NASA-CR-189293

SOFTWARE ENGINEERING LABORATORY SERIES SEL-89-103

# SOFTWARE MANAGEMENT ENVIRONMENT (SME)

# CONCEPTS AND ARCHITECTURE

# REVISION 1

SEPTEMBER 1992

(NASA-CR-189293) SOFTWARE MANAGEMENT ENVIRONMENT (SME) CONCEPTS AND ARCHITECTURE, REVISION 1 (NASA. Goddard Space Flight Center) 84 p

N94-29727

Unclas

G3/61 0004479

NASA

National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

[illegible]

SOFTWARE ENGINEERING LABORATORY SERIES SEL-89-103

# SOFTWARE MANAGEMENT ENVIRONMENT (SME)

# CONCEPTS AND ARCHITECTURE

# REVISION 1

SEPTEMBER 1992

National Aeronautics and
Space Administration

**Goddard Space Flight Center**
Greenbelt, Maryland 20771

# FOREWORD

The Software Engineering Laboratory (SEL) is an organization sponsored by the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC) and created to investigate the effectiveness of software engineering technologies when applied to the development of applications software. The SEL was created in 1976 and has three primary organizational members:

NASA/GSFC, Software Engineering Branch

University of Maryland, Department of Computer Science

Computer Sciences Corporation, Software Engineering Operation

The goals of the SEL are (1) to understand the software development process in the GSFC environment; (2) to measure the effect of various methodologies, tools, and models on this process; and (3) to identify and then to apply successful development practices. The activities, findings, and recommendations of the SEL are recorded in the Software Engineering Laboratory Series, a continuing series of reports that includes this document.

The original version of the *Software Management Environment (SME) Concepts and Architecture* was published in August 1989. This new edition contains updated material and constitutes a major revision to the 1989 version.

The major contributors to the original document are

William Decker (Computer Sciences Corporation)
Jon Valett (NASA/GSFC)

The major contributors to this version are

Robert Hendrick (Computer Sciences Corporation)
David Kistler (Computer Sciences Corporation)
Jon Valett (NASA/GSFC)

Single copies of this document can be obtained by writing to

Software Engineering Branch
Code 552
Goddard Space Flight Center
Greenbelt, Maryland 20771

[illegible]

## ABSTRACT

This document presents the concepts and architecture of the Software Management Environment (SME), developed for the Software Engineering Branch (Code 552) of the Flight Dynamics Division (FDD) of the Goddard Space Flight Center (GSFC). The SME provides an integrated set of experience-based management tools that can assist software development managers in managing and planning flight dynamics software development projects. This document provides a high-level description of the types of information required to implement such an automated management tool, and it presents an architectural framework in which a set of management services can be provided.

This document is a major revision of SEL-89-003.

[illegible]

## Table of Contents

PRECEDING PAGE BLANK NOT FILMED

10001966

PAGE VI INTENTIONALLY BLANK

## Table of Contents (Cont'd)

# List of Illustrations

## List of Illustrations (Cont'd)

## SECTION 1—INTRODUCTION

The Software Management Environment (SME) is an automated management tool being developed under the sponsorship of the Software Engineering Laboratory (SEL) at the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC). The tool is intended to assist managers of software development projects in the GSFC flight dynamics environment with their day-to-day management and planning activities.

The SME is unique in that it incorporates the organizational experience gained from past development projects—the data, the research results, and the knowledge of experienced software managers—and makes that experience available to managers of current projects. The SME does this through an integrated set of graphically oriented features that enable the software manager to compare an ongoing development effort with previous efforts and with models of typical projects in the environment, to predict future project status, to analyze a project's strengths and weaknesses, to examine "what if" scenarios by varying a project's plan, and to assess the project's quality relative to previous efforts.

The experience encapsulated and packaged within the SME takes the form of software process and product measures, relationships that characterize the local GSFC environment, models of how various measures can be expected to behave in the environment, and management rules of thumb that capture the conventional wisdom of experienced managers in the environment.

This concept of packaging experience underlies one of the major activities in which the SEL engages. Established in 1976, the SEL performs software engineering research within the context of the GSFC flight dynamics application environment (Reference 1). Its organizational members include NASA/GSFC, Computer Sciences Corporation, and the University of Maryland. The SEL's goals are to support continual process improvement by characterizing and understanding the development process and, in a controlled, iterative fashion, introducing improvements into that process and measuring their impact.

To achieve these goals, the SEL measures the development process and products, analyzes the measurement data to characterize and understand the environment, and performs experiments to determine the impact and feasibility of introducing new technologies or methodologies into the process. Once a candidate improvement has been deemed beneficial, it is tailored for the local environment and packaged for insertion into the standard processes employed by software developers and managers. These activities map directly to the stages of a theoretical framework for modeling software process improvement known as the experience factory (Reference 2).

The idea of developing the SME to package the SEL's experience base in an integrated tool for managers originated in 1984. From 1984 through 1987, the basic concepts of the tool and the architecture to support them were investigated and refined by a series of prototypes. In late 1986, these initial efforts were thoroughly analyzed and requirements were developed for a more complete software system (Reference 3). Based on an analysis of these prototypes, work began on the current version of the SME. Subsequently, this version became a testbed for exploring the feasibility of implementing the additional management functions envisioned in the initial concept for the tool.

Now that the tool has matured to the level where most of the original concepts have been implemented, hence proven feasible, the SME can serve as a model for the development of similar tools in other environments. Although the SME as implemented by the SEL reflects measures, models, and rules that apply specifically to the GSFC flight dynamics environment, the underlying concepts can be exported to other software development organizations that wish to build similar tools for their environment.

This document is intended to introduce the SME to individuals and organizations interested in understanding or in implementing a measurement-oriented, integrated management tool that is based on local experience. It describes the concepts behind the major components of the SME and outlines the architecture of the system as it has been implemented in the SEL. The remainder of the document is organized as follows:

- Section 2 describes a model of management activities on which the SME is based, introduces SEL data that can support those activities, and summarizes the functions that the SME provides.
- Section 3 presents the architecture of the SME and describes the structure of the data, functions, and hardware.

# SECTION 2—CONCEPTS

This section presents the high-level concepts required for a basic understanding of the SME. The concepts discussed below relate to three major areas: a set of key management activities that the SME must address to be an effective management tool, the data available in the SEL environment that a comprehensive SME can use as an experience base to support these key activities, and an overview of the SME functions designed to help managers perform those activities. A concept summary provided at the end of this section illustrates (1) how information is used in the SME and (2) how the manager's activities are mapped into SME functions.

## 2.1 MANAGEMENT ACTIVITIES

The SME operates under the assumption of a certain pattern, or model, of the way managers do things. The model reflects a software development project managed in a well-defined management environment. The activities described below are carried out to varying levels of detail, depending on factors such as the size of the project, its criticality, or its current status.

*Observation:* The manager monitors the progress of the project by tracking several key dynamic parameters (such as weekly effort, lines of code, or software changes) and combinations of those parameters (such as lines of code per hour or reported errors per line of code). The parameters tracked by the manager typically reflect performance. This is the foundation on which the other activities are based.

*Comparison:* The manager uses archived data from completed projects (or nominal performance guidelines) as references to judge the progress and health of the current project.

*Prediction:* The manager extrapolates from the current project status toward project completion to estimate schedules, costs at completion, product size, and other parameters of interest.

*Analysis:* The manager examines the observations and applies subjective information about the project to identify the probable causes of any deviations from nominal performance guidelines.

*Assessment:* The manager weighs all the information about the project to form a judgment of the project's quality and productivity.

*Planning:* The manager reevaluates and modifies the project plan as needed. This involves periodically updating or refining the current project schedule and estimates during the development life cycle.

*Control:* The manager decides on a course of action and modifies the activities occurring on the project. This involves initiating corrective actions in response to any recognized problems and taking steps to improve or enhance the development process.

## 2.2 THE SEL ENVIRONMENT AND THE SME

The SME integrates the experience gained from past projects with current measurement activities to provide the manager with a wide variety of information for monitoring and controlling an ongoing software project. The information required to provide this functionality can be divided into three major components: a repository of data collected from software development projects, research results from studies of the software development process, and management rules for software development. The availability of these data in the SEL makes a comprehensive SME possible.

### 2.2.1 The SEL Database

One underlying assumption of the SME is the existence of an organized, consistent process for collecting software development data and storing those data for subsequent use. In this environment, the SEL database serves as a central repository of information on ongoing, as well as completed, projects (References 4 and 5). The establishment of such a repository forms the foundation for all measurement and improvement activities. The SEL database, which has evolved into its current form over the years of its existence, provides the SME with the raw data required to observe a project's behavior.

The major items of data provided to the SME by the database are SEL measurements of software parameters that are of interest to the software manager. These include such parameters as the following:

- Effort data (technical, management)
- Computer resource usage [central processing unit (CPU), jobs]
- Software changes
- Software errors
- Product size [lines of code (LOC), modules]

Many of the other data items needed by the SME are also acquired from the SEL database. These include objective data, such as the project's application or the languages and tools used; subjective data, which evaluate projects on a series of software methodology questions; and planning data (schedules and estimates), which are supplied to the SEL by the manager.

These items, and others, are available for currently active projects and for the past projects that a manager may want to use as a basis for comparison.

### 2.2.2 SEL Research Results

A second major component of the SME is the research results from studies of the software development process in the SEL environment. Information derived from studies developed through experimentation and through analysis of the SEL database is a key part of the SME (Reference 6). The SME incorporates these research results via specific models and relationships. Based on a comprehensive understanding of the software development environment, these models and relationships are used by the SME to enable the manager to better understand how a particular project compares to the typical project within the environment and to aid in predicting and estimating future conditions on the software project.

A model describes the expenditure, utilization, or production of a software development parameter as a function of time and can represent a guideline for managers to follow while planning or observing a project. For example, a model of the staffing profile would capture the typical expenditure of effort over the entire software development life cycle (Reference 7). Another example of a useful guideline would be a model of the growth of source code during a project.

