KDSI is thousands of delivered source statements, and 'a' and 'b' are parameters dependent on the version being used and the mode of development.

\[ d \\
TDEV = c(PM) \\
\]

where TDEV is the development schedule in months, and 'c' and 'd' are parameters dependent on the mode of development.

CoCoMo identifies a series of three development modes which designates the type of development environment that is in use. These development modes are:

- **Organic Mode** - describes a small development group that produces software in a highly familiar in-house environment.

- **Embedded Mode** - describes a product that must operate within a strongly coupled complex of hardware, software, regulations, and operational procedures.

- **Semidetached Mode** - describes an intermediate stage between the organic and embedded modes.

The refined intermediate and detailed CoCoMo models utilize the concept of cost drivers. These are additional adjustment factors that influence the effort required to produce a software product. The models identify 15 cost drivers grouped into four categories.
- 18 -

- Product Attributes -

- RELY - Required Software Reliability: The probability that software performs its intended functions satisfactorily over its next run or its next quantum of execution time.

- DATA - Data Base Size: The amount of data to be assembled and stored in non-main storage (tapes, disks, drums, bubble memory, etc.) by the time of software acceptance.

- CPLX - Product Complexity: Software development effort as a function of the level of complexity of the module to be developed.

- Computer Attributes -

- TIME - Execution Time Constraint: A function of the degree of execution time constraint imposed upon a software subsystem.

- STOR - Main Storage Constraint: A function of the degree of main storage constraint imposed on a software subsystem.

- VIRT - Virtual Machine Volatility: A function of the level of volatility of the virtual machine underlying the subsystem to be developed.

- TURN - Computer Turnaround Time: A function of the level of computer response time experienced by the project team developing the subsystem.
• Personnel Attributes -

  • ACAP - Analyst Capability: An indication of how much the nominal level is to be multiplied to account for the difference in the capability of the analysts, with respect to the nominal level of analyst activity.

  • AEXP - Applications Experience: A function of the level of applications experience of the project team developing the software subsystem.

  • PCAP - Programmer Experience: A function of the level of capability of the programmers who will be working on the software module.

  • VEXP - Virtual Machine Experience: A function of the level of virtual machine experience of the project team developing the software module.

  • LEXP - Programming Language Experience: A function of the level of programming language experience of the project team developing the software module.

• Project Attributes -

  • MODP - Modern Programming Practices: A function of the degree to which modern programming practices (top-down, structured design, etc.) are used in developing software.

  • TOOL - Use of Software Tools: A function of the degree to which software tools are used in developing the software subsystem.

  • SCED - Required Development Schedule: A function of the level of schedule constraint imposed on the project team.
developing a software subsystem.

Each cost driver is ranked on a scale to indicate its relative importance to a particular product. The multipliers are applied to the estimate of effort (nominal) determined from the above basic effort equation to produce the now refined estimate of effort. The cost drivers are ignored in the basic model, but in the intermediate model are applied to either the whole product or the components of the product.

2.3 Application of Putnam's System Sizing Technique

Putnam describes his concept of feasibility sizing as a means of early estimating. As described earlier, it is founded on the notion that source lines of code are a particularly difficult quantity to determine far in advance. This gives rise to the application of three estimates: the smallest possible size, the most likely size, and the largest possible size. The results provide for an improved level of estimation accuracy when combined with Boehm's CoCoNo model. Each E-R-A component will accommodate an additional attribute definition to apply these sizing estimates. The incorporation of this value allows E-R-A specification documents to be measured against CoCoNo results.

2.4 Application of DeMarco's Specification Measure

DeMarco designed his software sizing measure around the specification model as an early indicator of scope and complexity. The dataflow diagram is exemplified in his model, but it is the Entity-Relationship-Attribute format that is used in this
application. The E-R-A specification defines a clear statement of
the problem that can rigorously be decomposed into its lowest level
sub-components. This measured evaluation DeMarco describes as a
"Bang" indicator and is used as a general predictor of effort.

The completely decomposed problem statement as depicted in the E-
R-A specification is processed at the component level. Each
component is analyzed to determine the transformation of input
level to output level tokens. This value is then weighted for size
correction and then applied against the complexity for its given
function class. Finally each of the component results are summed
for the project to provide the "Bang" measure. DeMarco defines the
following categories as relevant for complexity:

<table>
<thead>
<tr>
<th>Category</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Separation</td>
<td>Primitives that divide incoming data items.</td>
</tr>
<tr>
<td>Amalgamation</td>
<td>Primitives that combine incoming data items.</td>
</tr>
<tr>
<td>Data Direction</td>
<td>Primitives that steer data according to a control variable.</td>
</tr>
<tr>
<td>Simple Update</td>
<td>Primitives that update one or items of stored data.</td>
</tr>
<tr>
<td>Storage Management</td>
<td>Primitives that analyze stored data, and act based on the state of that data.</td>
</tr>
<tr>
<td>Edit</td>
<td>Primitives that evaluate net input data at the person machine boundary.</td>
</tr>
<tr>
<td>Verification</td>
<td>Primitives that check for and report on internal inconsistency.</td>
</tr>
<tr>
<td>Text Manipulation</td>
<td>Primitives that deal with text strings.</td>
</tr>
<tr>
<td>Synchronization</td>
<td>Primitives that decide when to act or prompt others to act.</td>
</tr>
<tr>
<td>Output Generation</td>
<td>Primitives that format net output data flows (other than tabular outputs).</td>
</tr>
</tbody>
</table>
Display - Primitives that construct two-dimensional outputs (graphs, pictures, etc).

Tabular Analysis - Primitives that do formatting and simple tabular reporting.

Arithmetic - Primitives that do simple mathematics.

Initiation - Primitives that establish starting values for stored data.

Computation - Primitives that do complex mathematics.

Device Management - Primitives that control devices adjacent to the computer boundary.

By itself, "Bang" appears to possess little meaning and its value lies in being able to evaluate the result compared to other projects of similar scope and complexity. Properly used, DeMarco's measure can predict early measures of effort and resource expenditure.

2.5 Application of Britcher and Gaffney's State Machine Measure

The Britcher and Gaffney measure demonstrates the length or size (in source lines of code) of programs represented as state machines can be reliably estimated based on the number of internal state machine variables. The application of this technique is directed towards the E-R-A specification and DeMarco's software sizing paradigm where the estimated "Bang" can be used to represent the number of state machine variables. The adapted formula is presented as follows:

\[ \text{SLOC} = [(\text{constant})^\text{(expansion)}^\text{(un-commented)}] \times [V^\text{LOGe}^V] \]

where: constant = environment specific weight
expansion = language density weight (assembler is nominal)
uncommented = percent of measured code that is uncommented
V = number of state machine variables as represented by 'Bang'
Britcher and Gaffney are the pivotal focal point in transforming the generalized software sizing measure of DeMarco to a number of source lines of code (SLOC). This value then feeds the CoCoMo model that draws heavily on SLOC as the major input variable for all its estimations.

Clearly, as experience is gained in sizing several projects, the state machine formula may be modified accordingly.

2.6 Design Consideration Impacted By Requirements—

The cost estimation tool is written in Bourne "Shell" and produced using a series of tables that maintain the various multiplier and weighting factors used to compute the cost formulas of Boehm and DeMarco. The advantage provides flexibility for projects by recalibrating the measures to a specific environment when needed. It is assumed that for proper project control and planning, the E-R-A specification will be continually updated as development progresses. The use of a database will provide timely monitoring throughout the development period. The cost formulas are recomputed for each invocation with a clear audit trail of the estimates.
3. Chapter Three - Design of Cost Estimation Application

3.1 Overview

The cost estimation tool is designed to operate interactively. It captures basic background information and then perform E-R-A analysis and report results in background mode. This allows users to conveniently invoke the tool with most of the processing performed outside the interactive session. The mechanics of the procedure are illustrated in Figure 3-1 and Figure 3-2. The system is sub-divided into 5 major areas of processing. The areas are discussed in the following sections.

3.2 Obtaining Interactive Input

Invocation of the tool is provided via an interactive interface (See Figure 3-3) designed to capture three groups of data. The first group is project background data and this data is used for historical perspective. This includes project identifying characteristics such as project name, type, development language, etc. These characteristics are significant when the results are compared over many projects. The second group of data includes cost estimating data on a project level, CoCoMo development modes, and CoCoMo effort multiplier scaling factors. The last grouping is I/O related and includes the file name of the E-R-A specification used to compute the cost estimate. To minimize operator input during this first stage, the procedure has been designed to accommodate default values when deemed appropriate. All additional processing is accomplished in background mode.
3.3 Scanning the Specification Input File

The E-R-A specification provides the input for CoCoMo/Bang estimation and is scanned for specific attribute definitions. "Input" and "Output" definitions are counted for each activity/periodic function and are used in conjunction with the "complexity_factor" definition for "Bang" computation. "Estimated_source" values are also scanned at the activity/periodic function level and are used for computation of Basic and Intermediate CoCoMo. Appendix I contains the Backus-Naur-Form (BNF) specification of the E-R-A syntax. Appendix II contains the E-R-A specification for the design of this cost estimation tool and is representative of the appropriate style required for input.

3.4 Calculation of the Cost Estimation

"Input" and "Output" definitions are counted for each activity/periodic function and then the function token count (FTC) sum is applied to a functional primitive size correction table and multiplied by a complexity weight (CW). The weight is determined via table lookup of a functional complexity weighting factor table. The "Complexity_Factor" attribute is used to determine the weight for that activity. The resulting Function Bang (FB) value for each activity/periodic function is totaled for generate the Project Bang (PB) measure. The following restates the formula in more graphic terms:
For each [ 'Activity' | 'Periodic Function' ] in E-R-A Spec
Do
   TC = Sum of 'Input' and 'Output' Definitions
   CC = Table Lookup of TC For Size-Correction
   CW = Table Lookup of CC Based on 'Complexity-Factor'
   PTC = PTC + TC /* Project Token Count */
   FB = CC * CW /* Function Level Bang */
   PB = PB + FB /* Project Level Bang */
   FB = TC = 0
Done

Next the resulting Project Bang (PB) and Project Token Count (PTC) values are applied to the Britcher & Gaffney state machine measure that results in a source lines of code estimate (KDSI) for each:

   SLOC1 = [ (constant)"(expansion)"(%un-commented)] X [(PB)LOGe(PB)]
   SLOC2 = [ (constant)"(expansion)"(%un-commented)] X [(PTC)LOGe(PTC)]

Next the 'Estimated_Source' attribute consisting of three source estimates in the form of smallest / most likely / largest is summed for each Activity and Periodic Function to obtain project level totals. These values are then applied to Putnam’s Estimated Source Formula to obtain an average SLOC estimate:

   AverageSLOC = (smallest + (4 * most likely) + largest) / 6

Finally, all of these 6 SLOC estimates (smallest, most likely, largest, average, state machine token, state machine Bang) are applied to the Basic and Intermediate CoCoMo formulas. The results are produced in four groupings for each of the two CoCoMo models. They are, Effort in Programmer Months, Development Schedule in Programmer Months, Productivity, and Average Staffing.
3.5 Generating the Report

The report is generated once per invocation of the estimation tool and consists of a header section and a detailed section (See Figures 3-4A and 3-4B). The header section provides project specific information such as profile data and includes much of the background values specified during the tool's interactive invocation. The detailed section provides the following cost relevant summary: Total KDSI, Project Bang, Effort in Programmer-Months (Basic/Intermediate), Development Schedule in Programmer-Months (Basic/Intermediate), Productivity (Basic/Intermediate), and Average Staffing (Basic/Intermediate). Each printed report also itemizes all invocations of the tool since the project's inception and the results are printed in reverse chronological order. This provides for convenient on-going analysis of development progress.

3.6 Archiving the Data

Fundamental in the design of the cost estimating tool, is ability to invoke the tool many times during the development of the project and maintain an historical perspective of development costs. The information is invaluable for monitoring progress during various phases of the development life cycle as well as using the data to correlate between other developing projects. The key is providing a method to archive the data not by a life-cycle tag, but by julian date of the day the tool is invoked. As described earlier, life cycle phases are not distinct entities that are invoked serially and often overlap during project development. Management may perceive significant specification changes taking place
irrespective of the life cycle phase thus making the date of invocation a more meaningful measure. The cost data is maintained for each of the DeMarco, Boehm, and Britcher & Gaffney models plus the E-R-A input specs and report files are kept for cross reference. The database (Figure 3-5) is hierarchical in structure and allows for multiple projects to be maintained. Each project keeps its own set of cost data by julian date. The database is sub-divided to contain both active and inactive sets of project cost estimates that allows for important comparisons long after the project is implemented. Figure 3-5 illustrates the tools' database structure layout.
Figure 3-1. The Software Cost Estimator Procedure Flow
SOFTWARE COST ESTIMATION PROCESS FLOW

E-R-A SPECIFICATION (TOKENS) → SUM 'ESTIMATED-SOURCE' ATTRIBUTES (SLOC) → COMPUTE SOURCE LINES OF CODE (BRITCHER & GAFFNEY) (SLOC) → COMPUTE BASIC & INT. COCOMO (BOEMM) (SLOC) → EFFORT

E-R-A SPECIFICATION (TOKENS) → COMPUTE 'BANG' (DEMARCO) (BANG) → COMPUTE SOURCE LINES OF CODE (BRITCHER & GAFFNEY) (SLOC) → SCHEDULE

E-R-A SPECIFICATION (TOKENS) → COMPUTE SOURCE LINES OF CODE (BRITCHER & GAFFNEY) (SLOC) → PRODUCTIVITY

E-R-A SPECIFICATION (TOKENS) → COMPUTE SOURCE LINES OF CODE (BRITCHER & GAFFNEY) (SLOC) → AVG. STAFFING

---

Figure 3-2. The Software Cost Estimator Measure Flow
Figure 3-3. The Software Cost Estimator Front End
### SOFTWARE COST ESTIMATION TOOL

**DATE:** MM/DD/YY  **HH:MM**  
**PROJECT ID:** 9999  

**PROJECT PROFILE DATA:**  
- **COMPUTER TYPE:** XXX  
- **APPLICATION TYPE:** XXX  
- **PERSONNEL CONTINUITY ESTIMATE:** XXX  
- **PRINCIPAL DEVELOPMENT LANGUAGE:** XXX

<table>
<thead>
<tr>
<th>JULIAN DATE</th>
<th>TOTAL KDSI</th>
<th>PROJECT BANG</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SM ML LG ED SS1 SS2</td>
<td>FP TC1 TCavg FSB</td>
</tr>
<tr>
<td>86150</td>
<td>999.9 999.9 999.9 999.9 999.9 99.99</td>
<td>9999 9999 9999 9999</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JULIAN DATE</th>
<th>EFFORT IN PROGRAMMER-MONTHS (PM)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>SM ML LG ED SS1 SS2 SM ML LG ED SS1 SS2</td>
</tr>
<tr>
<td>86150</td>
<td>999.9 999.9 999.9 999.9 999.9 99.99 999.9 999.9 999.9 99.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JULIAN DATE</th>
<th>DEVELOPMENT SCHEDULE IN PROGRAMMER-MONTHS (PM)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>BASIC COCOMO INTERMEDIATE COCOMO</td>
</tr>
<tr>
<td>86150</td>
<td>999.9 999.9 999.9 999.9 999.9 99.99 999.9 999.9 999.9 99.99</td>
</tr>
</tbody>
</table>

**Figure 3-4A.** The Software Cost Estimator Report Layout (Page 1)
## Software Cost Estimation Tool

**Abbreviation Key:**
- **RCO**: REVERSE CHRONOLOGICAL ORDER
- **FSP**: FULL TIME SERVICE PERSONNEL
- **SM**: SMALLEST
- **ML**: MOST LIKELY
- **LG**: LARGEST
- **ED**: ESTIMATED DELIVERED SOURCE INSTRUCTIONS
- **SS1**: SOFTWARE SIZING METRIC (Using 'Bang' As Input)
- **SS2**: SOFTWARE SIZING METRIC (Using Token Count As Input)
- **FP**: TOTAL OF ALL ACTIVITIES / PERIODIC ACTIVITIES
- **TC1**: PROJECT TOKEN COUNT SUM
- **TCavge**: AVERAGE TOKEN COUNT PER PRIMITIVE
- **FSB**: FUNCTION STRONG BANG

### Figure 3-4B. The Software Cost Estimator Report Layout (Page 2)
Figure 3-5. The Software Cost Estimator Database Structure
4. Chapter Four - Implementation of Cost Estimation Tool

4.1 Overview

The tool is written in Bourne "Shell" and operates on an AT&T 3B5 running System V UNIX. All user-definable, cost-formula attributes are table-driven to maintain program independence and ease of maintenance. The following tables are supported:

[A] DeMarco Functional Primitive Size Correction - Corrects input/output token value totals at the function level.