A relationship describes the correlation between two or more software development parameters at a specific point in time. Typically used in estimation, these relationships can help managers estimate project completion values based on the known or estimated values of other measures. For example, an equation that expresses total staff hours as a function of lines of code may be used to estimate the effort required for a project of a given size.

Other SEL research results capture data on the known effects of specific software development methodologies. For example, one result states that code reading is the most effective method for finding errors in this environment (Reference 8). These results help identify and verify key software development rules that can augment management experience.

### 2.2.3 SEL Management Experience

A final major component of the SME is a set of software development rules. The SME attempts to integrate the experience of software managers into an expert system concept to provide the ability to analyze project measures and status. Previously, this experience was captured only in "lessons learned" or summary documents. The SME formalizes this knowledge into a basic structure that will continually evolve as the experience and knowledge are validated. By automating the knowledge utilization into an expert system, the SME gives the manager the ability to analyze current projects by applying past experience. The incorporation of this concept into the SME is a difficult area of research; however, the basic concept of utilizing expert systems for software management was proved feasible by SEL research (References 9 and 10).

The experienced manager's knowledge can be used in many areas within the SME. Specific rules for software development have been collected from interviews with numerous managers and from extensive research into the causes and effects of observed deviations in the software development process. One such rule states the following:

If the reported error count is lower than normal, possible causes are

- Insufficient testing
- Experienced team
- Problem less difficult than expected

When applied with a set of similar rules to the measures and status of the software project, the SME can reach conclusions pertaining to deviating trends in project measures, such as a higher (or lower) than normal count of errors. The analysis can also be extended to the overall project to diagnose project problems and suggest actions for correcting those problems.

## 2.3 SME FUNCTIONS

The functionality of the SME extends to each area of a manager's activities. A brief description of the way the SME addresses each area of activity is given below. Detailed descriptions of these processes and the data required to support them are presented in Section 3.

### 2.3.1 Observation

The observation function displays project data for measures of interest such as effort, LOC, or CPU utilization. The cumulative plot of a particular measure is maintained as a backdrop against which other SME functions display their results. This function gives the manager an overall view of how each measure is growing: it reveals trends and emphasizes the importance of estimates of completion values. Figure 2-1 shows the SME display after the manager selects a project and measure of interest. The observation function makes direct use of the regularly collected SEL data.

The observation function can also display project data for the ratio of any two such measures. This allows managers to view an extended set of measures such as LOC per hour (coding productivity) and reported errors per LOC (error density). Using a ratio of two measures as the measure of interest facilitates the examination of multiple projects by minimizing the effects of project size.

### 2.3.2 Comparison

The comparison function displays either archived data from the SEL database for completed projects or guidelines derived from models of the selected measure for a normal project. By overlaying comparison data on top of the observational plot, this function gives the manager the information needed to judge a project's behavior in the context of a specific previous project or of a "typical" project. Figure 2-2 shows a comparison of the number of errors on a current project with the number of errors on a past project. Figure 2-3 shows a comparison of the number of errors on a current project with the development environment's guidelines for error count growth.

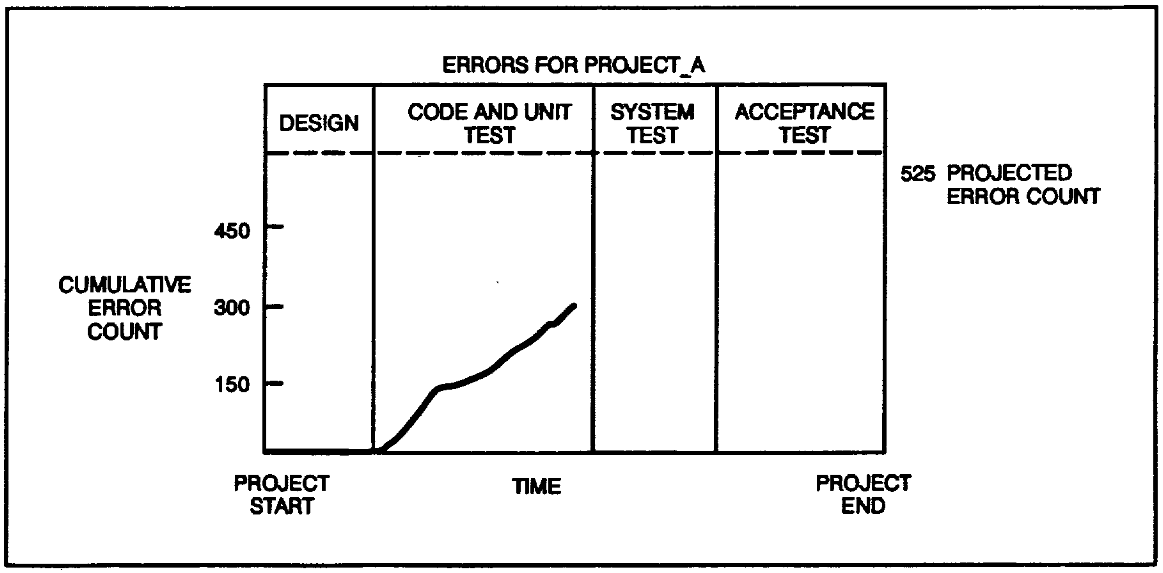


**Figure 2-1. Observation Display**

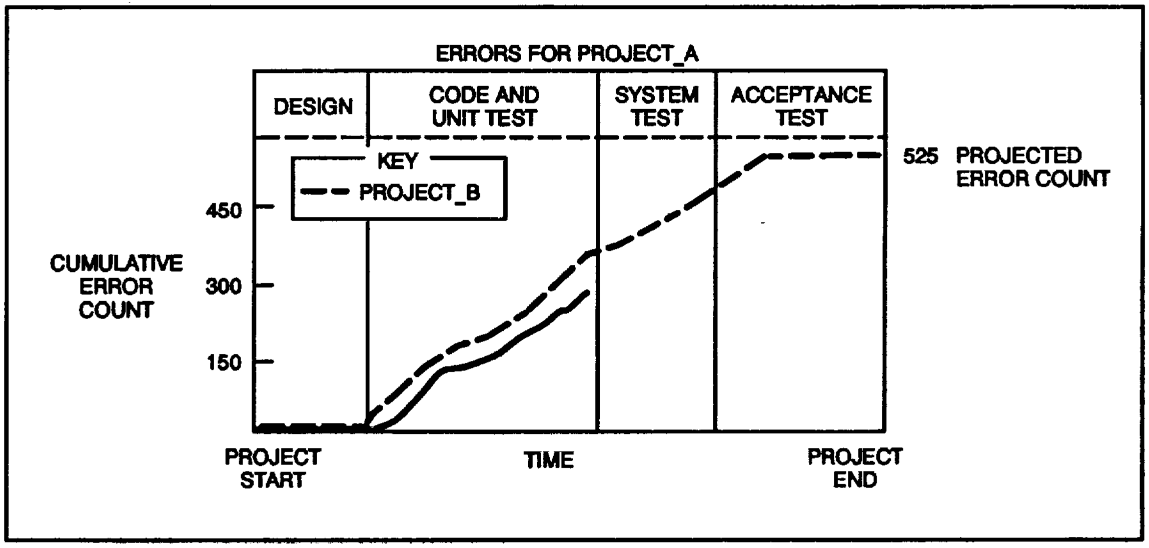


**Figure 2-2. Comparison Display—Completed Project**

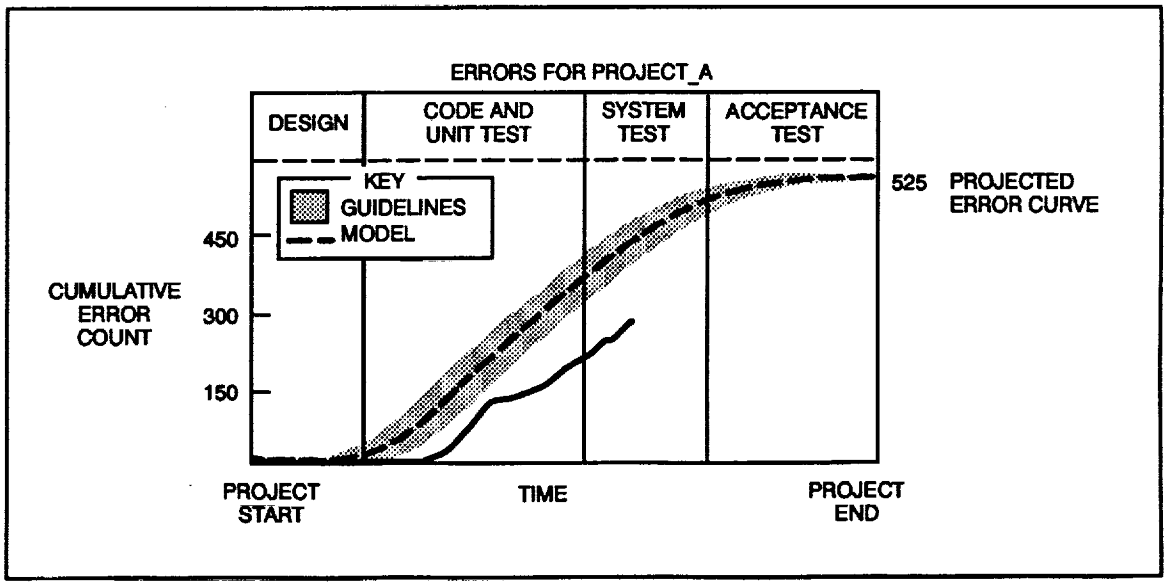


**Figure 2-3. Comparison Display—Model/Guidelines**

### 2.3.3 Prediction

The prediction function produces a completion estimate for a measure of interest based on models of the measure's typical behavior in the environment. This function extends the observational plot of the measure's growth to project completion, giving the manager a view of the project's probable future behavior. For example, knowing the current phase and count of errors for a project enables the SME to use an error growth model to predict the final error count for the system (Figure 2-4). These estimates of completion values are invaluable to the manager in planning and controlling a software project.

### 2.3.4 Analysis

The analysis function helps managers analyze the current value for the measure of interest using two distinct methods—trend analysis and profile analysis.

Trend analysis compares the current value of a selected measure to the model of the measure and reaches conclusions that help explain any deviations from the norm. Although the analysis is focused on the measure of interest, the function interprets a wide range of current project data according to the reasoning captured in a set of management rules as a basis for reaching any conclusion. The reasons appear against the backdrop of the cumulative plot for the deviating measure. Figure 2-5 shows the display of a list of probable causes for a lower-than-expected trend in the growth of errors.

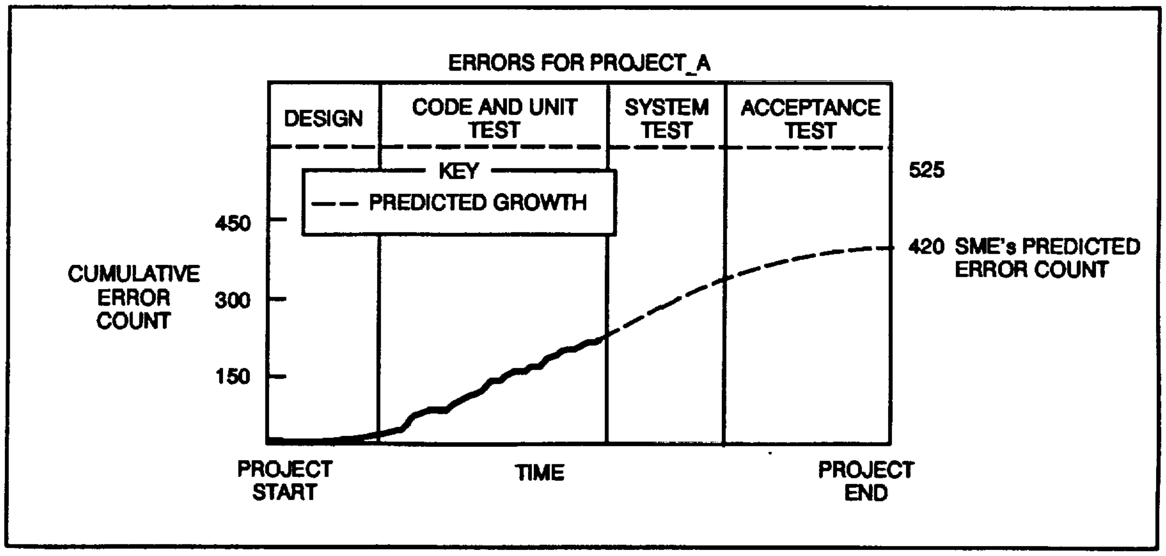


**Figure 2-4. Prediction Display**

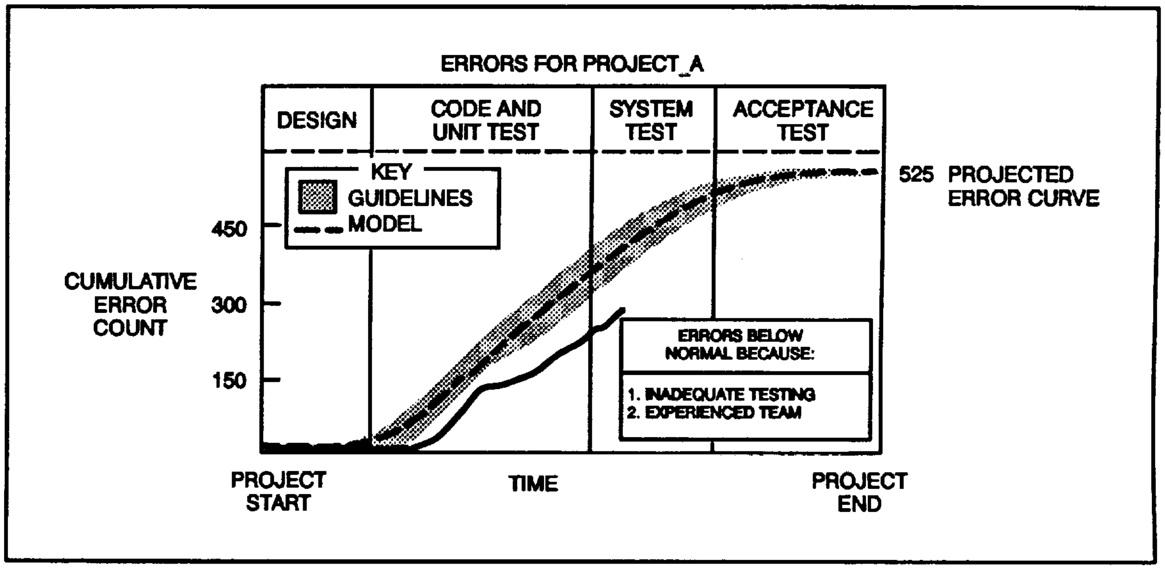


**Figure 2-5. Analysis Display—Trends**

Profile analysis displays a detailed breakdown of the current value of a selected measure into discrete categories that constitute a profile. Since more than one profile may be associated with each measure, the manager may view the data in several different ways. Figure 2-6 shows a profile display of reported errors based on the effort required to isolate each error.

### 2.3.5 Assessment

The assessment function presents the results of an overall project assessment that evaluates high-level quality attributes such as maintainability, correctability, and reliability. The function looks at current project data to compute a relative value for each attribute based on algorithms that define the objective measures to be factored into the calculation. Figure 2-7 shows the results of an assessment of overall project quality attributes with respect to a typical project in the environment.

### 2.3.6 Planning

The planning function helps managers reevaluate and modify the plan for the project of interest. The function allows managers to create and update alternative schedules and completion estimates with values input via an editor or with values derived from models of typical projects. Use of these alternative plans lets the manager see "what if" some aspect of the plan is changed.

### 2.3.7 Control

The SME assists managers in controlling project activities by providing guidance on how to correct possible weaknesses in a project. The guidance function examines the problems detected through analysis and assessment, searches for common factors, and suggests solutions based on changing the factors. Figure 2-8 shows the display of suggested actions for correcting an anticipated problem with software reliability as indicated by a higher-than-expected number of reported errors.

## 2.4 CONCEPT SUMMARY

Two views of the SME concept summarize the information presented thus far: (1) how information is used in the SME and (2) how the manager's activities are mapped into SME functions. These two views are presented in Figures 2-9 and 2-10, respectively.

Figure 2-9 summarizes the information flow in the SME and forms the basis for the data architecture presented in Section 3.1. The key importance of SEL data, research results, and management experience is clear.

Figure 2-10 summarizes the manager's activities and indicates how they can be grouped into four major SME functions: monitoring, assessment, planning, and guidance. This grouping serves as foundation for the functional architecture discussed in Section 3.2.