[B] DeMarco Functional Complexity Weighting Factors - Adjusts the function level 'Bang' for a given degree of complexity.

[C] CoCoMo Schedule Effort and Multiplier - Establishes the multiplier and exponent values used in basic and intermediate CoCoMo computation.

[D] CoCoMo Development Effort Multiplier - Provides an overall weighting for intermediate CoCoMo based upon the effort rankings entered during initial tool invocation.

[E] Software Size Measure Weight - Weighs Britcher & Gaffney's measure with a given development language and degree the source is to be commented.

4.2 Invocation and Interactive Data Entry

The tool is invoked interactively through a shell front end to capture and validate initial project related data. The current machine date is captured and converted to julian format for
database access. The project is identified to the system through a
unique ID code that will be used also as a database key. When a
new project is entered for the first time, the system will derive
the project ID. Next, project background data is captured
including the project name, the principal development language, the
level of personnel continuity, type of development computer, and
type of project application. This project background data will be
helpful to management for tracking purposes and justifying the
measure results. Project background data reflects overall project
conditions and is used in the CoCoMo and Britcher & Gaffney
measures. The degree of commented source is an important value in
the Britcher & Gaffney metric state machine formula. For CoCoMo
cost estimation, data is obtained that describes the project
development mode (organic / embedded / semidetached) and a ranking
of the development effort multipliers. Each of the 15 multipliers
are ranked on a scale of 1-6 to describe its relative effort from
very low (=1) to extra high (=6). The final data entry is the UNIX
path name that locates the E-R-A file used for cost analysis.

Once all interactive data is captured and validated syntactically
for correctness, the main program is invoked in background mode to
perform the remaining processing. Here the E-R-A input file is
evaluated along with the interactive data entries to produce a cost
estimation summary and database update. The summary report can be
printed locally or redirected to a UNIX file for inspection. An
added feature to the user is producing the report containing all
previous invocations for a given project ID. This is useful for
measuring prior project results of the same or different projects.
4.3 Processing the E-R-A Specification File

The Entity-Relationship-Attribute model is built upon the requirements specification for project development. Each fragment or basic unit is known as an entity. The entity defines the unit and relationships with other units. The tool will interpret entity types such as Activities and Periodic Functions. Each entity describes numerous attribute definitions that include "Inputs" and "Outputs" which are measured by the tool and are represented as a token count total. The Activity or Periodic Function entity require additional attribute definitions for "complexity-factor" and "estimated-source" that provide a method of interweaving Boehm/DeMarco methodologies within the E-R-A specification style. The process will scan for the attribute name and will generate error messages for invalid entries. For example, the "complexity-factor" attribute requires a value in conformity to DeMarco's specification model and will be validated against the complexity weighting factor table. The "estimated-source" attribute will require an tri-numeric value separated by slashes used by the CoCoMo model. This corresponds to the smallest / most likely / largest estimate and will be validated for correct syntax.

4.4 Cost Computation and Report

The interactive and file input are used to compute the respective project 'Bang' and Basic/Intermediate CoCoMo costs. Several tables provide key multiplier and weighting factors used in the computation. The tables provide operational independence should the values need to be changed or recalibrated in any way.
The following 'Bang' estimates are computed:

[TC] Total of all Activities and Periodic Functions.

[TCP] Total of all tokens about each Activity/Periodic Function boundary.

[TCavg] Average token count per primitive ([TCP]/[TC]).

[PB] Project Bang -
(For each activity/periodic function in the specification find total of all tokens about each boundary, then use the primitive size correction table and multiply the resulting value by the respective complexity weight. The result is then summed of all remaining activities/periodic functions.)

The following Britcher & Gaffney estimates are computed:

[SLOC1] Total Software Size Lines of Source (in thousands): KDSI
(Based on input from 'Bang' as a state machine variable and is weighted to a value that takes into consideration language used, percent of code commented, and an environment constant. The result will be input to Basic and Intermediate CoCoMo.)

[SLOC2] Total Software Size Lines of Source (in thousands): KDSI
(Based on input from the token count total as a state machine variable and is weighted to a value that takes into consideration language used, percent of code commented, and an environment constant.
The result will be input to Basic and Intermediate CoCoMo.)

The following CoCoMo Estimates are computed:

[DSI] Total Delivered Lines of Source (in thousands): KDSI.
(Computed by taking sum of the "estimated-source" attribute for each smallest / most likely / largest values of all activities/periodic functions in the project.)

(Computed by using the CoCoMo Exponent/Multiplier table to multiply KDSI by the appropriate effort multiplier and exponent for the given development mode.)

(Computed by using the CoCoMo Exponent/Multiplier table to multiply PM-Effort from above by the appropriate effort multiplier and exponent for the given development mode.)

[PROD] Productivity (DSI/PM): Basic/Intermediate
(Computed by dividing PM-Effort from above into KDSI.)

(Computed by dividing PM-Schedule from above into PM-effort.)

Each of these totals DSI, PM, TDEV, PROD, and STAFF will consist of a
set of six computed estimates as follows:

SM - Based on Smallest possible source estimate
ML - Based on Most Likely possible source estimate
LG - Based on Largest possible source estimate
ED - Based on Expected number of DSI computed through Putnam's formula of:
      \( (SM + (4 \times ML) + LG) / 6 \)
SS1 - Based on Britcher & Gaffney's Software Size Estimate using 'Bang' as input.
SS2 - Based on Britcher & Gaffney's Software Size Estimate using token count total as input.

The report output uses the keys of project ID and julian date to extract all archived data. The results along with all current computed data as described above are printed in reverse chronological order (RCC) to provide an historical perspective for analysis.

4.5 The Cost Estimation Database

Instrumental in cost estimation is being able to evaluate multiple projects over a long time frame. The advantage of this will enable future estimators to fine tune the model(s) for any given environment. The database as implemented offers this flexibility and also allows for further growth. The implementation uses the UNIX file system hierarchy (inverted tree structure). The archiving scheme is essentially organized to maintain an active and an inactive file structure which is useful for when the database ages. Under each structure, records are maintained for each project ID; within each project by julian date; and within date by respective cost model. The design provides the ability to reconstruct an estimate by maintaining not only the computed data but by keeping the E-R-A input files and formatted report files as well. Additionally, all tables used during the computation are
maintained for audit purposes.

4.6 Evaluating The Cost Estimation Results

To actually assess the accuracy of the various cost estimation measures, the tool was applied to the E-R-A specification of the tool itself. The results as illustrated in Figure 4-1 indicate the kind of values generated with the tool. The profile of the tool as shown on the upper portion of the report, describes the cost estimation tool background data. The tool was developed on a 'mini' computer as a support ('sup') software application containing approximately 10 percent comments. The principal development language was Bourne 'shell', the level of personnel continuity during development was 'low', and the CoCoMo Development Mode of the tool was estimated as being organic ('org').

The E-R-A file processed (Appendix II), produced a total of 17 entities and a total of 228 Input/Output tokens. The average token count per entity was 13.4. DeMarco's Bang formulation produced a result of 312.3 for the Project Bang. Among the delivered source instruction estimates, the range was from 820 to 1750 DSI. The actual source code was a little over 1200 lines of shell lines of code and most closely matches the Most-Likely estimate.

The Effort in Programmer Months (PM) estimate ranged from 1.2 to 4.3 months for Basic and 1.6 to 5.7 months for Intermediate CoCoMo. To be meaningful, these results should be measured in conjunction with the Average Staffing result. Here the estimate for Full-time Service Personnel (FSP) ranges from 0.4 to 1 individuals using Basic and 0.5 to 1.1 individuals using Intermediate CoCoMo. Since
the project was developed using a staff total of 1 (the author), the results are well within reason.

The Development Schedule in Programmer Months estimate ranged from 2.6 to 4.3 months for Basic and 2.9 to 4.8 months for Intermediate CoCoMo. The actual development schedule was approximately 6 weeks and is about average when adjusting these results to the Average Staffing value. The Productivity results are a function of the DSI estimates and ranged from estimates of 300 to 400 lines of code per month for both Basic and Intermediate CoCoMo.
**SOFTWARE COST ESTIMATION REPORT**

**DATE:** 07/21/86 (86202)

**PROJECT ID:** a101

**PROJECT PROFILE DATA:**

- **PROJECT NAME:** E-R-A Cost Estimation Tool
- **COMPUTER TYPE:** mfn
- **COCOMO MODE:** org
- **APPLICATION TYPE:** sup
- **PERCENT COMMENTS:** 10
- **PERSONNEL CONTINUITY ESTIMATE:** nom
- **PRINCIPAL DEVELOPMENT LANGUAGE:** th

<table>
<thead>
<tr>
<th>JULIAN DATE</th>
<th>TOTAL DSI</th>
<th>PROJECT BANG</th>
</tr>
</thead>
<tbody>
<tr>
<td>(RCO)</td>
<td>SMALL LIKELY LARGEST AVERAGE FSM BANG</td>
<td>ENTITY TOKENS AVERAGE BANG</td>
</tr>
<tr>
<td>86202</td>
<td>820 1285 1750 1285 573.8 544.8</td>
<td>17 228 13.4 312.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JULIAN DATE</th>
<th>-effort in programmer months (PM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(RCO)</td>
<td>BASIC COCOMO</td>
</tr>
<tr>
<td>86202</td>
<td>1.8 3.1 4.3 3.1 1.3 1.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JULIAN DATE</th>
<th>Development Schedule in programmer months (PM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(RCO)</td>
<td>BASIC COCOMO INTERMEDIATE COCOMO</td>
</tr>
<tr>
<td>86202</td>
<td>1.8 3.1 4.3 3.1 1.3 1.3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JULIAN DATE</th>
<th>Productivity (DSI/PM)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(RCO)</td>
<td>BASIC COCOMO INTERMEDIATE COCOMO</td>
</tr>
<tr>
<td>86202</td>
<td>0.4 0.4 0.4 0.4 0.4 0.4</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>JULIAN DATE</th>
<th>Average Staffing (FSP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(RCO)</td>
<td>BASIC COCOMO INTERMEDIATE COCOMO</td>
</tr>
<tr>
<td>86202</td>
<td>0.6 0.8 1 0.8 0.4 0.4</td>
</tr>
</tbody>
</table>

**Figure 4-1. Cost Estimation Results**
5. Chapter Five - Conclusions and Extensions

The Requirements Specifications SCE Tool establishes a benchmark for initial indications of project size and effort. The product ties together E-R-A specifications with established CoCoMo estimation models to provide strong measurements early in the project life cycle. This is when the least is known about the project and the greatest need exists for a prediction. When used over time and measured against past projects with similar characteristics, the tool adds further insight to future planning.

5.1 Extensions

This project can be extended through many possible extensions. First, the resulting database where much of the SCE data is stored can be analyzed after using the tool on a number of projects. Statistical and graphical representations can be performed to compare an on-going project development or by measuring multiple projects for a given organization. Second, the CoCoMo model provides the ability to tailor the model for a given environment[1]. The CoCoMo tailoring methodology is designed to increase the precision and stability of the model and is accomplished by deleting unnecessary cost drivers in an heuristic manner.

Future work can involve new applications in the areas of software cost estimation technology mentioned in the Cost Estimation Workshop Study[1'] and are highlighted below:

- In Artificial Intelligence Systems - The AI or knowledge-based
system is developed in an un-traditional environment that is usually less formal and more evolutionary. There is greater interaction between developer and user to develop the knowledge base than present software development projects. The goal then is to develop cost estimation models for this new environment. For example, the number of data items and the number of logical inference rules may be used as a replacement of source lines of code (in SCE) for AI systems.

- In Distributed Development Workstations - The use of new applications involving for example; ADA, program generators, fourth generation languages, VHLs (very high level languages), and PDLs (program design languages) will have a significant effect on existing software cost estimation models. Model parameters such as size, complexity, application, and environment will have to be mapped to these new technologies. New definitional issues and data collection techniques are needed to give rise to new values in the parameters of existing SCE models.
APPENDIX I - BNF SYNTAX FOR E-R-A SPECIFICATION

<er-spec> ::= <er_title> <NL> <er_body>
<er_title> ::= PROCESS <name>
<er_body> ::= <frame> | <er_body> <frame>
<frame> ::= <NL> <frame_header> <frame_body> | <NL> MODE_TABLE <mode_table>
<frame_header> ::= <frame_type> : <name>
<frame_body> ::= <relation> | <frame_body> <relation>
<relation> ::= <NL> <BLANK> <relation_type> : <relation_value>
<frame_type> ::= <capital_letter> <lower_case_text>
<relation_type> ::= <lower_case_text>
<relation_value> ::= <text> | <structure> | <text> <text_cont>
;text_cont ::= <NL> <BLANK> : <text> | <NL> <BLANK> : <text> <text_cont>
<structure> ::= <name> | <text> | <text> <structure> | <name> <structure> | <NL> : <structure>
:name ::= $ <text> $ | <mode_name>
<mode_name> ::= * <text> *
<mode_table> ::= <mode_list> <initial_mode> <transition_body>
<mode_list> ::= <NL> Modes : <mode_list_body>
<mode_list_body> ::= <mode_name> | <mode_list_body> <NL> <BLANK> : <mode_name>
<initial_mode> ::= <NL> Initial Mode : <mode_name>
<transition_body> ::= <NL> Allowed Transition : <transition_list>
<transition_list> ::= <transition> | <transition_list> <transition>
<transition> ::= <NL> event : <mode_name> -> <mode_name>
<lower_case_text> ::= <lower_case_char> | <lower_case_text> <lower_case_text>
<lower_case_char> ::= a b c d e f g h i j k l m n o p q r s t u v w x y z 0 1 2 3 4 5 6
7 | 8 | 9 | <symbol>

<symbol> ::= # | % | & | ( | ) | ?
<capital_letter> ::= A | B | C | D | E | F | G | H | I | J | K |
                  L | M | N | O | P | Q | R | S | T | U | V |
                  W | X | Y | Z
<text> ::= <char> | <text> <char>
<char> ::= <lower_case_char> | <capital_letter>
APPENDIX II - SCE E-R-A SPECIFICATION

PROCESS: A Requirements Specification Software Cost Estimation Tool

Comment: This is the requirements specification for the requirements specification software cost estimation tool in the ERA format

Activity : Process_SPEC
  keywords : read through ERA file; process attributes
  input : $record_line$
  input : $min_attr_table$
  input : $complex_weight_table$
  output : $record_line$
  output : $eof$
  output : $token_total$
  complexity_factor : 've'
  estimated_source : 150/200/250
  required_mode : "START"
  action : if attribute invalid, then required mode changes to "ILLEGAL"

Activity : Compute_Bang
  keywords : compute bang from era specifications
  input : $token_total$
  input : $complex_weight_table$
  input : $size_correction_table$
  output : $function_bang$
  complexity_factor : 'ar'
  estimated_source : 20/40/60
  required_mode : "NORMAL"

Activity : Compute_Basic_Cocomo
  keywords : Compute basic level cocomo estimates
  input : $sm_source_est$
  input : $sml_source_est$
  input : $lg_source_est$
  input : $sdl_source_est$
  input : $ssl_source_est$
  input : $ssl2_source_est$
  input : $cocomult_table$
  output : $basic_effort_sm$
  output : $basic_effort_m1$
  output : $basic_effort_lg$
  output : $basic_effort_ed$
  output : $basic_effort_ssi$
  output : $basic_effort_ss2$
  output : $basic_schedule_sm$
  output : $basic_schedule_m1$
  output : $basic_schedule_lg$
  output : $basic_schedule_ed$
  output : $basic_schedule_ssi$
  output : $basic_schedule_ss2$
  output : $basic_prod_sm$
output : $basic_prod_ml$
output : $basic_prod_lg$
output : $basic_prod_ed$
output : $basic_prod_ss1$
output : $basic_prod_ss2$
output : $basic_staff_sm$
output : $basic_staff_ml$
output : $basic_staff_lg$
output : $basic_staff_ss1$
output : $basic_staff_ss2$
complexity_factor : 'ar'
estimated_source : 25/50/75
required_mode : "NORMAL"

Activity : Compute_BG_Measure
keyword : compute britcher/gaffney bang source estimate
input : $software_sizing_table$
input : $percent_comments$
input : $token_total$
input : $project_bang$
output : $ss1_source_est$
output : $ss2_source_est$
complexity_factor : 'ar'
estimated_source : 20/40/60
required_mode : "NORMAL"

Activity : Compute_Int_Cocomo
keywords : compute intermediate level cocomo estimates
input : $sm_source_est$
input : $ml_source_est$
input : $lg_source_est$
input : $ed_source_est$
input : $ss1_source_est$
input : $ss2_source_est$
input : $coc_mult_table$
input : $coc_effort_table$
input : $eff_adj$
output : $int_effort_sm$
output : $int_effort_ml$
output : $int_effort_lg$
output : $int_effort_ed$
output : $int_effort_ss1$
output : $int_effort_ss2$
output : $int_schedule_sm$
output : $int_schedule_ml$
output : $int_schedule_lg$
output : $int_schedule_ed$
output : $int_schedule_ss1$
output : $int_schedule_ss2$
output : $int_produced_sm$
output : $int_produced_ml$
output : $int_produced_lg$
output : $int_produced_ed$
output : $int_produced_ss1$
output: \$\text{int\_prod\_ss2}\$
output: \$\text{int\_staff\_sm}\$
output: \$\text{int\_staff\_ml}\$
output: \$\text{int\_staff\_lg}\$
output: \$\text{int\_staff\_ed}\$
output: \$\text{int\_staff\_ss1}\$
output: \$\text{int\_staff\_ss2}\$
complexity\_factor: 'ar'
estimated\_source: 50/75/100
required\_mode: "NORMAL"

Activity: Running\_Total
keywords: compute running totals from processing era
input: \$\text{function\_bang}\$
input: \$\text{sm\_source\_est}\$
input: \$\text{ml\_source\_est}\$
input: \$\text{lg\_source\_est}\$
input: \$\text{token\_total}\$
input: \$\text{funct\_prim}\$
input: \$\text{avg\_token}\$
output: \$\text{project\_bang}\$
output: \$\text{sm\_source\_est}\$
output: \$\text{ml\_source\_est}\$
output: \$\text{lg\_source\_est}\$
output: \$\text{ed\_source\_est}\$
output: \$\text{token\_total}\$
output: \$\text{funct\_prim}\$
output: \$\text{avg\_token}\$
complexity\_factor: 'ar'
estimated\_source: 20/40/60
required\_mode: "NORMAL"

Activity: Report\_Header
keywords: print report heading for sce results
input: NONE
output: \$\text{sce\_head\_rec}\$
complexity\_factor: 'og'
estimated\_source: 50/75/100
required\_mode: "NORMAL"

Activity: Report\_Body
keywords: print report heading for sce results
input: \$\text{rpt\_lv1\_list}\$
input: \$\text{rpt\_lv2\_list}\$
input: \$\text{rpt\_lv3\_list}\$
input: \$\text{rpt\_lv4\_list}\$
input: \$\text{rpt\_lv5\_list}\$
output: \$\text{sce\_body\_rec}\$
complexity\_factor: 'og'
estimated\_source: 50/75/100
required\_mode: "NORMAL"

Activity: Update\_Info\_Seg
keywords: update database info segment with sce results
input: \$\text{proj\_id}\$
input: $proj_name$
input: $appl_type$
input: $jul_date$
input: $dev_lang$
input: $comp_type$
input: $pers_cont$
output: $info_seg_list$
complexity_factor: 'su'
estimated_source: 20/40/60
required_mode: "NORMAL"

Activity: Update_Cocomo_Seg
keywords: update database cocomo segment with sce results
input: $proj_id$
input: $jul_date$
input: $cocomo_mode$
input: $sm_source_est$
input: $ml_source_est$
input: $lg_source_est$
input: $ed_source_est$
input: $ss1_source_est$
input: $ss2_source_est$
input: $basic_effort_sm$
input: $basic_effort_ml$
input: $basic_effort_lg$
input: $basic_effort_ed$
input: $basic_effort_ss1$
input: $basic_effort_ss2$
input: $basic_schedule_sm$
input: $basic_schedule_ml$
input: $basic_schedule_lg$
input: $basic_schedule_ed$
input: $basic_schedule_ss1$
input: $basic_schedule_ss2$
input: $basic_prod_sm$
input: $basic_prod_ml$
input: $basic_prod_lg$
input: $basic_prod_ed$
input: $basic_prod_ss1$
input: $basic_prod_ss2$
input: $basic_staff_sm$
input: $basic_staff_ml$
input: $basic_staff_lg$
input: $basic_staff_ed$
input: $basic_staff_ss1$
input: $basic_staff_ss2$
input: $int_effort_sm$
input: $int_effort_ml$
input: $int_effort_lg$
input: $int_effort_ed$
input: $int_effort_ss1$
input: $int_effort_ss2$
input: $int_schedule_sm$
input: $int_schedule_ml$
input: $int_schedule_lg$
input: $int_schedule_ed$
input: $int_schedule_ss1$
input: $int_schedule_ss2$
input: $int_prod_sm$
input: $int_prod_ml$
input: $int_prod/lg$
input: $int_prod_ed$
input: $int_prod/ss1$
input: $int_prod/ss2$
input: $int_staff/sm$
input: $int_staff/ml$
input: $int_staff/lg$
input: $int_staff_ed$
input: $int_staff/ss1$
input: $int_staff/ss2$
output: $coc_seg_list$
complexity_factor: 'su'
estimated_source: 75/100/125
required_mode: "NORMAL"

Activity: Update_Bang_Seg
keywords: update database bang segment with sce results
input: $proj_id$
input: $jul_date$
input: $funct_prim$
input: $token_total$
input: $avg_token$
input: $function_bang$
output: $bang_seg_list$
complexity_factor: 'su'
estimated_source: 20/40/60
required_mode: "NORMAL"

Activity: Update_Driver_Seg
keywords: update database cocomo driver segment
input: $proj_id$
input: $jul_date$
input: $prod_rely$
input: $prod_data$
input: $prod_cplx$
input: $comp_time$
input: $comp_stor$
input: $comp_virt$
input: $comp_turn$
input: $pers_acap$
input: $pers_aexp$
input: $pers_pctap$
input: $pers_vexp$
input: $pers_lexp$
input: $proj_modp$
input: $proj_tool$
input: $proj_sced$
input: $effort_adj$
output: $driver_seg_list$
complexity_factor: 'su'
Activity: SCE_Capture
  keywords: capture interactive project information for sce
  input:
  $proj_id$
  $proj_name$
  $dev_lang$
  $percent_comments$
  $pers_cont$
  $comp_type$
  $appl_type$
  $coc_mode$
  $prod_rely$
  $prod_data$
  $prod_cplx$
  $comp_time$
  $comp_stor$
  $comp_virt$
  $comp_turn$
  $pers_acap$
  $pers_aexp$
  $pers_pcap$
  $pers_vexp$
  $pers_lexp$
  $proj_modp$
  $proj_tool$
  $proj_sced$
  $era_file$
  output: NONE
  complexity_factor: 'am'
  estimated_source: 200/300/400
  required_mode: "START"

Activity: Audit_Trail
  keywords: print era audit results for each input record
  input:
  $audit_rec_list$
  output:
  $sce_audit_rec$
  complexity_factor: 'og'
  estimated_source: 30/60/90
  required_mode: "START"

Periodic_Function: Invalid_Token_Sum
  required_mode: "START"
  occurrence:
  an activity or periodic_function token count
  sum equals zero
  input:
  $record_line$
  output:
  $invalid_token_msg$
  action:
  Change mode to "ILLEGAL"
  Continue processing
  complexity_factor: 've'
  estimated_source: 20/30/40

Periodic_Function: Invalid_Complex
  required_mode: "START"
occurrence : an activity or periodic_function containing an
    : incorrect/missing complexity_factor attribute
input      : $record_line$
output     : $invalid_complex_msg$
action     : change mode to "ILLEGAL"
    : continue processing
complexity_factor : 've'
estimated_source : 20/30/40

Periodic_Function : Invalid_Source
required_mode : "START"
ocurrence          : an activity or periodic_function containing an
    : incorrect/missing estimated_source attribute
input      : $record_line$
output     : $invalid_source_msg$
action     : change mode to "ILLEGAL"
    : continue processing
complexity_factor : 've'
estimated_source : 20/30/40
Input: \$min\_attr\_table\$
  media : internal
  structure : array of \$attr\_name\$

Input: \$complex\_weight\_table\$
  media : internal
  structure : \$cw\_table\$

Input: \$size\_correction\_table\$
  media : internal
  structure : \$sc\_table\$

Input: \$software\_sizing\_table\$
  media : internal
  structure : array of structure
    : \$lang\_code\$ \$ss1\_weight\_factor\$ \$ss2\_weight\_factor\$
    : \$lang\_expansion\$

Input: \$percent\_comments\$
  media : cfr
  structure : integer

Input: \$rpt\_lvl1\_list\$
  media : internal
  structure : array of structure
    : \$integer\$ array[1..6] of \$integer\$
    : array[1..4] of \$real\$

Input: \$rpt\_lvl2\_list\$
  media : internal
  structure : array of structure
    : \$integer\$ array[1..6] of \$real\$
    : array[1..6] of \$real\$

Input: \$coc\_mult\_table\$
  media : internal
  structure : \$cm\_table\$

Input: \$coc\_effort\_table\$
  media : internal
  structure : \$ce\_table\$

Input: \$audit\_rec\_list\$
  media : internal
  structure : array of structure
    : \$alphanumeric\$ \$integer\$ \$character\$ \$real\$ \$real\$
    : \$real\$ \$integer\$ \$integer\$
Output : $coc_seg_list$
   media   : internal
   structure : array of structure
      : $alphabetic$  $alphabetic$  $alphabetic$
      : array[1..54] of $real$

Output : $bang_seg_list$
   media   : internal
   structure : array of structure
      : $alphabetic$  $character$  array[1..4] of $real$

Output : $info_seg_list$
   media   : internal
   structure : array of structure
      : $alphabetic$  $alphabetic$  array[1..5] of $character$

Output : $driver_seg_list$
   media   : internal
   structure : array of structure
      : $alphabetic$  $alphabetic$  array[1..16] of $real$

Output : $sce_audit_rec$
   media   : output device
   structure :
      : 'E-R-A Audit Trail'  'Date:'  $date_time$
      : 'Function Name'  'Token Count'  'Cumulative Token Count'
      : 'Size Correction'  'Complexity Code'  'Complexity Weight'
      : 'Function Bang'  'Cumulative Bang'  'Estimated Source'
      : 'Cumulative Source'
      : '----------------------'
   : $audit_rec_list$
   : 'There Are A Total Of '  $funt prima$
   : 'Activities / Periodic Functions Processed'

Output : $invalid_token_msg$
   media   : output device
   structure : "$ERROR* Token Count is 0"
   : 'Activity / Periodic Function Ignored'

Output : $invalid_complex_msg$
   media   : output device
   structure : "$ERROR* Incorrect/Missing Complexity Factor Attribute"
   : 'Activity / Periodic Function Ignored'

Output : $invalid_source_msg$
media : output device
structure : "ERROR" Incorrect/Missing Estimated Source Attribute'
            'Activity / Periodic Function Ignored'