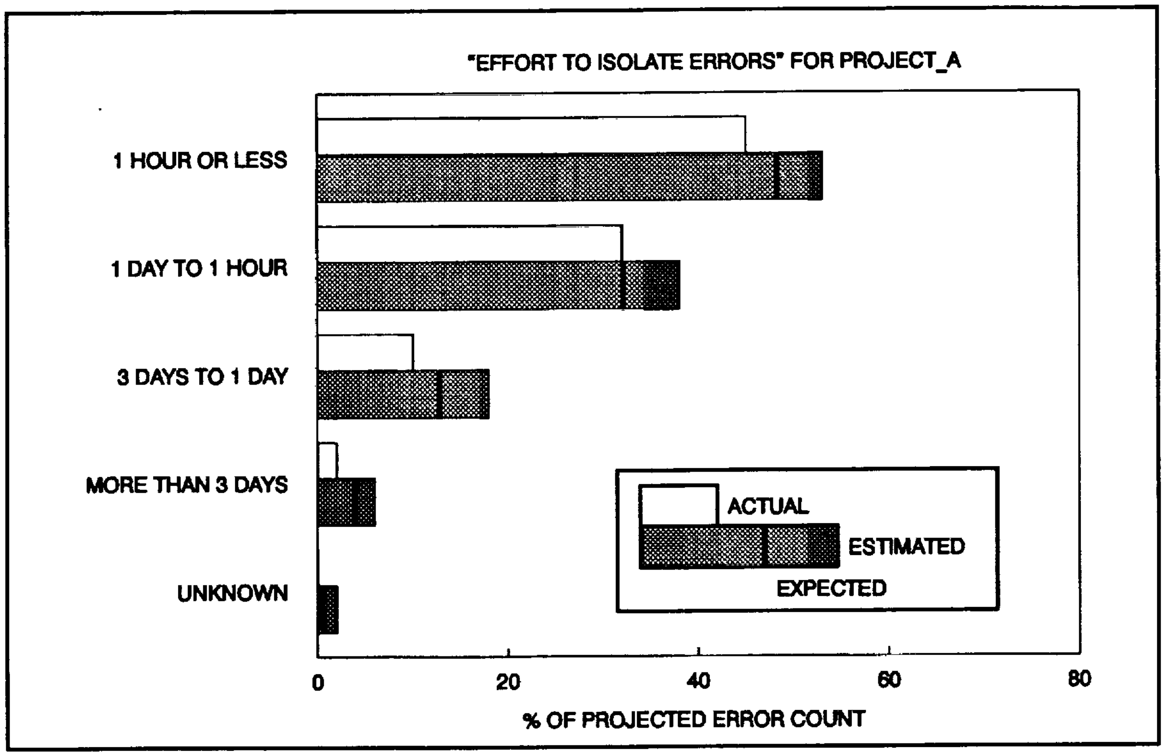


**Figure 2-6. Analysis Display—Profiles**

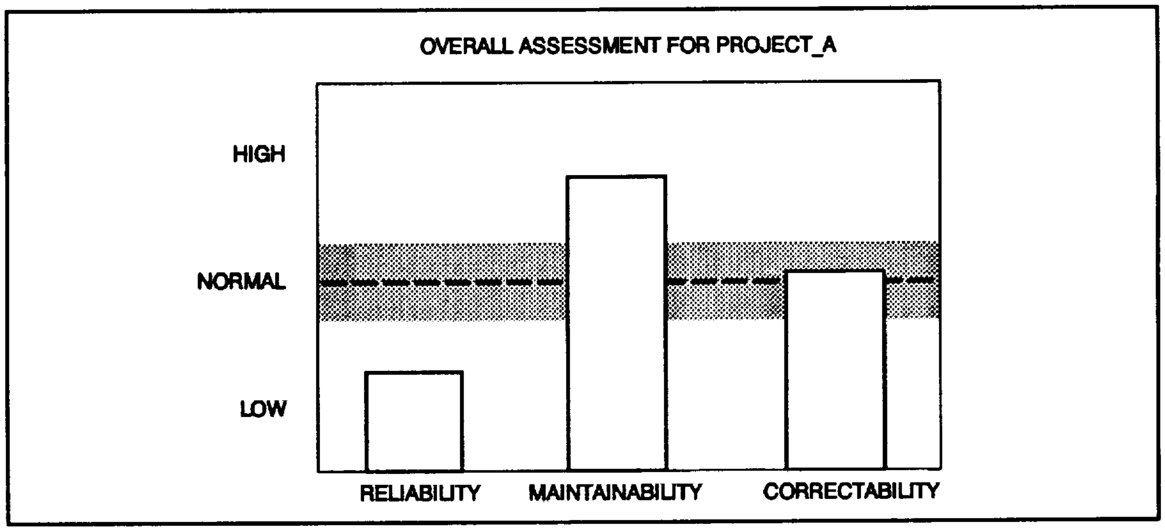


**Figure 2-7. Assessment Display—Overall**

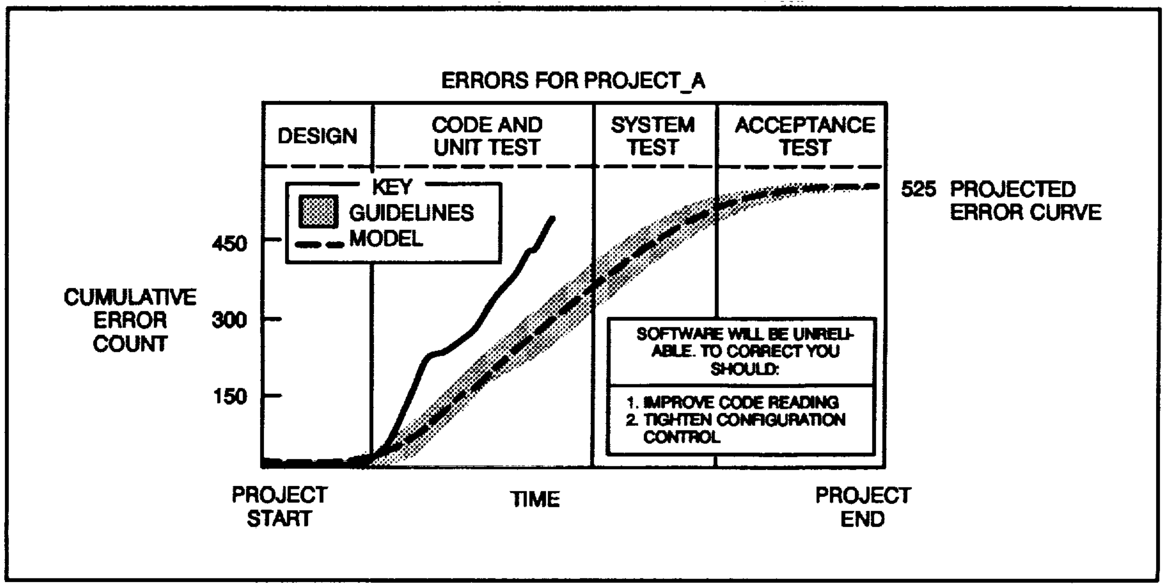


**Figure 2-8. Guidance Display**

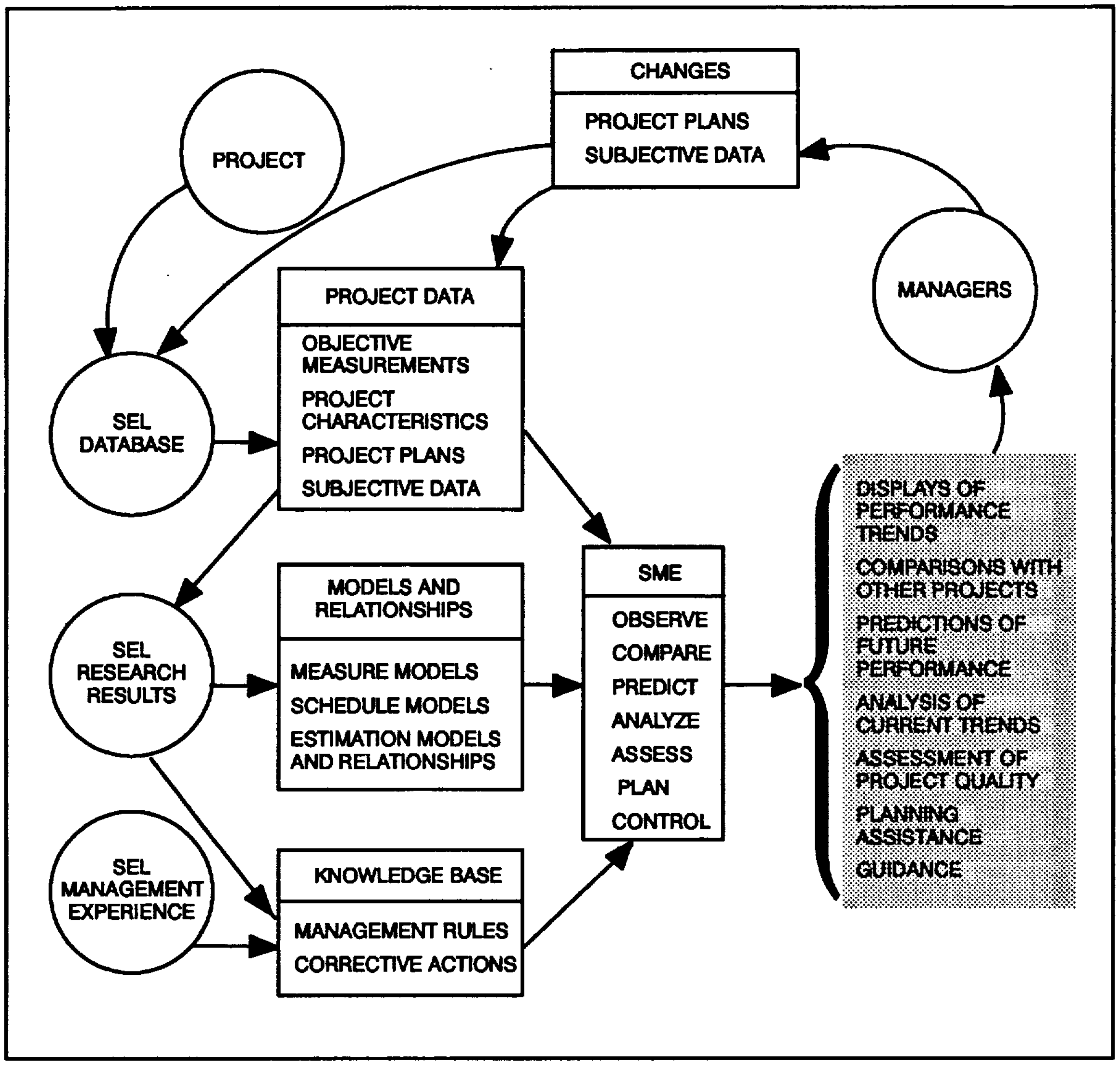


**Figure 2-9. SME Information Flow**

| MANAGEMENT ACTIVITY | SUBACTIVITY | SME FUNCTIONAL GROUPS | | | |
|---|---|---|---|---|---|
| | | MONITOR-ING | OVERALL ASSESS-MENT | PLANNING | GUIDANCE |
| OBSERVATION | OBSERVATION | X | | | |
| COMPARISON | COMPARISON TO OLD PROJECT | X | | | |
| | COMPARISON TO MODEL | X | | | |
| PREDICTION | PREDICTION | X | | | |
| ANALYSIS | ANALYSIS OF TREND OF MEASURE | X | | | |
| | ANALYSIS OF PROFILE OF MEASURE | X | | | |
| ASSESSMENT | ASSESSMENT OF OVERALL PROJECT | | X | | |
| PLANNING | ESTIMATING | | | X | |
| | SCHEDULING | | | X | |
| CONTROL | GUIDANCE | | | | X |

**Figure 2-10. Management Activities and SME Functional Groups**

# SECTION 3—ARCHITECTURE

This section presents a description of the high-level structure of the SME. The SME structure is influenced by three factors: the types and sources of data used by the system, the organization and partitioning of the functions performed by the SME, and the distribution of the data and functionality among the hardware components available to the SME. Each of these factors will be discussed below as the data architecture, functional architecture, and hardware architecture, respectively.

## 3.1 DATA ARCHITECTURE

The data architecture is a description of the source, content, and usage of the data elements required by the SME. The majority of the data elements are drawn from the SEL environment, as described in Section 2.2. Figure 3-1 shows the data elements used by the SME, grouped by source.

| SOURCE | DATA ELEMENT |
|---|---|
| SEL DATABASE | PROJECT LIST |
| | MEASURE LIST |
| | PROFILE LIST |
| | PROJECT/MEASURE AVAILABILITY LIST |
| | PROJECT/PROFILE AVAILABILITY LIST |
| | MEASURE DATA |
| | PROFILE DATA |
| | CURRENT SCHEDULE |
| | CURRENT ESTIMATES |
| | PROJECT CHARACTERISTICS |
| SEL RESEARCH | TABULAR MODELS |
| | ANALYTICAL MODELS |
| | ATTRIBUTE DEFINITIONS |
| SEL EXPERIENCE | KNOWLEDGE BASE |
| | RULE BASE |
| MANAGERS USING SME | ALTERNATIVE SCHEDULES |
| | ALTERNATIVE ESTIMATES |
| | PHASE ESTIMATES |
| | SUBJECTIVE DATA |

**Figure 3-1. Data Elements**

The individual data elements are discussed in detail below. They are presented in four main groups corresponding to the sources shown in Figure 3-1. The discussions of the data elements in each group are preceded by an introduction to the characteristics of the source in the SEL environment from which they are drawn.