Input_output : $era_file$
media : internal
structure : array of $era_records$

Input_output : $record_line$
media : internal
structure : alphanumerical

Input_output : $function_bang$
media : internal
structure : real number

Input_output : $sm_source_est$
media : internal
structure : real number

Input_output : $ml_source_est$
media : internal
structure : real number

Input_output : $lg_source_est$
media : internal
structure : real number

Input_output : $ed_source_est$
media : internal
structure : real number

Input_output : $ssl1_source_est$
media : internal
structure : real number

Input_output : $ss2_source_est$
media : internal
structure : real number

Input_output : $basic_effort_sm$
media : internal
structure : real number

Input_output : $basic_effort_ml$
media : internal
structure : real number

Input_output : $basic_effort_lg$
media : internal
structure : real number

Input_output : $basic_effort_ed$
media : internal
structure : real number

Input_output : $basic\_effort\_ss1$
media : internal
structure : real number

Input_output : $basic\_effort\_ss2$
media : internal
structure : real number

Input_output : $basic\_schedule\_sm$
media : internal
structure : real number

Input_output : $basic\_schedule\_ml$
media : internal
structure : real number

Input_output : $basic\_schedule\_lg$
media : internal
structure : real number

Input_output : $basic\_schedule\_ed$
media : internal
structure : real number

Input_output : $basic\_schedule\_ss1$
media : internal
structure : real number

Input_output : $basic\_schedule\_ss2$
media : internal
structure : real number

Input_output : $basic\_prod\_sm$
media : internal
structure : real number

Input_output : $basic\_prod\_ml$
media : internal
structure : real number

Input_output : $basic\_prod\_lg$
media : internal
structure : real number

Input_output : $basic\_prod\_ed$
media : internal
structure : real number

Input_output : $basic\_prod\_ssi$
media : internal
structure : real number

Input_output : $basic\_prod\_ss2$
media : internal
structure : real number

Input_output : $basic_staff_sm$
   media : internal
   structure : real number

Input_output : $basic_staff_ml$
   media : internal
   structure : real number

Input_output : $basic_staff_lg$
   media : internal
   structure : real number

Input_output : $basic_staff_ed$
   media : internal
   structure : real number

Input_output : $basic_staff_s1$
   media : internal
   structure : real number

Input_output : $basic_staff_s2$
   media : internal
   structure : real number

Input_output : $int_effort_sm$
   media : internal
   structure : real number

Input_output : $int_effort_ml$
   media : internal
   structure : real number

Input_output : $int_effort_lg$
   media : internal
   structure : real number

Input_output : $int_effort_ed$
   media : internal
   structure : real number

Input_output : $int_effort_s1$
   media : internal
   structure : real number

Input_output : $int_effort_s2$
   media : internal
   structure : real number

Input_output : $int_schedule_sm$
   media : internal
   structure : real number
<table>
<thead>
<tr>
<th>Input_output</th>
<th>media</th>
<th>structure</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\int_{\text{schedule_m1}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{schedule_lg}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{schedule_ed}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{schedule_ss1}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{schedule_ss2}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{prod_sm}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{prod_ml}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{prod_lg}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{prod_ed}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{prod_ss1}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{prod_ss2}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{staff_sm}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{staff_ml}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
<tr>
<td>$\int_{\text{staff_lg}}$</td>
<td>internal</td>
<td>real number</td>
</tr>
</tbody>
</table>
Input_output : $int_staff_ed$
  media : internal
  structure : real number

Input_output : $int_staff_ss1$
  media : internal
  structure : real number

Input_output : $int_staff_ss2$
  media : internal
  structure : real number

Input_output : $proj_name$
  media : internal
  structure : alphanumeric

Input_output : $comp_type$
  media : internal
  structure : alphanumeric

Input_output : $appl_type$
  media : internal
  structure : alphanumeric

Input_output : $pers_cont$
  media : internal
  structure : alphanumeric

Input_output : $dev_lang$
  media : internal
  structure : alphanumeric

Input_output : $proj_id$
  media : internal
  structure : alphanumeric

Input_output : $jul_date$
  media : internal
  structure : integer

Input_output : $coc_mode$
  media : internal
  structure : character

Input_output : $funct_prim$
  media : internal
  structure : integer

Input_output : $avg_token$
  media : internal
  structure : real number

Input_output : $eff_adj$
  media : internal
  structure : real number
Input_output : $proj_sced$
    media : crt
    structure : integer

Type : $era_records$
    structure : 1 to 80 characters

Type : $attr_name$
    structure : 'input' OR 'Input' OR 'input_output' OR 'Input_output'
             OR 'output' OR 'Output' OR 'activity' OR 'Activity'
             OR 'periodic_function' OR 'Periodic_Function'
             OR 'complexity_factor' OR 'Complexity_factor'
             OR 'estimated_source' OR 'Estimated_source'

Type : $cw_table$
    structure : array of structure
             : $class_code$ $cw_weight_factor$

Type : $class_code$
    structure : 'se' OR 'am' OR 'dd' OR 'su' OR 'sm' OR 'ed' OR 've' OR
             : 'tm' OR 'sy' OR 'og' OR 'di' OR 'ta' OR 'ar' OR 'in' OR
             : 'co' OR 'dm'

Type : $cw_weight_factor$
    structure : real number

Type : $token_total$
    structure : integer

Type : $sc_table$
    structure : array of structure
             : $token_count$ $cfpi$

Type : $token_count$
    structure : '2' OR '3' OR '4' OR '5' OR '6' OR '7' OR '8' OR '9' OR
             : '10' OR '11' OR '12' OR '13' OR '14' OR '15' OR '16' OR
             : '17' OR '18' OR '19' OR '20'

Type : $cfpi$
    structure : real number

Type : $lang_code$
    structure : 'asm' OR 'bas' OR 'c' OR 'cob' OR 'pli' OR 'pas'

Type : $ssl_weight_factor$
    structure : real number

Type : $ss2_weight_factor$
    structure : real number

Type : $integer$
    structure : integer

Type : $character$
structure : character

Type : $alphanumeric$
structure : alphanumeric

Type : $real$
structure : real number

Type : $cm_table$
structure : array of structure
   : $cm_model$ $cm_mode$ $cm_eff_mult$ $cm_eff_exp$
   : $cm_sched_mult$ $cm_sched_exp$

Type : $cm_model$
structure : 'bas' OR 'int'

Type : $cm_mode$
structure : 'org' OR 'sem' OR 'emb'

Type : $cm_eff_mult$
structure : real number

Type : $cm_eff_exp$
structure : real number

Type : $cm_sched_mult$
structure : real number

Type : $cm_sched_exp$
structure : real number

Type : $ce_table$
structure : array of structure
   : $ce_efforts$ $ce_abbrev$ $ce_vlow$ $ce_low$ $ce_nom$
   : $ce_high$ $ce_vhigh$ $ce_xhigh$

Type : $ce_efforts$
structure : 'prod' OR 'comp' OR 'pers' OR 'proj'

Type : $ce_abbrev$
structure : 'rely' OR 'data' OR 'cplx' OR 'time' OR 'stor' OR
   : 'virt' OR 'turn' OR 'acap' OR 'aexp' OR 'pcap' OR
   : 'vexp' OR 'lexp' OR 'modp' OR 'tool' OR 'sced'

Type : $ce_vlow$
structure : real number

Type : $ce_low$
structure : real number

Type : $ce_nom$
structure : real number

Type : $ce_high$
structure : real number
Type : $ce_vhigh$
    structure : real number

Type : $ce_xhigh$
    structure : real number

MODE_TABLE
    mode : "ILLEGAL"
    mode : "NORMAL"
    mode : "START"
    mode : "END"

Initial_Mode : "START"

Allowed_Mode_Transitions :

$eof$   : "START" -> "NORMAL"

$invalid_token_msg$ : "START" -> "ILLEGAL"
$invalid_complex_msg$ : "START" -> "ILLEGAL"
$invalid_source_msg$ : "START" -> "ILLEGAL"

$invalid_token_msg$ : "ILLEGAL" -> "START"
$invalid_complex_msg$ : "ILLEGAL" -> "START"
$invalid_source_msg$ : "ILLEGAL" -> "START"

$sce_body_rec$ : "NORMAL" -> "END"
APPENDIX III - SCE KEY DATABASE SEGMENT LAYOUTS

/**************************************************************************/
/*
/* DEMARCO BANG DATA FILE LAYOUT
/*
/*
/* ID VALUE DESCRIPTION
/**************************************************************************/
pro_id A999 /* project identification */
jun_date YYDDDD /* Julian Date */
func_prim 9999 /* functional primitives */
tot_token 9999 /* project token count sum */
avg_token 9999 /* average token count per primitive */
func_bang 9999 /* function strong bang */
/**************************************************************************/

Figure III-1. Bang Database Segment Layout
CoCoMo Cost Driver Database Segment Layout

<table>
<thead>
<tr>
<th>ID</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>proj_id</td>
<td>A999</td>
<td>project identification</td>
</tr>
<tr>
<td>jul_date</td>
<td>YYDDD</td>
<td>Julian Date</td>
</tr>
<tr>
<td>prod_rely</td>
<td>9.99</td>
<td>required software reliability</td>
</tr>
<tr>
<td>prod_data</td>
<td>9.99</td>
<td>data base size</td>
</tr>
<tr>
<td>prod_cplx</td>
<td>9.99</td>
<td>product complexity</td>
</tr>
<tr>
<td>comp_time</td>
<td>9.99</td>
<td>execution time constraint</td>
</tr>
<tr>
<td>comp_stor</td>
<td>9.99</td>
<td>main storage constraint</td>
</tr>
<tr>
<td>comp_virt</td>
<td>9.99</td>
<td>virtual machine volatility</td>
</tr>
<tr>
<td>comp_turn</td>
<td>9.99</td>
<td>computer turnaround time</td>
</tr>
<tr>
<td>pers_acap</td>
<td>9.99</td>
<td>analyst capability</td>
</tr>
<tr>
<td>pers_acexp</td>
<td>9.99</td>
<td>applications experience</td>
</tr>
<tr>
<td>pers_pcap</td>
<td>9.99</td>
<td>programmer capability</td>
</tr>
<tr>
<td>pers_vexp</td>
<td>9.99</td>
<td>virtual machine experience</td>
</tr>
<tr>
<td>pers_lexp</td>
<td>9.99</td>
<td>programming language experience</td>
</tr>
<tr>
<td>proj_modp</td>
<td>9.99</td>
<td>use of modern programming practices</td>
</tr>
<tr>
<td>proj_tool</td>
<td>9.99</td>
<td>use of modern tools</td>
</tr>
<tr>
<td>proj_scud</td>
<td>9.99</td>
<td>required development schedule</td>
</tr>
</tbody>
</table>

Figure III-2. CoCoMo Cost Driver Database Segment Layout
### COCOMO COST ESTIMATE DATA LAYOUT

<table>
<thead>
<tr>
<th>ID</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>proj_id</td>
<td>A899</td>
<td>/* project identification */</td>
</tr>
<tr>
<td>jul_date</td>
<td>YYDDD</td>
<td>/* Julian Date */</td>
</tr>
</tbody>
</table>
| mode     | AAA            | /* Embedded = emb                                                           */
|          |                | /* Semidetached = sem                                                        */
|          |                | /* Organic = org                                                            */
| tot_kdsi_sm | 9505.9         | /* total kdsi (thousands) - smallest                                        */
| tot_kdsi_ml | 9505.9         | /* total kdsi (thousands) - most likely                                     */
| tot_kdsi_lg | 9505.9         | /* total kdsi (thousands) - largest                                         */
| tot_kdsi_es | 9505.9         | /* total estimated source instructions                                      */
| tot_kdsi_sst2 | 9505.9        | /* software size source instructions - Bang                                  */
| tot_kdsi_sst2 | 9505.9         | /* software size source instructions - Token Count                          */
| bs_m_sm   | 9505.9         | /* basic nominal man-months - smallest                                       */
| bs_m_ml   | 9505.9         | /* basic nominal man-months - most likely                                   */
| bs_m_lg   | 9505.9         | /* basic nominal man-months - largest                                       */
| bs_m_es   | 9505.9         | /* basic nominal man-months - estimate                                      */
| bs_m_sst1 | 9505.9         | /* basic nominal man-months - SS Bang                                        */
| bs_m_sst2 | 9505.9         | /* basic nominal man-months - SS Token Count                                */
| int_m_sm  | 9505.9         | /* int. estimate (kdsi) - smallest                                          */
| int_m_ml  | 9505.9         | /* int. estimate (kdsi) - most likely                                       */
| int_m_lg  | 9505.9         | /* int. estimate (kdsi) - largest                                          */
| int_m_es  | 9505.9         | /* int. estimate (kdsi) - estimated                                        */
| int_m_sst1 | 9505.9        | /* int. estimate (kdsi) - SS Bang                                          */
| int_m_sst2 | 9505.9         | /* int. estimate (kdsi) - SS Token Count                                   */
| bs_tdev_sm | 9505.9         | /* basic schedule (mm) - smallest                                          */
| bs_tdev_ml | 9505.9         | /* basic schedule (mm) - most likely                                       */
| bs_tdev_lg | 9505.9         | /* basic schedule (mm) - largest                                          */
| bs_tdev_es | 9505.9         | /* basic schedule (mm) - estimated                                        */
| bs_tdev_sst1 | 9505.9     | /* basic schedule (mm) - SS Bang                                          */
| bs_tdev_sst2 | 9505.9        | /* basic schedule (mm) - SS Token Count                                    */
| int_tdev_sm | 9505.9         | /* int. dev. schedule (mm) - smallest                                      */
| int_tdev_ml | 9505.9         | /* int. dev. schedule (mm) - most likely                                   */
| int_tdev_lg | 9505.9         | /* int. dev. schedule (mm) - largest                                       */
| int_tdev_es | 9505.9         | /* int. dev. schedule (mm) - estimated                                     */
| int_tdev_sst1 | 9505.9       | /* int. dev. schedule (mm) - SS Bang                                       */
| int_tdev_sst2 | 9505.9        | /* int. dev. schedule (mm) - SS Token Count                                */
| bs_prod_sm | 9505.9         | /* basic productivity - small                                               */
| bs_prod_ml | 9505.9         | /* basic productivity - most likely                                        */
| bs_prod_lg | 9505.9         | /* basic productivity - largest                                             */
| bs_prod_es | 9505.9         | /* basic productivity - estimated                                           */
| bs_prod_sst1 | 9505.9        | /* basic productivity - SS Bang                                            */
| bs_prod_sst2 | 9505.9         | /* basic productivity - SS Token Count                                     */
| int_prod_sm | 9505.9         | /* int. productivity - small                                                */
| int_prod_ml | 9505.9         | /* int. productivity - most likely                                         */
| int_prod_lg | 9505.9         | /* int. productivity - largest                                              */
| int_prod_es | 9505.9         | /* int. productivity - estimated                                            */
| int_prod_sst1 | 9505.9        | /* int. productivity - SS Bang                                              */
| int_prod_sst2 | 9505.9         | /* int. productivity - SS Token Count                                      */
| bs_staff_sm | 9505.9         | /* basic average staffing - small                                           */
| bs_staff_ml | 9505.9         | /* basic average staffing - most likely                                     */
| bs_staff_lg | 9505.9         | /* basic average staffing - largest                                         */
| bs_staff_es | 9505.9         | /* basic average staffing - estimated                                       */
| bs_staff_sst1 | 9505.9        | /* basic average staffing - SS Bang                                        */
| bs_staff_sst2 | 9505.9         | /* basic average staffing - SS Token Count                                 */
| int_staff_sm | 9505.9         | /* int. average staffing - small                                            */
| int_staff_ml | 9505.9         | /* int. average staffing - most likely                                     */
| int_staff_lg | 9505.9         | /* int. average staffing - largest                                          */
| int_staff_es | 9505.9         | /* int. average staffing - estimated                                       */
| int_staff_sst1 | 9505.9        | /* int. average staffing - SS Bang                                         */
| int_staff_sst2 | 9505.9         | /* int. average staffing - SS Token Count                                 */

Figure III-3. CoCoMo Basic/Intermediate Database Segment Layout
<table>
<thead>
<tr>
<th>ID</th>
<th>VALUE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>proj_id</td>
<td>A999</td>
<td>project identification</td>
</tr>
<tr>
<td>proj_name</td>
<td>30A/N</td>
<td>project description - type</td>
</tr>
<tr>
<td>proj_type</td>
<td>AAA</td>
<td>type of project</td>
</tr>
<tr>
<td>jul_date</td>
<td>YYDDD</td>
<td>julian date</td>
</tr>
<tr>
<td>lang</td>
<td>AAA</td>
<td>principal programming language used</td>
</tr>
<tr>
<td>comments</td>
<td>AAA</td>
<td>percentage of code containing comments</td>
</tr>
<tr>
<td>comp_type</td>
<td>AAA</td>
<td>hardware project implemented on:</td>
</tr>
<tr>
<td>pers_cont</td>
<td>AAAA</td>
<td>stability of labor force:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>low/nom/high</td>
</tr>
</tbody>
</table>

*Figure III-4. General Project Database Segment Layout*
APPENDIX IV - SCE COMPUTATIONAL TABLES

//***********************************************************************
//
// SOFTWARE SIZING METRIC WEIGHTING TABLE
//
// Table Layout:
//  cc1-3  = Language abbreviation
//  cc5-10 = Metric Weighting Factor I
//  cc12-17 = Metric Weighting Factor II
//  cc19-22 = Language Expansion Ratio
//  cc24-80 = Description
//
//***********************************************************************

1  2  3  4  5  6  7
asm 8.825 0.588 1.0 Assembler
bas 8.825 0.588 1.5 Basic
c  8.825 0.588 1.2 C-language
cob 8.825 0.588 1.5 Cobol
pli 8.825 0.588 1.2 PL/I
pasc 8.825 0.588 1.2 Pascal
sh  8.825 0.588 1.5 Bourne-Shell

Figure IV-1. Britcher & Gaffney Software Sizing Measure Table
### DEVELOPMENT EFFORT MULTIPLIER TABLE

|--------|-------------|--------|--------|----------|-------------------|----------|-------------|----------|----------------|----------|--------------|----------|-------------------|----------|-------------------|----------|-------------|

**Figure IV-2. CoCoMo Development Effort Table**
MULTIPLIER AND EXPONENT TABLE

Table Layout:

<table>
<thead>
<tr>
<th>col-3</th>
<th>cc5-7</th>
<th>cc9-11</th>
<th>cc13-16</th>
<th>cc18-20</th>
<th>cc22-25</th>
<th>cc27-80</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoComo Model</td>
<td>Mode</td>
<td>Effort/Size Multiplier (MM)</td>
<td>Effort/Size Exponent (MM)</td>
<td>Schedule/Effort Multiplier (TDEV)</td>
<td>Schedule/Effort Exponent (TDEV)</td>
<td>Description</td>
</tr>
</tbody>
</table>

---

**Figure IV-3. CoComMo Multiplier & Exponent Table**
FUNCTIONAL PRIMITIVE SIZE CORRECTION TABLE

Table Layout:
- ccl-2 = Token Count (TCi)
- cc9-12 = Corrected Functional Primitive Increment (CFPI)

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>1.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>4.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>5.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>7.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>9.8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>12.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>14.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>16.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>19.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>21.5</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>24.1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>26.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>29.3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>16</td>
<td>32.0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>17</td>
<td>34.7</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>18</td>
<td>37.6</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>19</td>
<td>40.4</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>43.2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure IV-4. Bang Size Correction Table
**FUNCTIONAL COMPLEXITY WEIGHTING FACTOR TABLE**

Table Layout:
- cl1-2 = Class Code
- ccl9-11 = Weighting Factor
- ccl7-30 = Description

<table>
<thead>
<tr>
<th>SE</th>
<th>AM</th>
<th>DD</th>
<th>SU</th>
<th>SM</th>
<th>ED</th>
<th>VE</th>
<th>TM</th>
<th>SY</th>
<th>OG</th>
<th>DI</th>
<th>TA</th>
<th>AR</th>
<th>IN</th>
<th>CO</th>
<th>DM</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.6</td>
<td>0.3</td>
<td>0.5</td>
<td>1.0</td>
<td>0.8</td>
<td>1.0</td>
<td>1.0</td>
<td>1.5</td>
<td>1.0</td>
<td>1.8</td>
<td>1.0</td>
<td>0.7</td>
<td>1.0</td>
<td>2.0</td>
<td></td>
</tr>
</tbody>
</table>

**Figure IV-5. Bang Functional Complexity Table**
APPENDIX V - SCE SOURCE CODE

```
SCE=/usr/local/lib/scedb/active
echo "REQUIREMENTS SPECIFICATION SOFTWARE COST ESTIMATION TOOL [v1.0]"

juldate=`date +%Y%m%d`; regdate=`date +%Y%m%d`
echo "n
\$\{(regdate) \$(\{juldate\})\}n"
newidsw=NO; changemode=NO
#
# Capture and Validate Project ID No
#
while [ 1 ]
do
    echo "Enter Project ID (RETURN for new ID): \c"
    read idno
    if [ "! -r \$SCE/pidlist" ]; then
        \$SCE/pidlist
        chmod 777 \$SCE/pidlist
        chmod root \$SCE/pidlist
        chmod root \$SCE/pidlist
    fi
    if [ "$\{idno\}" = "!" ]; then
        cat \$\{(SCE)/pidlist
    else
        if [ "$\{idno\}" = "!" ]; then
            pid=`tail -1 \$\{(SCE)/pidlist`
            if [ "$\{pid\}" = "!" ]; then
                pid="a100"
            fi
            pid1=`echo \$\{pid\} | sed "s/\./\./g"
            pid2=`echo \$\{pid\} | sed "s/\./\./g"
            newpid2=`expr \$\{pid2\} + 1`
            idno="$\{pid\}$\{newpid2\}"
            echo " Your Project ID Number is: \$\{idno\} \$\{juldate\}/projinfo"
            doall=YES; loopcnt=1
            break
        else
            if [ ! -e \$\{(SCE)/pidlist` ]; then
                echo "Non-Existent Project ID entered!"
            else
                loopcnt=99
            fi
            LATEST=`ls -dt \$\{(SCE)/\$\{idno\}/[0-9]*) | tail -1`
```
PROJINFO="$LATEST/projinfo"

doall=NO

break

fi

done

# Capture and Validate: Project Title, Development Language,
# Percent Code Containing Comments,
# Personnel Continuity, Type of Computer,
# Type of Project, CoCoMo Development Mode,
# Development Effort Multipliers

# The 'while' loop establishes being able to selectively capture
# and validate a specific item.

while [ "$(loopcnt)" -gt 0 ]
do
  case "$(loopcnt)" in
    1)
      while [ 1 ]
      do
        echo "Enter Project Title\c"
        read title
        if [ "$title" = "" -o "echo "$title" wc -c" -gt 30 ]
        then
          echo "Project Title must be between 1 and 30 characters!"
        else
          if [ "$newidsw" = "YES" ]
          then
            loopcnt="expr \$(loopcnt) + 1"
          else
            >&$SCE/newpidlist
            exec < $SCE/pidlist
            while pidline="line"
            do
              if [ "`echo $pidline` | egrep -c "^$idno" = 0 ]
              then
                echo "$pidline" >&$SCE/newpidlist
              else
                echo "$idno - $title" >&$SCE/newpidlist
              fi
            done
            mv $SCE/newpidlist $SCE/pidlist
            exec < /dev/tty
          fi
        fi
      done
    *)
      echo "Invalid input. Please try again."
      break
  esac
done

break
fi
done
;

2)
while [ 1 ]
do
  echo "Enter Principal Development Language: \c"
  read lang
  if [ "$lang" = "?" ]
    then
      echo "\nlanguage choices are: as\m <assembler>, bas <basic>, c, cob <cobol>,"
      echo "for <fortran>, pas <pascal>, pli <pl/i>, sh <shell>"
    else
      case "$lang" in
        asm|bas|c|cob|for|pas|pli|sh)
          if [ "$newidsw" = "YES" ]
            then
              loopcnt=`expr "$loopcnt" + 1`
            fi
          break 1
        ;;
        esac
      fi
  done
;

3)
while [ 1 ]
do
  echo "Enter % Code to Contain Comments : \c"
  read comment
  if [ "$comment" = "?" ]
    then
      echo "\nValue must be numeric, range 0-99"
    else
      expr "$comment" + 1 >\dev/null 2>&\dev/null
      if [ $? -eq 0 -a "$comment" -ge 0 -a "$comment" -le 99 ]
        then
          if [ "$newidsw" = "YES" ]
            then
              loopcnt=`expr "$loopcnt" + 1`
            fi
          break
        else
          echo "\nInvalid Value Entered!"
4) while [ 1 ]
  do
    echo "Enter Personnel Continuity: \c"
    read contin
    if [ "$contin" = "?" ]
      then
        echo \nValues are: low - less than 10% turnover
        echo nom - nominal, 10-30% turnover
        echo hi - greater than 30% turnover
      else
        case "$contin" in
          low|nom|hi)
            if [ "$newidsw" = "YES" ]
              then
                loopcnt="expr \$\((\text{loopcnt}) + 1\)"
              fi
            break 1
            ;;
          *)
            echo \nInvalid Continuity Value!\n        esac
      fi
    done
  ;;
5) while [ 1 ]
  do
    echo "Enter Type of Development Computer: \c"
    read comtype
    if [ "$comtype" = "?" ]
      then
        echo \nValues are: max - Maxicomputer
        echo mid - Midicomputer
        echo min - Minicomputer
        echo mic - Microcomputer
      else
        case "$comtype" in
          mid|min|mic|max)
            if [ "$newidsw" = "YES" ]
              then
                loopcnt="expr $$(\text{loopcnt}) + 1\)"
            fi
          *)
        esac
    fi
  done
6) while [ 1 ]
   do
     echo "Enter Type of Project"
     read project
     if [ "$project" = "?" ]
       then
         echo \nValues are: bus - Business Application
         echo "ctl - Process Control"
         echo "hmi - Human Machine Interface"
         echo "sci - Scientific Application"
         echo "sup - Support Software"
         echo "ssy - Systems Software"
       else
         case "$project" in
           bus|ctl|hmi|sci|sup|ssy)
             if [ "$newidow" = "YES" ]
               then
                 loopcnt="expr \$(loopcnt) + 1"
             fi
             break 1
             ;;
           ;;
         esac;
       esac
     done
   ;;
7) while [ 1 ]
   do
     echo "Enter CoCoMo Mode (org/sem/emb)"
     read mode
     if [ "$mode" = "?" ]
       then
         echo \nValues are: org - Organic
         ;;
       else
         echo \nValues are: sem - Semi-Automatic
         echo "emb - Embryonic"
       esac
     done
   ;;
echo "sem - Semidetached"
else
    echo "emb - Embedded"

    case "${mode}" in
        org; sem; emb)
            if [ "$newidsw" = "YES" ]
                then
                    loopcnt="expr "$loopcnt" + 1"
                    fi
            break 1
        ;;
    esac
    echo \n"Invalid Model"
fi
done

3)
    echo \n"Enter Development Effort for Following 15 Multipliers"
while [ 1 ]
do
    echo \n"Single-Entry OR Global-Entry (s/g): \c"
    read entry
    case "${entry}" in
    g)
        while [ 1 ]
do
            echo \n"Use Values \"all[1-6]\" For All Multipliers Receiving Same Value: \c"
            read allvalue
            case "${allvalue}" in
                all1; all2; all3; all4; all5; all6)
                    rank="echo "${allvalue}" \| sed -e \"s/\.|$/\|/\"
                    rely=$((rank)); data=$((rank)); cplx=$((rank))
                    time=$((rank)); stor=$((rank)); virt=$((rank))
                    turn=$((rank)); acap=$((rank)); aexp=$((rank))
                    pcap=$((rank)); vexp=$((rank)); lexp=$((rank))
                    mod=$((rank)); tooi=$((rank)); sced=$((rank))
                    break 2
                ;;
            esac
            echo \n"Invalid Value Entered"
            esac
        done
    esac
s)
    echo \n"Use Values Range of 1-6 (1=low -- 6=high) For Each Multiplier"
cnt=1
while [ "$\{cnt\}" -lt 16 ]
do
case "$\{cnt\}" in
1)
  echo "Required Software Reliability"
  read rely
  tst="$\{rely\}"
  ;;
2)
  echo "Data Base Size"
  read data
  tst="$\{data\}"
  ;;
3)
  echo "Product Complexity"
  read cplx
  tst="$\{cplx\}"
  ;;
4)
  echo "Execution Time Constraint"
  read time
  tst="$\{time\}"
  ;;
5)
  echo "Main Storage Constraint"
  read stor
  tst="$\{stor\}"
  ;;
6)
  echo "Virtual Machine Volatility"
  read virt
  tst="$\{virt\}"
  ;;
7)
  echo "Computer Turnaround Time"
  read turn
  tst="$\{turn\}"
  ;;
8)
  echo "Analyst Capability"
  read acap
  tst="$\{acap\}"
  ;;
9)
  echo "Applications Experience"
  read aexp
  tst="$\{aexp\}"
  ;;
echo "Programmer Capability" ;
read pcap
tst="\$(pcap)"

; echo "Virtual Machine Experience" ;
read vexp
tst="\$(vexp)"

; echo "Programming Language Experience" ;
read lexp
tst="\$(lexp)"

; echo "Use of Modern Programming Practices" ;
read modp
tst="\$(modp)"

; echo "Use of Modern Tools" ;
read tool
tst="\$(tool)"