The discussion of each element includes a description of which SME functions use the data, the source of the data, the data structure, and the number of occurrences of the data element. For simplicity, the structure of the data is described in terms of tables with rows and columns; the tables may be implemented as disk files, database tables, or internal memory structures. Many of the tables are interrelated and contain implied pointers to other tables.

### 3.1.1 Data From the SEL Database

One source for the SME is the data collected as part of the SEL database. In order to analyze ongoing software development efforts, the SME requires up-to-date information on the key dynamic parameters that characterize the software development process. The SEL database contains measures of a wide spectrum of process characteristics such as software changes and errors, effort expenditure, computer usage, and software product growth. Furthermore, the SEL database is the empirical basis for the other data used by the SME. Without the database, the models and relationships (Section 3.1.2) and the software management rules (Section 3.1.3) could not be derived or validated. Thus, a repository of collected data like the SEL database is critical to the development and structure of the SME.

Over the years, the SEL has collected data on nearly 100 software development projects in this environment. The types of data collected range from a weekly accounting of effort and computer utilization to statistics characterizing a completed project. These data are collected by means of forms, interviews, and automatic collection tools. Reference 5 contains a complete description of the data collection procedures. After the data are collected they are quality assured and entered into the SEL database, which resides under a database management system on a DEC VAX computer. By keeping archived historical information on completed projects as well as project information on ongoing development efforts, the database provides a rich source of software development data for this environment. References 1, 4, 5, and 11 contain more information on the SEL database.

The following subsections describe 10 types of data that the SME obtains from the SEL database. The first five types discussed (project list, measure list, profile list, project/measure availability list, and project/profile availability list) are used by the SME to identify and locate the project data available to it; these five types are often referred to in the discussions of the other types. The remaining five data types obtained from the SEL database (measure and profile data, current schedules and estimates, and project characteristics) contain project-specific data for each project.

### 3.1.1.1 PROJECT LIST

The project list identifies the names of all projects available for access through the SME.

In the SEL environment, the development effort for a single software product is called a project. The project serves as the basic entity about which the SEL collects data. All information stored in the SEL database is associated with a project.

The project list contains the names of ongoing and completed projects for which data are available. The SME presents this list to the manager for selection of a project to investigate. The SME uses the selected project name to point to the data that describe the development effort. The selected project is considered the project of interest.

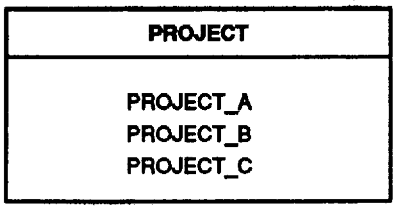
*Used by:* SME executive

*Source:* SEL database

*Structure:* Table with one column (project name)

*Instances:* There is one project list table.

*Example:* Figure 3-2 shows a sample list of project names.



| PROJECT |
|---|
| PROJECT_A |
| PROJECT_B |
| PROJECT_C |

**Figure 3-2. Project List**

### 3.1.1.2 MEASURE LIST

The measure list identifies the types of measure data defined for the SME.

The measure list is a key piece of data for the SME. The list contains the subset of SEL measurement data types that the SME is prepared to use. The presence of a measure in this list has implications beyond the fact that the data are collected in the SEL. The presence of a measure in the measure list also means that (1) the SME has some way of helping the manager estimate the value of the measure at the end of the project, (2) there are research results that describe how the measure is expected to grow during the project, and (3) there are software development management rules that help explain the causes of deviations in a measure from expected values.

The measure list also indicates that the SME has precisely defined how to extract the data from the SEL database. For some measure types the SME must select from among several similar types of data in the SEL database (for example, there are at least two ways to extract the count of software modules from the database). The definition used by the SME is contained in the software that interfaces the SME and the SEL database. Each of the three types of research results mentioned above (models, relationships, and management rules) is based on studies that used data extracted according to these definitions.

*Used by:* All SME functions

*Source:* Part of the SME implementation

*Structure:* Table with two columns (measure code, measure name)

*Instances:* There is one measure list.

*Example:* Figure 3-3 shows the current SME measure list.

| MEASURE CODE | MEASURE NAME |
|---|---|
| CPU | CPU HOURS |
| EFF | TOTAL STAFF HOURS |
| LOC | LINES OF CODE |
| MCH | MODULES CHANGED |
| MOD | MODULE COUNT |
| RCH | REPORTED CHANGES |
| RER | REPORTED ERRORS |
| RUN | COMPUTER JOBS |

**Figure 3-3. Measure List**

### 3.1.1.3 PROFILE LIST

The profile list identifies the types of profile data defined for the SME.

Each measure defined in the measure list (Section 3.1.1.2) may optionally have one or more associated profiles. Each profile provides additional detailed information on its related measure by capturing the breakdown of project data for the measure into discrete categories.

As with measures, the presence of a profile in the list implies that (1) the SEL collects the needed data, (2) the SME can estimate the values for the profile at the end of the project, and (3) a model exists that describes how the profile is expected to change during the project. The definition used by the SME in extracting the data is contained in the software that interfaces the SME and the SEL database.

*Used by*: Profile analysis, overall assessment

*Source*: Part of the SME implementation

*Structure*: Table with three columns (measure code, profile code, profile name)

*Instances*: There is one profile list.

*Example*: Figure 3-4 shows a sample SME profile list.

| MEASURE CODE | PROFILE CODE | PROFILE NAME |
|---|---|---|
| EFF | EFF1 | EFFORT BY ACTIVITY |
| MOD | MOD1 | MODULES BY TYPE |
| MOD | MOD2 | MODULES BY PURPOSE |
| RCH | RCH1 | EFFORT TO ISOLATE CHANGE |
| RCH | RCH2 | EFFORT TO IMPLEMENT CHANGE |
| RER | RER1 | EFFORT TO ISOLATE ERROR |
| RER | RER2 | EFFORT TO CORRECT ERROR |

**Figure 3-4. Profile List**

## 3.1.1.4 PROJECT/MEASURE AVAILABILITY LIST

The project/measure availability list associates each project with the types of measure data available for it.

The data collection process may not be the same for all projects. A particular type of measure may not be collected for a specific project. The knowledge that data are not collected is important to the SME because some SME functions might interpret a missing data value (such as no reported software changes halfway through system testing) as an indication that a problem exists with the development process.

*Used by:* All SME functions

*Source:* SEL database

*Structure:* Table with two columns (project name, measure name)

*Instances:* There is one project/measure availability list.

*Example:* Figure 3-5 shows a sample project/measure availability list. Note that in the sample, Project_A has no effort data available.

| PROJECT | MEASURE |
|---|---|
| PROJECT _A | CPU |
| PROJECT _A | RUN |
| PROJECT _A | LOC |
| PROJECT _A | MOD |
| PROJECT _A | RCH |
| PROJECT _A | MCH |
| PROJECT _A | RER |
| PROJECT _B | EFF |
| PROJECT _B | LOC |
| PROJECT _B | MOD |
| PROJECT _B | RCH |
| PROJECT _B | MCH |
| PROJECT _B | RER |
| PROJECT _C | EFF |
| PROJECT _C | CPU |
| PROJECT _C | RUN |
| PROJECT _C | LOC |
| PROJECT _C | MOD |
| PROJECT _C | RCH |
| PROJECT _C | MCH |
| PROJECT _C | RER |

**Figure 3-5. Project/Measure Availability List**

### 3.1.1.5 PROJECT/PROFILE AVAILABILITY LIST

The project/profile availability list associates each project with the types of profile data available for it.

As with measures, the data collection process may not be the same for all projects. Specific projects may not collect profile data for a given measure; however, profile data inherently cannot be collected without also collecting data for their related measure. As a result, whenever an entry exists in the project/profile availability list for a given profile, a corresponding entry must exist for its related measure in the project/measure availability list (Section 3.1.1.4).

*Used by*: Profile analysis, overall assessment

*Source*: SEL database

*Structure*: Table with two columns (project name, profile code)

*Instances*: There is one project/profile availability list.

*Example*: Figure 3-6 shows a sample project/profile availability list. Note that in the sample, Project_A has no profile data related to logical changes (RCH) and Project_C has no profile data related to reported errors (RER).

| PROJECT | PROFILE |
|---|---|
| PROJECT_A | RER1 |
| PROJECT_A | RER2 |
| PROJECT_B | EFF1 |
| PROJECT_B | RCH1 |
| PROJECT_B | RCH2 |
| PROJECT_B | RER1 |
| PROJECT_B | RER2 |
| PROJECT_C | RCH1 |
| PROJECT_C | RCH2 |

**Figure 3-6. Project/Profile Availability List**

### 3.1.1.6 MEASURE DATA

Measure data represent an historical record of measurements of a dynamic parameter over the project life cycle.

Measure data capture the growth of a measure as a function of time (the measures are discussed in Section 3.1.1.2). The data begin at zero at the start of a project and the cumulative value to date is recorded at the sampling frequency. The data stop at the end of a project for completed projects and at the most recent sampling date for ongoing projects. The measure data for ongoing and completed projects are stored in the same way.

The data are referred to as "weekly data" in the SEL because this is the sampling frequency used in SEL data collection activities; other frequencies might be used in another environment.