; echo "Required Development Schedule" ;
read sced
tst="\$(sced)"

esac
if [ \"\$(tst)\" = "?" ]
then
  echo "nValues are: 1=Very Low  2=Low"
  echo "  3=Nominal  4=High"
  echo "  5=Very High  6=Xtra High"
else
case \"\$(tst)\" in
  1|2|3|4|5|6
cnt="expr \"\$(cnt)\" + 1"
  )
    echo "nInvalid Value Entered!"
  esac
fi
done
break


```bash
    echo "\nInvalid Value Entered"

    esac
done
if [ "${newidsw}" = "YES" ]
then
    loopcnt=`expr "$\{loopcnt\}" + 1`
doall=NO
fi

99

esac

ans=""
while [ "${doall}" = "NO" -a "${ans}" != "y" -a "${ans}" != "n" ]
do

  # Process Profile Segment From Database
  #
  if [ "${changemode}" = "NO" ]
  then
    if [ "${newidsw}" = "YES" ]
    then
      echo "\nDo you wish to review project profile? (y/n): \\
      read ans
    else
      echo "Retrieving Profile Segment....\n"
    fi
  fi
  exec <PROJINFO
  recnt=1
  while rec="line"
    do
      linepart=`echo "$\{rec\}" | sed -e s/^............$/\n      case "$\{recnt\}" in
        1)
            title="${linepart}" 
            ;;
        2)
            lang="${linepart}" 
            ;;
        3)
            comment="${linepart}" 
            ;;
        4)
            contin="${linepart}" 
            ;;
        5)
```
comtype="@{linepart}";

7)
project="@{linepart}";

8)
mode="@{linepart}";

9)
set $rec
rely=$2; data=$3; cplx=$4
	time=$5; stor=$6; virt=$7
turn=$8; acap=$9; shift
	asep=$9; shift
	pcap=$9; shift
	vexp=$9; shift
	lexp=$9; shift
	modp=$9; shift
	tool=$9; shift
	sced=$9

10)
brack 1

esac
recont=`expr "${recont}" + 1`
done
eexec /dev/tty
ans="y"
fi
else
ans="y"
fi

# Display Entered Information OR Retrieved Profile Data
#
case "${ans}" in
  y)
    echo " Project ID " : ${idno}
    echo " Project Title " : ${title}
    echo " Development Language " : ${lang}
    echo " Code Containing Comments " : ${comment}
    echo " Personnel Continuity " : ${contin}
    echo " Type of Development Computer " : ${comtype}
    echo " Type of Project " : ${project}
    echo " CoCoMo Mode " : ${mode}
    echo " Development Effort Multipliers "
    echo " RELY: $rely  DATA: $data  CPLX: $cplx"
    echo " TIME: $time  STOR: $stor  VIRT: $virt"
```
echo "TURN: $turn  ACAP: $acap  AEXP: $aexp"
echo "PCAP: $pcap  VEXP: $vexp  LEXP: $lexp"
echo "MODP: $modp  TOOL: $tool  SCEV: $sced"
echo "X) No Changes - Data is Correct"
while [ 1 ]
do
  echo "Enter Response (1-8 or X): \c"
  read change
  case "$change" in
    1|2|3|4|5|6|7|8)
      loopcnt="$change"
      changemode="YES"
      break
    ;;
    X|x)
      loopcnt=0
      break 2
    ;;
  esac
  esac
  done
)
loopcnt=0
*)
  echo "Invalid Response"
 esac
done

```
```
break
else
  if [ `echo "$\{filename\}` | egrep -c "\.hir" -eq 1 ]
    process="hir"
    if [ ! -r "$\{filename\}" ]
      echo "\nEntered File Does Not Exist Or Is Not Readable!"
    else
      echo "\nProject Specification File Must Terminate With '.era' Or '.hir'"
    fi
  fi
else
  echo "\nProject Specification File Must Terminate With '.era' Or '.hir'"
fi
done

# Construct Shell 'Here-Document' For Background Processing
# cat <! /tmp/sce$$
COCOMOSCE="$SCE/$\{idno\}/$\{juldate\}/datafiles/cocomo_sce"
BANGSCE = "$SCE/$\{idno\}/$\{juldate\}/datafiles/bang_sce"
RPTFILE = "$SCE/$\{idno\}/$\{juldate\}/$\{idno\}.$\{juldate\}.rpt"
AUDITFILE="$SCE/$\{idno\}/$\{juldate\}/$\{idno\}.$\{juldate\}.aud"
compute() {
  bsans=`\bs <<<!
  \|
  ans=`\s echo "$\{bsans\} 1 sed -e "s/\(\.,\)\./\1/"`
}

# Update Valid Project ID List File
# if [ "$\{newidsw\}" = "YES" ]
#  echo "$\{idno\} - $\{title\}" >>$\{SCE\}/pidlist
fi

# Update Project Information Segment
# if [ ! -d "$SCE/$\{idno\}" ]
#  mkdir "$SCE/$\{idno\}"
#  chmod 0777 "$SCE/$\{idno\}"
#  chgrp root "$SCE/$\{idno\}"
#  chown root "$SCE/$\{idno\}"


if [ ! -d "$SCE/$(idno)/$(juldate)" ]
then
  mkdir "$SCE/$(idno)/$(juldate)"
  chmod 0777 "$SCE/$(idno)/$(juldate)"
  chgrp root "$SCE/$(idno)/$(juldate)"
  chown root "$SCE/$(idno)/$(juldate)"
fi
if [ ! -r "$PROJINFO" ]
then
  echo "$PROJINFO"
  chmod 0777 "$PROJINFO"
  chgrp root "$PROJINFO"
  chown root "$PROJINFO"
fi

echo "pid	$(idno)">>$PROJINFO
echo "title	$(title)">>$PROJINFO
echo "language	$(lang)">>$PROJINFO
echo "comments	$(comment)">>$PROJINFO
echo "contin	$(contin)">>$PROJINFO
echo "computer	$(comtype)">>$PROJINFO
echo "project	$(project)">>$PROJINFO
echo "mode	$(mode)">>$PROJINFO

localeconf=$(GETConf $CONF "($acap) $(data) $(cppix) $(time) $(stor) $(virt) $(turn) $(acap)"
localeconf2="$(aexp) $(pcap) $(vexp) $(lexp) $(modp) $(tool) $(sced)"
localeconf effmult "$localeconf1" "$localeconf2" >>$PROJINFO

## Use Input Language Type to Extract The Following Data From Software Size
## Metric Weighting Table:
##   Weighting Factor I - Using 'Bang' As Input
##   Weighting Factor II - Using Token Count As Input
## Language Expansion Ratio

tmpline = 'egrep "$((lang))" $SCE/tables/size_metric ; sed -e "s/ */ */g"
ssweight1='echo "$(tmpline)" | cut -d"" -f2"
ssweight2='echo "$(tmpline)" | cut -d"" -f3"
sslangexp='echo "$(tmpline)" | cut -d"" -f4"

## Use Input Cocomo mode to Extract 'Basic' and 'Intermediate' formula
## Exponents and Multiplier Values:
##   MM Effort/Size Multiplier
##   MM Effort/Size Exponent
##   TDEV Schedule/Perf Multiplier
##   TDEV Schedule/Perf Exponent

tmpline = 'egrep "$((mode))" $SCE/tables/cocomo_mult ; sed -e "s/ */ */g"
mmlmult='echo "$(tmpline)" | cut -d"" -f3"
# Use Input Development Effort Values To Extract Rating Multiplier
# From Table : Compute Effort Adjustment Factor ; Update CoCoMo Cost
# Driver Segment

if [ ! -d "$SCE/$(idno)/$(juldate)/datafiles" ]
then
  mkdir "$SCE/$(idno)/$(juldate)/datafiles"
  chmod 0777 "$SCE/$(idno)/$(juldate)/datafiles"
  chgrp root "$SCE/$(idno)/$(juldate)/datafiles"
  chown root "$SCE/$(idno)/$(juldate)/datafiles"
fi

COSTDRIVER="$SCE/$(idno)/$(juldate)/datafiles/cocomo_driver"

# Retrieve Cost Driver Factors From Stored Table

# $tmval = "expr $tval + 2"
relval = "expr $rel + 2"
dataval = "expr $data + 2"
cpiwval = "expr $cpiw + 2"
timeval = "expr $time + 2"
storval = "expr $stor + 2"
virtual = "expr $virtual + 2"
turnval = "expr $turn + 2"
acapval = "expr $acap + 2"
acexpval = "expr $acexp + 2"
ccapval = "expr $ccap + 2"
capval = "expr $cap + 2"
expval = "expr $exp + 2"
lexpval = "expr $lexp + 2"
tmpval = `expr $(modp) + 2`  
modpval = `grep " modp " $SCE/tables/cocomo_effort | sed -e "s/ */ */g" | cut -d" " -f$(tmpval)`

tmpval = `expr $(tool) + 2`  
toolval = `grep " tool " $SCE/tables/cocomo_effort | sed -e "s/ */ */g" | cut -d" " -f$(tmpval)`

tmpval = `expr $(sced) + 2`  
scedval = `grep " sced " $SCE/tables/cocomo_effort | sed -e "s/ */ */g" | cut -d" " -f$(tmpval)`

# Compute Effort Adjustment Factor
bc <<! >/*tmp/bc$*/
scale=2
a = $\{r\text{e\text{y\text{a\text{v\text{a\text{l}}}}}}\} \times \{\text{d\text{a\text{t\text{a\text{v\text{a\text{l}}}}}}\} \times \{\text{c\text{p\text{i\text{x\text{a\text{v\text{l}}}}}}\} \times \{\text{t\text{i\text{m\text{e\text{v\text{a\text{l}}}}}}\} \times \{\text{s\text{t\text{o\text{r\text{v\text{a\text{l}}}}}}\} \}$
b = $\{\text{v\text{i\text{r\text{t\text{u\text{n\text{v\text{a\text{l}}}}}}}}\} \times \{\text{t\text{u\text{n\text{v\text{a\text{l}}}}}}\} \times \{\text{a\text{c\text{a\text{p\text{a\text{v\text{a\text{l}}}}}}}}\} \times \{\text{a\text{e\text{x\text{p\text{v\text{a\text{l}}}}}}\} \times \{\text{p\text{c\text{a\text{p\text{v\text{a\text{l}}}}}}\}$
c = $\{\text{v\text{e\text{x\text{p\text{v\text{a\text{l}}}}}}\} \times \{\text{l\text{e\text{x\text{p\text{v\text{a\text{l}}}}}}\} \times \{\text{m\text{o\text{d\text{e\text{v\text{a\text{l}}}}}}\} \times \{\text{t\text{o\text{i\text{l\text{v\text{a\text{l}}}}}}\} \times \{\text{s\text{c\text{e\text{d\text{v\text{a\text{l}}}}}}\}$
a = b * c
quit
/
# Prepend Zero If Effort Adjustment Factor Begins With A Decimal Point
bcvar=`\cat /tmp/bc$`
if [ `echo \$bcvar` | sed -e "s/\^(.),/\^/\$/" -eq "." ]
then
bcvar="0$bcvar"
fi
if [ ! -r "$\text{COSTDRIVER}$" ]
then
\$\text{COSTDRIVER}
    chmod 0777 $\text{COSTDRIVER}
    chgrp root $\text{COSTDRIVER}
    chown root $\text{COSTDRIVER}
fi
#
# Update Cost Driver Segment In Database
#
echo "$\text{pid}\$$(\text{id\text{v\text{c}}})$" >>$\text{COSTDRIVER}
echo "$\text{juldate}\$$(\text{jul\text{date}})$" >>$\text{COSTDRIVER}
echo "$\text{pro\text{d\text{e\text{r\text{y\text{v\text{a\text{l}}}}}}}}\$$(\text{r\text{e\text{y\text{a\text{v\text{a\text{l}}}}}})}$" >>$\text{COSTDRIVER}
echo "$\text{pro\text{d\text{a\text{t\text{a\text{v\text{l}}}}}}\$$(\text{d\text{a\text{t\text{a\text{v\text{l}}}}}})$" >>$\text{COSTDRIVER}
echo "$\text{pro\text{d\text{c\text{i\text{x\text{a\text{v\text{l}}}}}}\$$(\text{c\text{p\text{i\text{x\text{a\text{v\text{l}}}}}})$" >>$\text{COSTDRIVER}
echo "$\text{com\text{p\text{t\text{i\text{m\text{e\text{v\text{a\text{l}}}}}}}}\$$(\text{t\text{i\text{m\text{e\text{v\text{a\text{l}}}}}})$" >>$\text{COSTDRIVER}
echo "$\text{com\text{p\text{t\text{u\text{n\text{v\text{a\text{l}}}}}}\$$(\text{t\text{u\text{n\text{v\text{a\text{l}}}}})$" >>$\text{COSTDRIVER}
echo "$\text{pers\text{a\text{c\text{a\text{p\text{a\text{v\text{a\text{l}}}}}}}}\$$(\text{a\text{c\text{a\text{p\text{a\text{v\text{a\text{l}}}}}}})$" >>$\text{COSTDRIVER}
echo "$\text{pers\text{a\text{x\text{p\text{v\text{a\text{l}}}}}}\$$(\text{a\text{e\text{x\text{p\text{v\text{a\text{l}}}}}})$" >>$\text{COSTDRIVER}
echo "$\text{pers\text{p\text{c\text{a\text{p\text{v\text{a\text{l}}}}}}}}\$$(\text{p\text{c\text{a\text{p\text{v\text{a\text{l}}}}}})$" >>$\text{COSTDRIVER}
echo "$\text{pers\text{x\text{e\text{p\text{v\text{a\text{l}}}}}}\$$(\text{v\text{e\text{x\text{p\text{v\text{a\text{l}}}}}})$" >>$\text{COSTDRIVER}
echo "$\text{pers\text{e\text{x\text{p\text{v\text{a\text{l}}}}}}\$$(\text{l\text{e\text{x\text{p\text{v\text{a\text{l}}}}}})$" >>$\text{COSTDRIVER}
echo "projmode \${modpval}" >>>$COSTDRIVER
echo "projtool \${toolval}" >>>$COSTDRIVER
echo "projsced \${scedval}" >>>$COSTDRIVER
echo "effadj \${bcvar}" >>>$COSTDRIVER
rm /tmp/bc$$

# Process E-R-A Input File
# Compute Intermediate Values
# Write To Audit Trail Report

firsttime ="YES"
multitoken="NO"
entitysw ="NO"
attribsw ="NO"
sourcesw ="NO"
complexsw ="NO"
severesw ="NO"
cumlo=0; cumml=0; cumhi=0
cumbang=0; cumpoken=0; tokencnt=0; entitycnt=0; sizecorrect=0
if [ ! -r "AUDITFILE" ]
then
    >&/AUDITFILE
    chmod 0777 /AUDITFILE
    chgrp root /AUDITFILE
    chown root /AUDITFILE
fi
echo "SOFTWARE COST ESTIMATION AUDIT TRAIL" PROJECT: \${idno} DATE: \${regdate} \${juldate}\n" >&/AUDITFILE
exec < \${filename}
while rec="line"
do
    recpart="echo "\${rec}" ; sed -e "s/\*\*\*/\*\*/g" ; sed -e "s/ */ */g" ; sed -e "s/ */ */g""
case "\${recpart}" in
    Activity;Activity;Periodic_Function;periodic_function; PROCEDURE:PROCEDURE:)
multitoken="NO"
if [ "\${entitysw}" = "NO" -a "\${attribsw}" = "YES" ]
then
    echo "**** ERROR **** FORMAT ERROR IN INPUT FILE - ENTITY DESCRIPTOR NOT FIRST RECORD\n" >&/AUDITFILE
    severesw="YES"
fi
if [ "\${firsttime}" = "NO" ]
then
    cumpoken="expr \${cumpoken} \{\$tokencnt\}
    sizecorrect="/\$\{sourcesw\}" = "YES" -a "\$\{complexsw\}" = "YES" ]
then
    compute "\${sizecorrect} * \${cplxweight}"
functbang=${ans}
compute "\${(cumbang) + \${functbang}}"

else
  sizecorrect=""
cplxpart ="
cplxweight ="
functbang ="
srcpart ="
fi
echo "Token Count : \${tokencnt} Total Token Count: \${(cumtoken)} Size Correction :\n \${(sizecorrect)} >>\$AUDITFILE\n"
echo "Complexity Code: \${(cplxpart)} Complexity Weight: \${(cplxweight)} Function Bang :\n \${(functbang)} >>\$AUDITFILE\n"
echo "Cumulative Bang: \${(cumbang)} Estimated Source : \${(srcpart)} Cumulative Source:\n \${(cumlo)}/\${(cumml)}/\${(cumhi)}\n" >>\$AUDITFILE

functbang=0
tokencnt=0
if [ "\${entitysw}" = "YES" -a "\${attribsw}" = "NO" ]
  then
    echo "***** ERROR ***** TOKEN COUNT IS 0\n" >>\$AUDITFILE
    severesw="YES"
  fi
if [ "\${sourcesw}" = "NO" ]
  then
    echo "***** ERROR ***** MISSING OR INVALID ESTIMATED SOURCE ATTRIBUTE\n" >>\$AUDITFILE
    severesw="YES"
  fi
if [ "\${complexsw}" = "NO" ]
  then
    echo "***** ERROR ***** MISSING OR INVALID COMPLEXITY FACTOR ATTRIBUTE\n" >>\$AUDITFILE
    severesw="YES"
  fi

entityline="`echo \${(rec)} | sed -e "s/^ \///\`"
echo "Function Name - \${(entityline)}" >>\$AUDITFILE
else
entityline="`echo \${(rec)} | sed -e "s/^ \///\`"
echo "Function Name - \${(entityline)}" >>\$AUDITFILE
  fi
time=0
entitysw ="YES"
attribsw ="NO"
sourcesw ="NO"
complexsw="NO"
entitycnt="`expr "\${(entitycnt)} + 1\`"
```
multitoken="YES"
if [ "$\{entitysw\}" = "YES" ]
then
  tokencnt="\`expr "$\{tokencnt\}" + 1"\`
  attribsw="YES"
fi

+Estimated_Source:+estimated_source:
multitoken="NO"
srcpart="\`echo "$\{rec\}" | sed -e "s/^
*: */\:\*:
*: /\/$\}\`"
sourcelo="\`echo "$\{srcpart\}" | cut -d"\n" -f1\`
sourcemel="\`echo "$\{srcpart\}" | cut -d"\n" -f2\`
sourcehi="\`echo "$\{srcpart\}" | cut -d"\n" -f3\`
expr "$\{sourcelo\}" + "$\{sourcemel\}" + "$\{sourcehi\}" > dev/null
2>/dev/null
if [ $\? -eq 0 -a "$\{sourcelo\}" -lt "$\{sourcemel\}" -a "$\{sourcemel\}" -lt "$\{sourcehi\}" ]
then
  sourcesw="YES"
  cumlo="\`expr "$\{cumlo\}" + "$\{sourcelo\}" -1\`
  cummel="\`expr "$\{cummel\}" + "$\{sourcemel\}" -1\`
  sourcehi="\`expr "$\{cumhi\}" + "$\{sourcehi\}" -1\`
else
  sourcesw="NO"
fi

+Complexity_Factor:+complexity_factor:
multitoken="NO"
cplxpart="\`echo "$\{rec\}" | sed -e "s/^
*: */\:\*:
*: /\/$\}\`"
cplxweight="\`egrep "^$\{cplxpart\}" $SCE/tables/bang_complex | cut -d"d" -f2\`
if [ "$\{cplxweight\}" == "" ]
then
  complexsw="NO"
else
  complexsw="YES"
fi

if [ "$\{process\}" = "hir" ]
then
  if [ "$\{multitoken\}" = "YES" ]
  then
    tokencnt="\`expr "$\{tokencnt\}" + 1"\`
  fi
fi

multitoken="NO"
done
Compute 'Bang' Formulations

```
cumtoken = \"\"expr \"\"( cumtoken ) \"\" + \"\"( tokencnt ) \"\"\nsizecorrect = \"\"egrep \"\"( tokencnt ) \"\" $SCE/tables/bang_size \"\" sed -e "s/ */ /g" ! cut -d -f2 \"\"
if [ \"\"( sourcesw ) \"\" = \"\"YES \"\" -a \"\"( complexsw ) \"\" = \"\"YES \"\" ]
then
  compute \"\"( sizecorrect ) \"\"( cplxweight ) \"\"( functbang ) \"\"( ans )
cumbang = \"\"( cumbang ) + \"\"( functbang ) \"\"( ans )
else
  sizecorrect = \"\"( cplxpart ) \"\"( cplxweight ) \"\"( functbang ) \"\"( srcpart )
fi
```

Token Count : $(tokencnt)  Total Token Count: $(cumtoken)  Size Correction : 
Complexity Code: $(cplxpart) Complexity Weight: $(cplxweight)  Function Bang : 
Cumulative Bang: $(cumbang)  Estimated Source : $(srcpart)  Cumulative Source:

```
if [ \"\"( entitysw ) \"\" = \"\"NO \"\" ]
then
  echo "\"\"( entitysw ) \"\" = \"\"NO \"\" > > \"\"$AUDITFILE
fi
```

```
if [ \"\"( entitysw ) \"\" = \"\"YES \"\" -a \"\"( attribsw ) \"\" = \"\"NO \"\" ]
then
  echo "\"\"( entitysw ) \"\" = \"\"NO \"\" > > \"\"$AUDITFILE
fi
```

```
if [ \"\"( sourcesw ) \"\" = \"\"NO \"\" ]
then
  echo "\"\"( sourcesw ) \"\" = \"\"NO \"\" > > \"\"$AUDITFILE
fi
```

```
if [ \"\"( complexsw ) \"\" = \"\"NO \"\" ]
then
  echo "\"\"( complexsw ) \"\" = \"\"NO \"\" > > \"\"$AUDITFILE
fi
```

```
if [ \"\"( severesw ) \"\" = \"\"YES \"\" ]
then
  echo "\"\"( severesw ) \"\" = \"\"YES \"\" > > \"\"$AUDITFILE
fi
```

```
if [ \"\"( entitycnt ) \"\" Entities Processed" > > \"\"$AUDITFILE
```

```
\n\nThere were a Total of $(entitycnt) Entities Processed" > > \"\"$AUDITFILE
```
cp \{filename\} $SCE/\{idno\}/\{juldate\}/\{idno\}.\{juldate\}.\{process\}
chmod 6777 $SCE/\{idno\}/\{juldate\}/\{idno\}.\{juldate\}.\{process\}
chgrp root $SCE/\{idno\}/\{juldate\}/\{idno\}.\{juldate\}.\{process\}
chown root $SCE/\{idno\}/\{juldate\}/\{idno\}.\{juldate\}.\{process\}

**Update CoCoMo Database Segment**
**Update Bang Database Segment**
**Update Report File Database Segment**

if [ "\$\{severesw\}" = "NO" ]
then
  compute "\$\{cumlo\} + (4 * \$\{cumml\}) + \$\{cumbh\} / 6"
  putnam="\$\{ans\}
  uncomment="\\exp 100 - \$\{comnt\}"
  compute "$\{aweight1\} \times \$\{slangexp\} \times \$\{uncomnt\} \times \$\{entitycnt\} \times \log(\$\{entitycnt\})"
  briterch1="\$\{ans\}
  compute "$\{sweight2\} \times \$\{sslangexp\} \times \$\{uncomnt\} \times \$\{cumbang\} \times \log(\$\{cumbang\})"
  briterch2="\$\{ans\}
  compute "$\{cumtoken\} / \$\{entitycnt\}"
  tcaqv="\$\{ans\}
  compute "$\{mm1mult\} \times (\$\{cumlo\} / 1000) \times \$\{mm1exp\}"
  bmm1="\$\{ans\}
  compute "$\{mm2mult\} \times (\$\{cumml\} / 1000) \times \$\{mm2exp\}"
  bmm2="\$\{ans\}
  compute "$\{mm3mult\} \times (\$\{cumbh\} / 1000) \times \$\{mm3exp\}"
  bmm3="\$\{ans\}
  compute "$\{mm4mult\} \times (\$\{uncomnt\} / 1000) \times \$\{mm4exp\}"
  bmm4="\$\{ans\}
  compute "$\{mm5mult\} \times (\$\{aweight1\} / 1000) \times \$\{mm5exp\}"
  bmm5="\$\{ans\}
  compute "$\{tdevmult\} \times \$\{bmm1\} \times \$\{tdevexp\}"
  btdv1="\$\{ans\}
  compute "$\{tdevmult\} \times \$\{bmm2\} \times \$\{tdevexp\}"
  btdv2="\$\{ans\}
  compute "$\{tdevmult\} \times \$\{bmm3\} \times \$\{tdevexp\}"
  btdv3="\$\{ans\}
  compute "$\{tdevmult\} \times \$\{bmm4\} \times \$\{tdevexp\}"
  btdv4="\$\{ans\}
  compute "$\{tdevmult\} \times \$\{bmm5\} \times \$\{tdevexp\}"
  btdv5="\$\{ans\}
  compute "$\{tdevmult\} \times \$\{bmm6\} \times \$\{tdevexp\} \times \$\{bcvar\}"
  btdv6="\$\{ans\}
  compute "$\{mm2mult\} \times (\$\{cumlo\} / 1000) \times \$\{mm2exp\} \times \$\{bcvar\}"
  imm1="\$\{ans\}
  compute "$\{mm2mult\} \times (\$\{cumml\} / 1000) \times \$\{mm2exp\} \times \$\{bcvar\}"
  imm2="\$\{ans\}
  compute "$\{mm2mult\} \times (\$\{cumbh\} / 1000) \times \$\{mm2exp\} \times \$\{bcvar\}"
  imm3="\$\{ans\}
  compute "$\{mm2mult\} \times (\$\{uncomnt\} / 1000) \times \$\{mm2exp\} \times \$\{bcvar\}"
  imm4="\$\{ans\}
  compute "$\{mm2mult\} \times (\$\{aweight1\} / 1000) \times \$\{mm2exp\} \times \$\{bcvar\}"
  imm5="\$\{ans\}
  compute "$\{tdevmult\} \times \$\{imm1\} \times \$\{tdevexp\}"
  jtdv1="\$\{ans\}
  compute "$\{tdevmult\} \times \$\{imm2\} \times \$\{tdevexp\}"
  jtdv2="\$\{ans\}
  compute "$\{tdevmult\} \times \$\{imm3\} \times \$\{tdevexp\}"
  jtdv3="\$\{ans\}
  compute "$\{tdevmult\} \times \$\{imm4\} \times \$\{tdevexp\}"
  jtdv4="\$\{ans\}
  compute "$\{tdevmult\} \times \$\{imm5\} \times \$\{tdevexp\}"
  jtdv5="\$\{ans\}
  compute "$\{tdevmult\} \times \$\{imm6\} \times \$\{tdevexp\}"
  jtdv6="\$\{ans\}
  compute "$\{cumlo\} / 1000 \times \$\{bmm1\}"
  bprod1="\$\{ans\}
  compute "$\{cumml\} / 1000 \times \$\{bmm2\}"
  bprod2="\$\{ans\}
  compute "$\{cumbh\} / 1000 \times \$\{bmm5\}"
  bprod3="\$\{ans\}
compute "({$bname})/1000"/"{$bname}"; bprod4="{$ans}"
compute "({$britcher1})/1000"/"{$bname}"; bprod5="{$ans}"
compute "({$britcher2})/1000"/"{$bname}"; bprod6="{$ans}"
compute "{($cumlo)}/1000"/"{$imm1}"; iprod1="{$ans}"
compute "{($cumml)}/1000"/"{$imm2}"; iprod2="{$ans}"
compute "{($cummh)}/1000"/"{$imm3}"; iprod3="{$ans}"
compute "{($putam)}/1000"/"{$imm4}"; iprod4="{$ans}"
compute "{($britcher1)}/1000"/"{$imm5}"; iprod5="{$ans}"
compute "{($britcher2)}/1000"/"{$imm6}"; iprod6="{$ans}"
compute "{($bmm)}/"{($bdev)}"; bstaff1="{$ans}"
compute "{($bmm2)}/"{($bdev2)}"; bstaff2="{$ans}"
compute "{($bmm3)}/"{($bdev3)}"; bstaff3="{$ans}"
compute "{($bmm4)}/"{($bdev4)}"; bstaff4="{$ans}"
compute "{($bmm5)}/"{($bdev5)}"; bstaff5="{$ans}"
compute "{($bmm6)}/"{($bdev6)}"; bstaff6="{$ans}"
compute "{($imm)}/"{($itdev1)}"; istaff1="{$ans}"
compute "{($imm2)}/"{($itdev2)}"; istaff2="{$ans}"
compute "{($imm3)}/"{($itdev3)}"; istaff3="{$ans}"
compute "{($imm4)}/"{($itdev4)}"; istaff4="{$ans}"
compute "{($imm5)}/"{($itdev5)}"; istaff5="{$ans}"
compute "{($imm6)}/"{($itdev6)}"; istaff6="{$ans}"
if [ ! -r "{$cocomosce}"
then
>/dev/0.777 "{$cocomosce"
chgrp root "{$cocomosce"
chown root "{$cocomosce"
fi
```
```
echo "btdevlo \$(btdev1)" >>$COCOMOSCE
echo "btdevl \$(btdev2)" >>$COCOMOSCE
echo "btdevhi \$(btdev3)" >>$COCOMOSCE
echo "btdevput \$(btdev4)" >>$COCOMOSCE
echo "btdevbg1 \$(btdev5)" >>$COCOMOSCE
echo "btdevbg2 \$(btdev6)" >>$COCOMOSCE
echo "itdevlo \$(itdev1)" >>$COCOMOSCE
echo "itdevl \$(itdev2)" >>$COCOMOSCE
echo "itdevhi \$(itdev3)" >>$COCOMOSCE
echo "itdevput \$(itdev4)" >>$COCOMOSCE
echo "itdevbg1 \$(itdev5)" >>$COCOMOSCE
echo "itdevbg2 \$(itdev6)" >>$COCOMOSCE
echo "bprodlo \$(bprod1)" >>$COCOMOSCE
echo "bprodhi \$(bprod2)" >>$COCOMOSCE
echo "bprodput \$(bprod3)" >>$COCOMOSCE
echo "bprodbg1 \$(bprod4)" >>$COCOMOSCE
echo "bprodbg2 \$(bprod5)" >>$COCOMOSCE
echo "iprodlo \$(iprod1)" >>$COCOMOSCE
echo "iprodhi \$(iprod2)" >>$COCOMOSCE
echo "iprodput \$(iprod3)" >>$COCOMOSCE
echo "iprodbg1 \$(iprod4)" >>$COCOMOSCE
echo "iprodbg2 \$(iprod5)" >>$COCOMOSCE
echo "bstaflo \$(bstaff1)" >>$COCOMOSCE
echo "bstafml \$(bstaff2)" >>$COCOMOSCE
echo "bstafhi \$(bstaff3)" >>$COCOMOSCE
echo "bstafput \$(bstaff4)" >>$COCOMOSCE
echo "bstafbg1 \$(bstaff5)" >>$COCOMOSCE
echo "bstafbg2 \$(bstaff6)" >>$COCOMOSCE
echo "istaflo \$(istaff1)" >>$COCOMOSCE
echo "istafml \$(istaff2)" >>$COCOMOSCE
echo "istafhi \$(istaff3)" >>$COCOMOSCE
echo "istafput \$(istaff4)" >>$COCOMOSCE
echo "istafbg1 \$(istaff5)" >>$COCOMOSCE
echo "istafbg2 \$(istaff6)" >>$COCOMOSCE