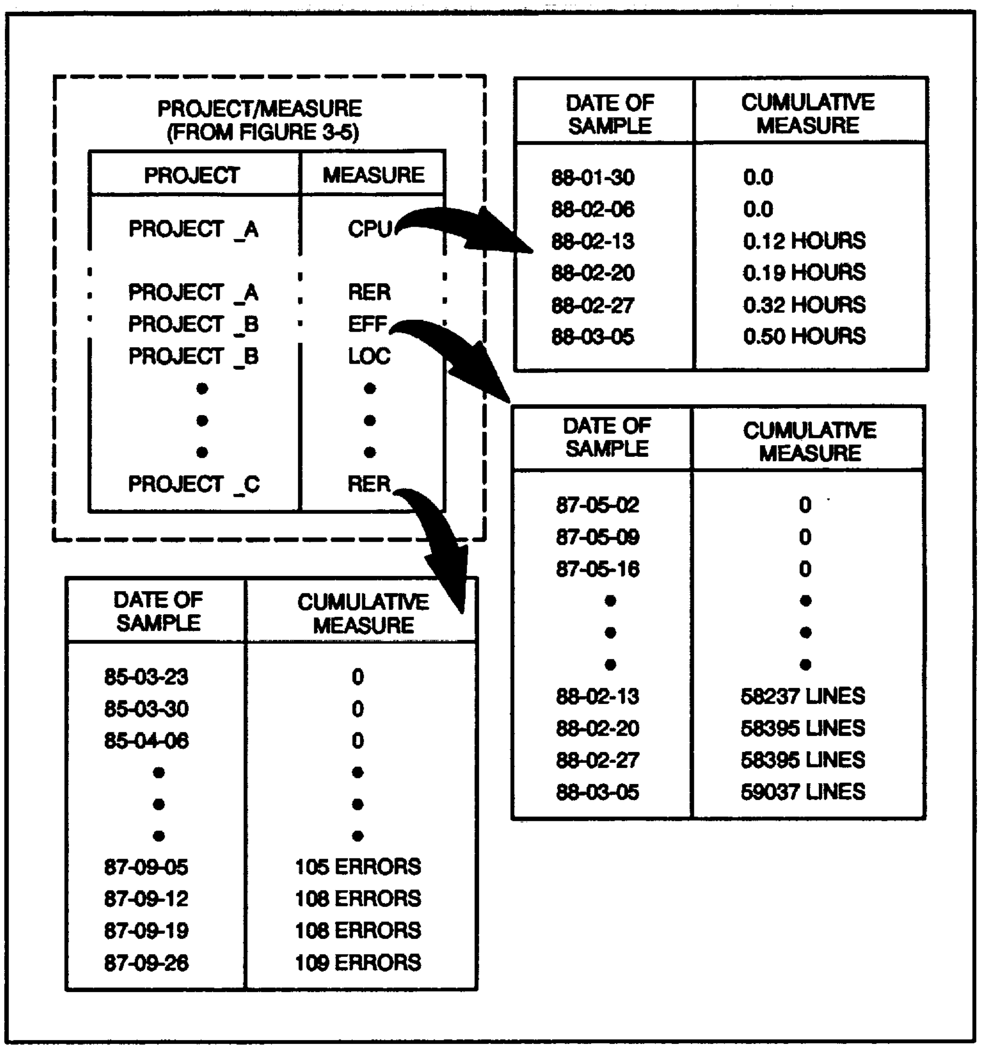
*Used by:* All SME functions

*Source:* SEL database

*Structure:* Table with two columns (date of sample, cumulative measure)

*Instances:* There is one table with measure data for each project/measure pair in the project/measure availability list.

*Example:* Figure 3-7 shows how the project/measure availability list points to each of the measure data tables. Each data table contains the series of cumulative measure values for one measure type for one project.


PROJECT/MEASURE (FROM FIGURE 3-5)

| PROJECT | MEASURE |
|---|---|
| PROJECT _A | CPU |
| PROJECT _A | RER |
| PROJECT _B | EFF |
| PROJECT _B | LOC |
| • | • |
| • | • |
| • | • |
| PROJECT _C | RER |

| DATE OF SAMPLE | CUMULATIVE MEASURE |
|---|---|
| 88-01-30 | 0.0 |
| 88-02-06 | 0.0 |
| 88-02-13 | 0.12 HOURS |
| 88-02-20 | 0.19 HOURS |
| 88-02-27 | 0.32 HOURS |
| 88-03-05 | 0.50 HOURS |

| DATE OF SAMPLE | CUMULATIVE MEASURE |
|---|---|
| 87-05-02 | 0 |
| 87-05-09 | 0 |
| 87-05-16 | 0 |
| • | • |
| • | • |
| • | • |
| 88-02-13 | 58237 LINES |
| 88-02-20 | 58395 LINES |
| 88-02-27 | 58395 LINES |
| 88-03-05 | 59037 LINES |

| DATE OF SAMPLE | CUMULATIVE MEASURE |
|---|---|
| 85-03-23 | 0 |
| 85-03-30 | 0 |
| 85-04-06 | 0 |
| • | • |
| • | • |
| • | • |
| 87-09-05 | 105 ERRORS |
| 87-09-12 | 108 ERRORS |
| 87-09-19 | 108 ERRORS |
| 87-09-26 | 109 ERRORS |

**Figure 3-7. Measure Data**

### 3.1.1.7 PROFILE DATA

Profile data represent an historical record of measurements over the project life cycle that detail the breakdown of a measure into discrete categories.

Profile data capture the growth of a measure over time as viewed in terms of the specific components established for that profile (the profiles are discussed in Section 3.1.1.3). For example, a profile of effort to isolate change (RCH1) categorizes the measure data for reported changes (RCH) into five components based on the actual time expended in isolating each change (1 hour or less, 1 hour to 1 day, 1 day to 3 days, more than 3 days, or unknown).

As with measure data, the data recorded for a profile begin at zero at the start of a project and the cumulative values to date are recorded at the sampling frequency. Instead of maintaining a single cumulative value for each sampling date, however, several running totals are kept with one value for each component in the profile. At every sample date, the sum of the values of the profile components should equal the corresponding cumulative value in its related measure data. The data stop at the end of a project for completed projects and at the most recent sampling date for ongoing projects. The profile data for ongoing and completed projects are stored in the same way.

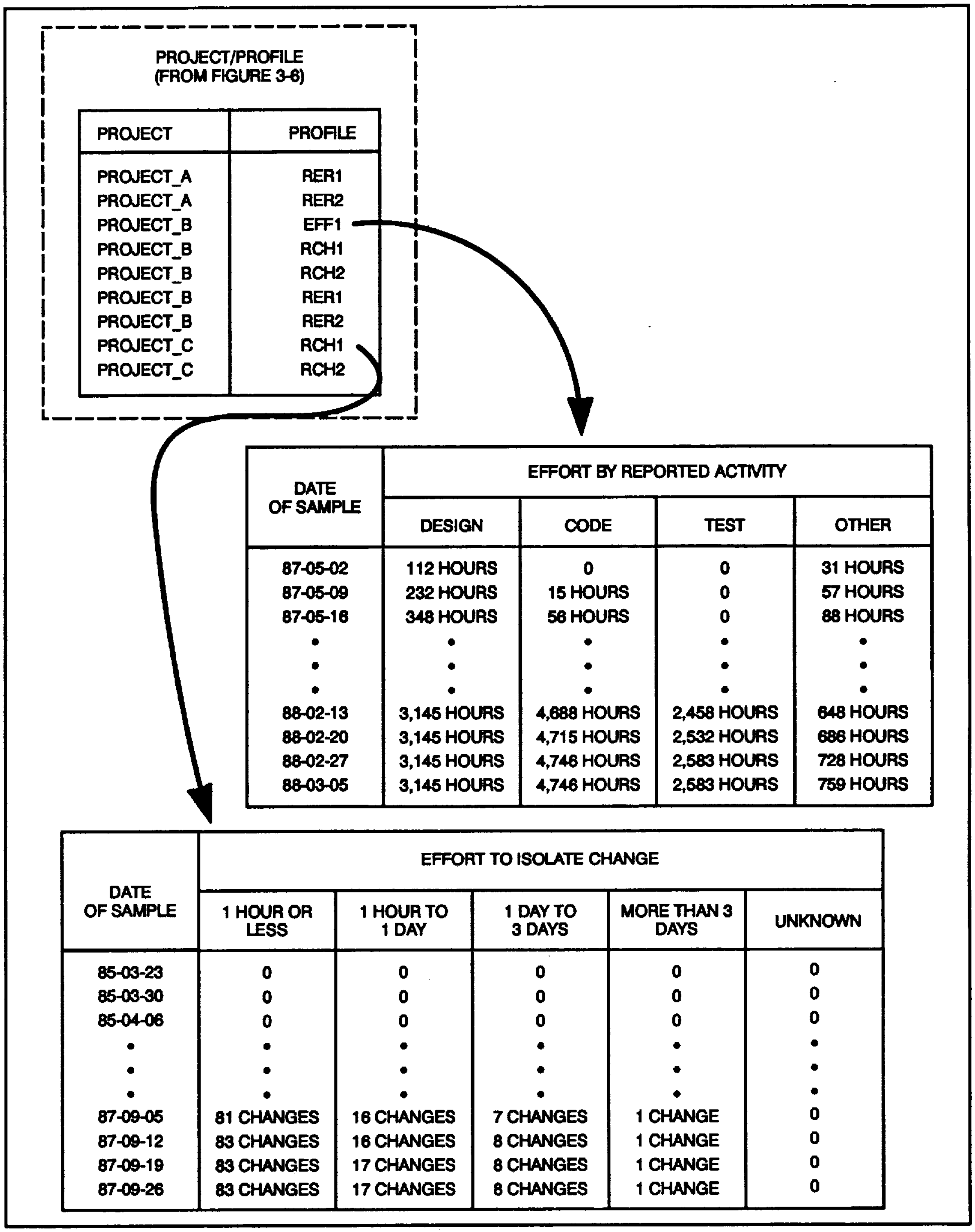
*Used by:* Profile analysis, overall assessment

*Source*: SEL database

*Structure*: Table with N+1 columns (date of sample, cumulative value for each component), where N is the number of profile components

*Instances:* There is one table with profile data for each project/profile pair in the project/profile availability list.