if [ ! -r "$BANGSCE" ]
then
  >&$BANGSCE
  chmod 0777 "$BANGSCE"
  chgrp root "$BANGSCE"
  chown root "$BANGSCE"
fi

echo "$pid $1dno" >&$BANGSCE
echo "$juldate $1dno" >&$BANGSCE
echo "$prmtot $1dno" >&$BANGSCE
echo "$tokentot $1dno" >&$BANGSCE
echo "$tcavg $1dno" >&$BANGSCE
echo "$projbang $1dno" >&$BANGSCE
if [ ! -r "$SRPTFILE" ]
then
  "SRPTFILE"
  chmod 0777 "$SRPTFILE"
  chgrp root "$SRPTFILE"
  chown root "$SRPTFILE"
fi

echo "SOFTWARE COST ESTIMATION REPORT
DATE: $(date) ($juldate)
PROJECT ID: $idno
PROJECT NAME: $(title)
COMPUTER TYPE: $(comtype)
CODMOD MODE: $(mode)
APPLICATION TYPE: $(project)
PERCENT COMMENTS: $(comment)
PERSONNEL CONTINUITY ESTIMATE: $(contin)
PRINCIPAL DEVELOPMENT LANGUAGE: $(lang)
"

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

echo "JULIAN
PROJECT BANG
DATE
"

echo "(RCD)
SMALL LIKELY LARGEST AVERAGE FSM BANG
ENTITY TOKENS AVERAGE BANG"

echo "$(juldate) $(cumlo) $(cumhi) $(putnam) $(britcher)
$(britcher2) $(entitycnt) $(cumtoken) $(tcavg) $(cumbang)"

skipit="NO"

# Extract and Report All Previous History For Project
for i in `ls -dt $SCE/$idno/[0-9]`
do
  if [ "$(i)" != "$SCE/$idno/$juldate" ]
  then
    BANGHIST="$i/datafiles/bang_sce"
    CODCHIST="$i/datafiles/cocoamo_sce"
    if [ ! -r ""$BANGHIST"
    then
      echo "Cannot Open "$BANGHIST" "$SRPTFILE"
      skipit="YES"
    fi
    if [ ! -r ""$CODCHIST"
    then
      fi
  done

# Extract and Report All Previous History For Project
for i in `ls -dt $SCE/$idno/[0-9]`
do
  if [ "$(i)" != "$SCE/$idno/$juldate" ]
  then
    BANGHIST="$i/datafiles/bang_sce"
    CODCHIST="$i/datafiles/cocoamo_sce"
    if [ ! -r ""$BANGHIST"
    then
      echo "Cannot Open "$BANGHIST" "$SRPTFILE"
      skipit="YES"
    fi
    if [ ! -r ""$CODCHIST"
    then
      fi
  done
```bash
# Extract and Report All Previous History For Project
for i in `ls -dt $SCE/$idno/[0-9]*`;
  do
    if [ ""$i"" ! = "$SCE/$idno/$juldate" ]
      then
        COCHIST="$i/datafiles/cocomo_sce";
        if [ ! -r "$COCHIST" ]
          then
            echo "Cannot Open $COCHIST" >> $RPTFILE
            skipit="YES"
          fi
        fi
```
then
  echo "Cannot Open \"COCHIST\" >>>RPTFILE
else
  julhist="\$juldate" \$COCHIST | sed -e "s/ / /g" | cut -d" " -f2
bmm1 ="\$\$(\$(bmm2) \$(bmm3) \$(bmm4) \$(bmm5) \$(bmm6) \$(bmm7))" \
  \$\$(\$(imm1) \$(imm2) \$(imm3) \$(imm4) \$(imm5) \$(imm6) \$(imm7))" >>>RPTFILE
fi

---

# Extract and Report All Previous History For Project
for i in \$\{10-9\}\n  do
    if [ "\$\{i\}" != "$\{SCE/\$(idno)/\$\{juldate\}" ]
      then
        COCHIST="\$\{i\}/datafiles/cocomo_sce"
    fi
  done
if [ ! -r "\$COCHIST" ]
then
echo "Cannot Open \$COCHIST" >> /RPTFILE
else
  julhist="\`egrep "\^juldate" \$COCHIST | sed -e "s/ */ */g" | cut -d"" -f2\`
  btdv1 = "\`egrep "btdv1" \$COCHIST | sed -e "s/ */ */g" | cut -d"" -f2\`
  btdv2 = "\`egrep "btdv2" \$COCHIST | sed -e "s/ */ */g" | cut -d"" -f2\`
  btdv3 = "\`egrep "btdv3" \$COCHIST | sed -e "s/ */ */g" | cut -d"" -f2\`
  btdv4 = "\`egrep "btdv4" \$COCHIST | sed -e "s/ */ */g" | cut -d"" -f2\`
  btdv5 = "\`egrep "btdv5" \$COCHIST | sed -e "s/ */ */g" | cut -d"" -f2\`
  itdev1 = "\`egrep "itdev1" \$COCHIST | sed -e "s/ */ */g" | cut -d"" -f2\`
  itdev2 = "\`egrep "itdev2" \$COCHIST | sed -e "s/ */ */g" | cut -d"" -f2\`
  itdev3 = "\`egrep "itdev3" \$COCHIST | sed -e "s/ */ */g" | cut -d"" -f2\`
  itdev4 = "\`egrep "itdev4" \$COCHIST | sed -e "s/ */ */g" | cut -d"" -f2\`
  itdev5 = "\`egrep "itdev5" \$COCHIST | sed -e "s/ */ */g" | cut -d"" -f2\`

  echo "" "(julhist) \$
  \$ btdv2 \$ btdv5 \$ itdev5 \$ itdev1\$
  \$ btdv1 \$ btdv3 \$ itdev4 \$ itdev2 \$ itdev3 \$ itdev5 \$
" >> /RPTFILE
fi

done
echo "---------------------------------------------------------------------\n\n" >> /RPTFILE
echo "\nJULIAN \nY (KDSI/PM) \n\n" >> /RPTFILE
echo "\nDATE \n\n" >> /RPTFILE
echo "INTERMEDIATE COCOMO \n\n" >> /RPTFILE
echo "\n(RCD) \n\n" >> /RPTFILE
echo "\nSMALL LIKELY LARGEST AVERAGE FSM BANG \n\n" >> /RPTFILE
echo "\n\n" >> /RPTFILE

echo "\$juldate\$
\$bprod1\$ \$bprod2\$ \$bprod3\$ \$bprod4\$ \$bprod5\$
\$iprod1\$ \$iprod2\$ \$iprod3\$ \$iprod4\$ \$iprod5\$
" >> /RPTFILE

# Extract and Report All Previous History For Project#
for i in \`ls -dt $SCE/$idno/[0-9]X``
do
  if [ "$i" != "$SCE/$idno/" ]
  then
  fi
done
COCHIST="\$/datafiles/cocomo_sce"
if [ ! -r ""COCHIST"
then
echo "Cannot Open "$COCHIST" >>>RPTFILE
else
  julhist="\$\{juldate\}"
  bprod1 =\$\{bprod1\} \$\{bprod2\} \$\{bprod3\} \$\{bprod4\} \$\{bprod5\} \$\{bprod6\}
  bprod2 =\$\{bprod2\} \$\{bprod3\} \$\{bprod4\} \$\{bprod5\} \$\{bprod6\}
  bprod3 =\$\{bprod3\} \$\{bprod4\} \$\{bprod5\} \$\{bprod6\}
  bprod4 =\$\{bprod4\} \$\{bprod5\} \$\{bprod6\}
  bprod5 =\$\{bprod5\} \$\{bprod6\}
  bprod6 =\$\{bprod6\}
  iprod1 =\$\{iprod1\} \$\{iprod2\} \$\{iprod3\} \$\{iprod4\} \$\{iprod5\} \$\{iprod6\}
  iprod2 =\$\{iprod2\} \$\{iprod3\} \$\{iprod4\} \$\{iprod5\} \$\{iprod6\}
  iprod3 =\$\{iprod3\} \$\{iprod4\} \$\{iprod5\} \$\{iprod6\}
  iprod4 =\$\{iprod4\} \$\{iprod5\} \$\{iprod6\}
  iprod5 =\$\{iprod5\} \$\{iprod6\}
  iprod6 =\$\{iprod6\}
  jecho "\$\{julhist\} \$\{bprod1\} \$\{bprod2\} \$\{bprod3\} \$\{bprod4\} \$\{bprod5\} \$\{bprod6\} \$\{iprod1\} \$\{iprod2\} \$\{iprod3\} \$\{iprod4\} \$\{iprod5\} \$\{iprod6\}" >>>RPTFILE
fi
done
echo "---------------------------------------------" >>RPTFILE
echo "!!! JULIAN !!! AVERAGE STAN FFING (FSP) !!!" >>RPTFILE
echo "!!! DATE !!! BASIC COCOMO ------" >>RPTFILE
echo "!!! (RCD) !!! SMALL LIKELY LARGE LIKELY AVERAGE FSM FMS FMS BANG !!!" >>RPTFILE

# Extract and Report All Previous History For Project
for i in \$\{idno\}/[0-9]*
do
  if [ "$\{i\}" != "$\{idno\}/\$\{juldate\}" ]
  do
    echo "---------------------------------------------" >>RPTFILE
    \$\{bstaff1\} \$\{bstaff2\} \$\{bstaff3\} \$\{bstaff4\} \$\{bstaff5\} \$\{bstaff6\}
    \$\{istaff1\} \$\{istaff2\} \$\{istaff3\} \$\{istaff4\} \$\{istaff5\} \$\{istaff6\}" >>RPTFILE
  done
done
then
  CDCHST="/datafiles/cocomo_sce"
  if [ ! -r "$CDCHST" ]
    then
echo "Cannot Open "$CDCHST" >>$SRPTFILE
  else
    julhist="egrep "^juldate" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    bstaff1="egrep "^bstaff1" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    bstaff2="egrep "^bstaff2" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    bstaff3="egrep "^bstaff3" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    bstaff4="egrep "^bstaff4" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    bstaff5="egrep "^bstaff5" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    bstaff6="egrep "^bstaff6" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    istaff1="egrep "^istaff1" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    istaff2="egrep "^istaff2" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    istaff3="egrep "^istaff3" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    istaff4="egrep "^istaff4" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    istaff5="egrep "^istaff5" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    istaff6="egrep "^istaff6" "$CDCHST" | sed -e "s/ / /g" | cut -d"n" -f2"
    echo """
    |
    | $(julhist)
    |
    | $(bstaff1) $(bstaff2) $(bstaff3) $(bstaff4) $(bstaff5) $(bstaff6) 
    | $(istaff1) $(istaff2) $(istaff3) $(istaff4) $(istaff5) $(istaff6) ""
    >>$SRPTFILE
fi
fi
done
echo "------------------------------------------------------------------------" >>$SRPTFILE
rc=0
else
rc=1
fi
mail "logname" <<<

SOFTWARE COST ESTIMATION PROCESSING COMPLETE

PROJECT: $(idno)/$(juldate) RETURN CODE: $(rc)

Check The '/tmp/' Files For Error Messages

nohup sh /tmp/sce$$ >/tmp/sce1$$ 2>/tmp/sce2$$ &
echo "\nYour Cost Estimation Data Has Been Accepted For Further Processing"
echo "The Estimation Results For Project "$idno" Will Be Placed In"
echo "$idno"$(juldate)/n"
echo "Cost Estimation Report File: "$idno"$(juldate).rpt"
echo "Project Specification Audit Trail: "$idno"$(juldate).aud"
echo "Project Specification Input File: "$idno"$(juldate).$(process)"
echo "Notification To "$logname" Will Be Sent When Processing Is Complete"
REFERENCES


A REQUIREMENTS SPECIFICATION SOFTWARE COST ESTIMATION TOOL

by

GARY DAVID SCHNEIDER

B.S.  Long Island University, 1974
M.B.A.  Long Island University, 1975

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

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the
requirements for the degree

MASTER OF SCIENCE

Department of Computer Science

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1986
As projects become increasingly larger and more complex, cost estimation becomes more important in planning for schedule and resource requirements. It is clear that the cost of software development has increased to the point that software represents the largest component of total system cost. Being able to accurately predict resource requirements and development effort serves as an effective planning aid for management.

This project merges the concepts of Boehm, Putnam, Britcher & Gafney, and DeMarco to construct a Cost Estimation paradigm based on a software requirement specification. The input is an E-R specification using established standards and includes additional attribute definitions. The tool is designed to perform interactively for generating basic and intermediate CoCoMo results using Boehm's model modified with concepts by Putnam. In addition, Britcher & Gaffney's measure is used with DeMarco's Bang formulations to produce a SLOC estimate. This result is used as input to the CoCoMo model with all generated results produced in report format. A historical perspective covering many projects and time frames is maintained through a database application. A keystone to this effort correlates the highly refined CoCoMo model normally used in later stages of the development process with the general sizing 'Bang' formulations used during the specification phase. The Britcher & Gaffney measure is used to bridge these two models.