*Example*: Figure 3-8 shows how the project/profile availability list points to each of the profile data tables. Each data table contains a series of cumulative values for each profile component for one project.



PROJECT/PROFILE
(FROM FIGURE 3-6)

| PROJECT | PROFILE |
|---|---|
| PROJECT_A | RER1 |
| PROJECT_A | RER2 |
| PROJECT_B | EFF1 |
| PROJECT_B | RCH1 |
| PROJECT_B | RCH2 |
| PROJECT_B | RER1 |
| PROJECT_B | RER2 |
| PROJECT_C | RCH1 |
| PROJECT_C | RCH2 |

| DATE OF SAMPLE | EFFORT BY REPORTED ACTIVITY | | | |
|---|---|---|---|---|
| | DESIGN | CODE | TEST | OTHER |
| 87-05-02 | 112 HOURS | 0 | 0 | 31 HOURS |
| 87-05-09 | 232 HOURS | 15 HOURS | 0 | 57 HOURS |
| 87-05-16 | 348 HOURS | 56 HOURS | 0 | 88 HOURS |
| • | • | • | • | • |
| • | • | • | • | • |
| • | • | • | • | • |
| 88-02-13 | 3,145 HOURS | 4,688 HOURS | 2,458 HOURS | 648 HOURS |
| 88-02-20 | 3,145 HOURS | 4,715 HOURS | 2,532 HOURS | 686 HOURS |
| 88-02-27 | 3,145 HOURS | 4,746 HOURS | 2,583 HOURS | 728 HOURS |
| 88-03-05 | 3,145 HOURS | 4,746 HOURS | 2,583 HOURS | 759 HOURS |

| DATE OF SAMPLE | EFFORT TO ISOLATE CHANGE | | | | |
|---|---|---|---|---|---|
| | 1 HOUR OR LESS | 1 HOUR TO 1 DAY | 1 DAY TO 3 DAYS | MORE THAN 3 DAYS | UNKNOWN |
| 85-03-23 | 0 | 0 | 0 | 0 | 0 |
| 85-03-30 | 0 | 0 | 0 | 0 | 0 |
| 85-04-06 | 0 | 0 | 0 | 0 | 0 |
| • | • | • | • | • | • |
| • | • | • | • | • | • |
| • | • | • | • | • | • |
| 87-09-05 | 81 CHANGES | 16 CHANGES | 7 CHANGES | 1 CHANGE | 0 |
| 87-09-12 | 83 CHANGES | 16 CHANGES | 8 CHANGES | 1 CHANGE | 0 |
| 87-09-19 | 83 CHANGES | 17 CHANGES | 8 CHANGES | 1 CHANGE | 0 |
| 87-09-26 | 83 CHANGES | 17 CHANGES | 8 CHANGES | 1 CHANGE | 0 |

**Figure 3-8. Profile Data**

10001966

### 3.1.1.8 CURRENT SCHEDULE

The current schedule refers to a project's schedule.

A schedule is a list of development phases and the start and end dates for each phase. The "current" schedule is obtained from the manager through the SEL database and is used by SME functions unless the manager selects an alternative schedule (Section 3.1.4.1).

The format of a schedule follows the SEL database definition of a schedule. The schedule assumes a waterfall life cycle where the development process is a sequential, noniterative process. There are other, more complex, schema for schedules (and they may be used by the SME in the future), but the SME measurement models are based specifically on contiguous, nonoverlapping phases. The SME currently works with a standard four-phase schedule: design, code and unit testing, system testing, and acceptance testing.

Schedule dates are used to normalize the horizontal (time) axis of displays of historical data presented to the manager by the SME.

*Used by:* All SME functions

*Source:* SEL database

*Structure:* Table with three columns (phase name, phase start date, phase end date)

*Instances:* There is one current schedule table for each project.

*Example:* Figure 3-9 shows the project list pointing to sample current schedule tables.

### 3.1.1.9 CURRENT ESTIMATE SET

The current estimate set is a set of estimated completion values for a project.

The estimates are a set of expected measure data values at project completion. The set contains one value for each type of measure data (Section 3.1.1.2). Project completion is defined as the end date of the last phase (acceptance testing) in the standard SME schedule. The "current" estimates are obtained from the manager through the SEL database and are used by SME functions until the manager selects an alternative estimate (Section 3.1.4.2).

The estimates are used to normalize the vertical (measurement) axis of displays of historical data presented to the manager by the SME.

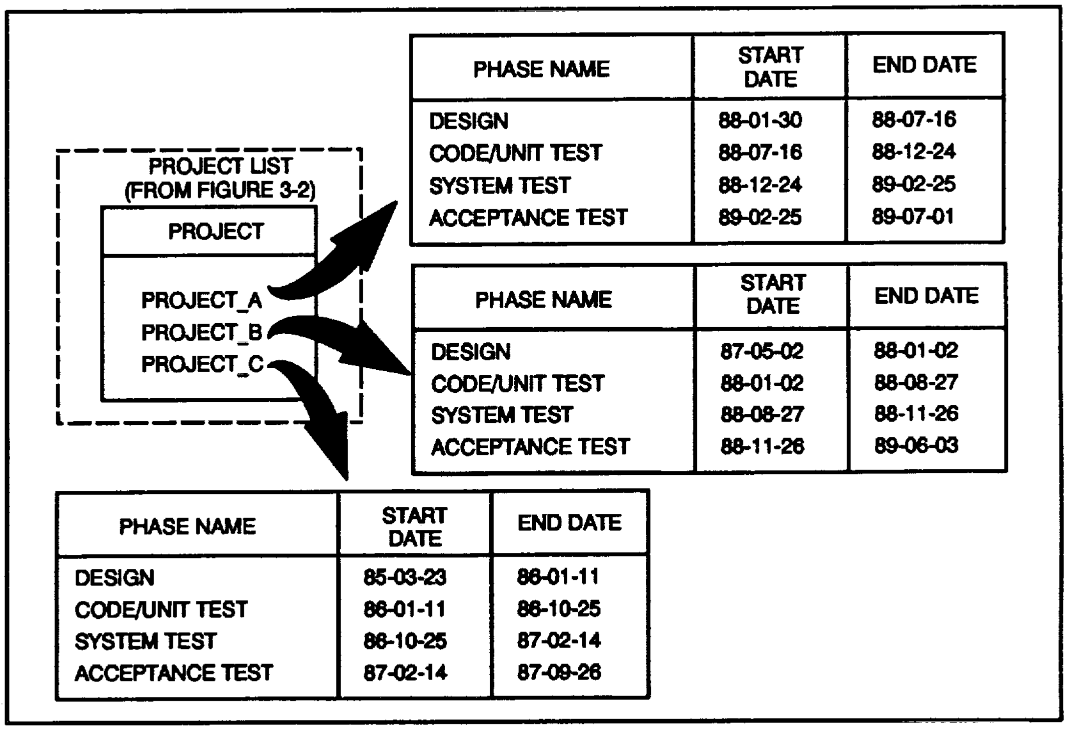
*Used by:* All SME functions

*Source:* SEL database

*Structure:* Table with two columns (measure name, estimated value at project completion)

*Instances:* There is one current estimate set for each project.

*Example:* Figure 3-10 shows samples of current estimate set tables.

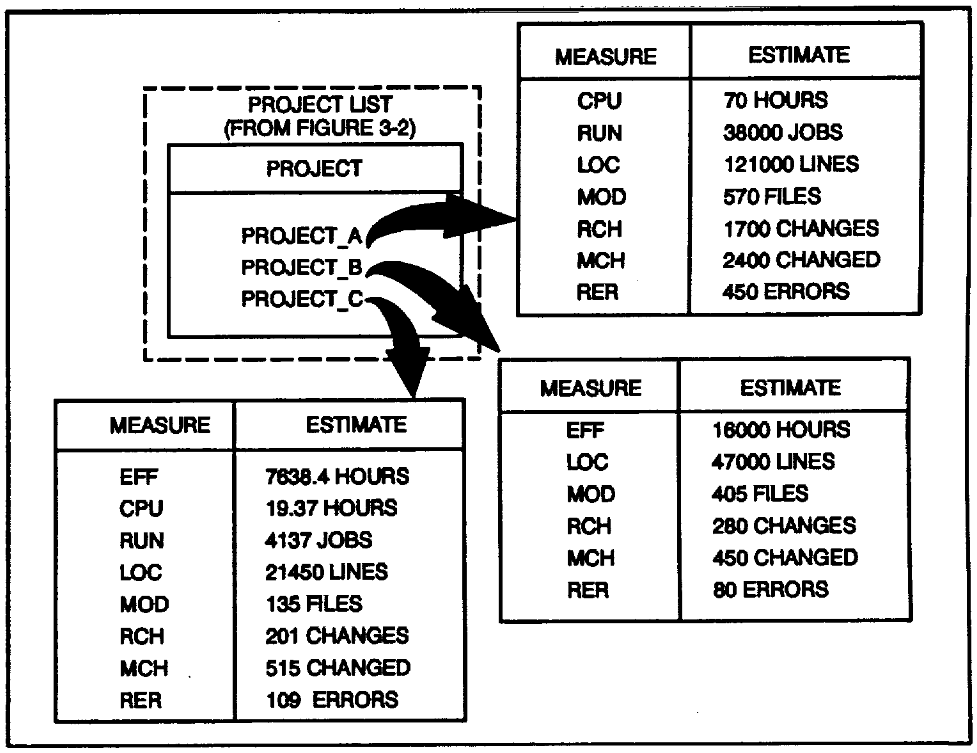


| PHASE NAME | START DATE | END DATE |
|---|---|---|
| DESIGN | 88-01-30 | 88-07-16 |
| CODE/UNIT TEST | 88-07-16 | 88-12-24 |
| SYSTEM TEST | 88-12-24 | 89-02-25 |
| ACCEPTANCE TEST | 89-02-25 | 89-07-01 |

| PHASE NAME | START DATE | END DATE |
|---|---|---|
| DESIGN | 87-05-02 | 88-01-02 |
| CODE/UNIT TEST | 88-01-02 | 88-08-27 |
| SYSTEM TEST | 88-08-27 | 88-11-26 |
| ACCEPTANCE TEST | 88-11-26 | 89-06-03 |

| PHASE NAME | START DATE | END DATE |
|---|---|---|
| DESIGN | 85-03-23 | 86-01-11 |
| CODE/UNIT TEST | 86-01-11 | 86-10-25 |
| SYSTEM TEST | 86-10-25 | 87-02-14 |
| ACCEPTANCE TEST | 87-02-14 | 87-09-26 |

**Figure 3-9. Current Schedule**



| MEASURE | ESTIMATE |
|---|---|
| CPU | 70 HOURS |
| RUN | 38000 JOBS |
| LOC | 121000 LINES |
| MOD | 570 FILES |
| RCH | 1700 CHANGES |
| MCH | 2400 CHANGED |
| RER | 450 ERRORS |

| MEASURE | ESTIMATE |
|---|---|
| EFF | 16000 HOURS |
| LOC | 47000 LINES |
| MOD | 405 FILES |
| RCH | 280 CHANGES |
| MCH | 450 CHANGED |
| RER | 80 ERRORS |

| MEASURE | ESTIMATE |
|---|---|
| EFF | 7638.4 HOURS |
| CPU | 19.37 HOURS |
| RUN | 4137 JOBS |
| LOC | 21450 LINES |
| MOD | 135 FILES |
| RCH | 201 CHANGES |
| MCH | 515 CHANGED |
| RER | 109 ERRORS |

**Figure 3-10. Current Estimates**

## 3.1.1.10 PROJECT CHARACTERISTICS

Project characteristics are a collection of objective facts that characterize a project.

The characteristics data are used by the SME to ensure that the correct set of research results (models or relationships) are used for analysis of the project. The characteristics are combined to produce a "project type," which is then used to select the appropriate set of research data.

Currently the data describe the following project characteristics: development language, computer environment, and application area.

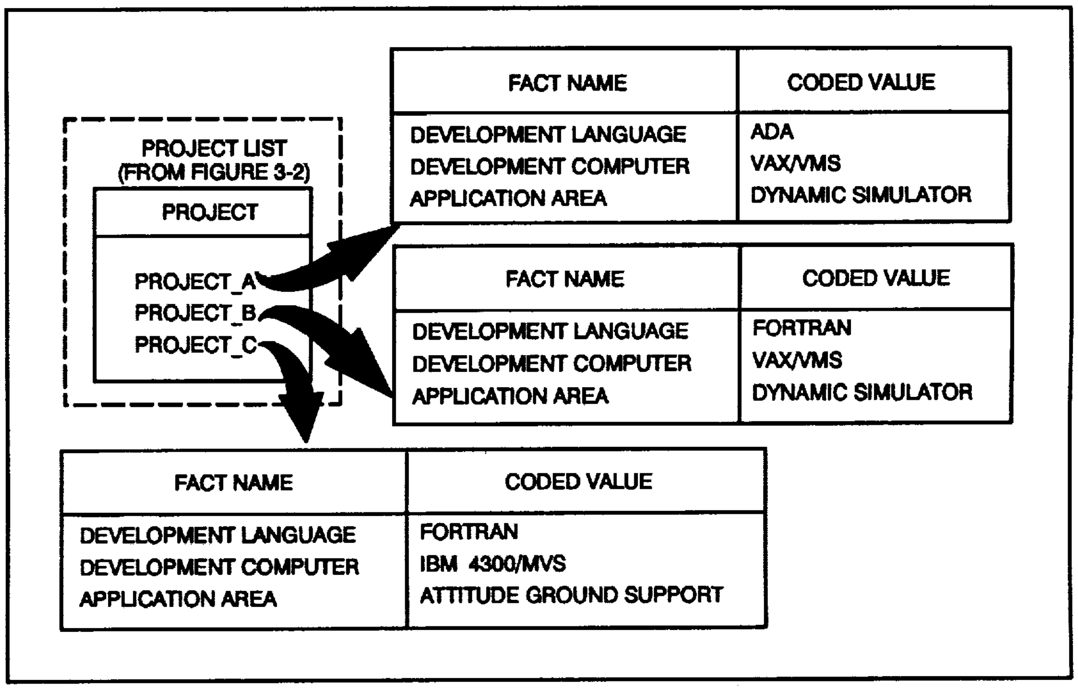
*Used by:* All SME functions

*Source:* SEL database

*Structure:* Table with two columns (fact name, coded value)

*Instances:* There is one project characteristics table for each project.

*Example:* Figure 3-11 shows sample project characteristics data.



| FACT NAME | CODED VALUE |
|---|---|
| DEVELOPMENT LANGUAGE | ADA |
| DEVELOPMENT COMPUTER | VAX/VMS |
| APPLICATION AREA | DYNAMIC SIMULATOR |

| FACT NAME | CODED VALUE |
|---|---|
| DEVELOPMENT LANGUAGE | FORTRAN |
| DEVELOPMENT COMPUTER | VAX/VMS |
| APPLICATION AREA | DYNAMIC SIMULATOR |

| FACT NAME | CODED VALUE |
|---|---|
| DEVELOPMENT LANGUAGE | FORTRAN |
| DEVELOPMENT COMPUTER | IBM 4300/MVS |
| APPLICATION AREA | ATTITUDE GROUND SUPPORT |

**Figure 3-11. Project Characteristics**

### 3.1.2 Data From Research

The data used by the SME include information obtained from numerous studies and experiments performed within the SEL. The utilization of these research results is an essential component of the SME.

The SEL has a well-established method for studying and experimenting on the software development process and product (Reference 12). Over the years, the SEL has evaluated numerous software development methodologies, characterized the software development process, and developed models of the software development environment through an extensive measurement program. This measurement program utilizes the SEL database (Section 3.1.1) to determine various models and measures of the software process and product. The results of these studies and experiments have been fed back into the software development and management process within the environment. One goal of the SME is to automate the utilization of these results.

The results used by the SME include data on estimation, models, and scheduling. By using the SEL database to analyze the software development process within this environment, numerous estimating relationships have been developed. One example of such a relationship is in the area of cost estimation. The SEL uses data on previous projects to fit a model for estimating the cost of a software project in this environment:

$$E = 8.45 * f_1 * f_2 * \ldots f_k * (L^{1.05})$$

| | | | |
|---|---|---|---|
| where | $E$ | = | development effort in staff-months |
| | $f_k$ | = | project-specific adjustment factors |
| | $L$ | = | size of the project in thousands of lines of developed code |

The SME uses this relationship to allow managers to estimate the cost of projects. Reference 13 contains more information on SEL cost estimation methods.

Another example of an estimation relationship useful to the SME is the relationship between project duration and developed lines of source code:

$$D = 4.84 * (L^{0.257})$$

| | | | |
|---|---|---|---|
| where | $D$ | = | project duration in months |
| | $L$ | = | size of the project in thousands of lines of developed code |

This relationship is extremely useful in predicting the duration of a project. Other estimating relationships of this type have been developed by the SEL and are being integrated into the SME (Reference 14).

The second area of research results involves the use of models of software development measures. For example, using SEL data it is possible to develop a model of the typical effort expenditure over the life of a project. This model can be represented as a table of values:

| PHASE | CUMULATIVE FRACTION OF EFFORT AT END OF PHASE |
|---|---|
| DESIGN | 0.30 |
| CODE AND UNIT TEST | 0.70 |
| SYSTEM TEST | 0.90 |
| ACCEPTANCE TEST | 1.00 |

Similar models have been developed for many other software development measures. This type of model is quite useful in comparing an ongoing project to the typical project within the environment.

Another example of a model of software development measures is a model of reliability over the life of a project. The following table shows a model of the number of errors uncovered per thousands of lines of code during a particular life-cycle phase within the SEL environment:

| PHASE | ERRORS REMOVED PER THOUSANDS OF LINES OF CODE |
|---|---|
| CODE AND UNIT TEST | 8 |
| SYSTEM TEST | 4 |
| ACCEPTANCE TEST | 2 |

Such a model is extremely useful in assessing the reliability of a project or in predicting and comparing the number of errors that might be found in a system. Many other such models have been developed and are being integrated into the SME.

Another area of research involves utilizing templates of schedules. These schedule templates were again developed by using data for past projects to determine the percentage of project duration typically spent in each development phase. As an example, for FORTRAN projects within this environment, the following schedule template usually holds:

| PHASE | CUMULATIVE FRACTION OF DURATION AT END OF PHASE |
|---|---|
| DESIGN | 0.35 |
| CODE AND UNIT TEST | 0.65 |
| SYSTEM TEST | 0.85 |
| ACCEPTANCE TEST | 1.00 |

This type of schedule template is used by managers in planning and in comparing an ongoing project to the template.

The examples of research results presented in this section are a small sample of the types of results being used in the SME. By utilizing the SEL database and an experimentation process, numerous results and equations have been developed. One of the goals of the SME is to integrate these results into an overall management environment.

The following subsections describe the ways in which the results of SEL research are made available to the SME: tabular models, analytical models, and attribute definitions. Tabular models are used by the SME to describe the growth of measurement data and to define schedules. Analytical models are used to estimate the value of measures at project completion. Attribute definitions are used to assess overall project quality factors.

#### 3.1.2.1 TABULAR MODELS—MEASURES

Tabular models of measure data describe the expected growth of the cumulative value of those data.

Measure models are tabulations of a measure as a function of development phase. The measure is expressed as a fraction starting at 0 at the start of the first phase and reaching 1 at the end of the last phase. Values are tabulated at the end of each phase and at some intermediate fractions of a phase. Each model has an associated value that represents the normal allowable deviation of a measure from the tabulated values.

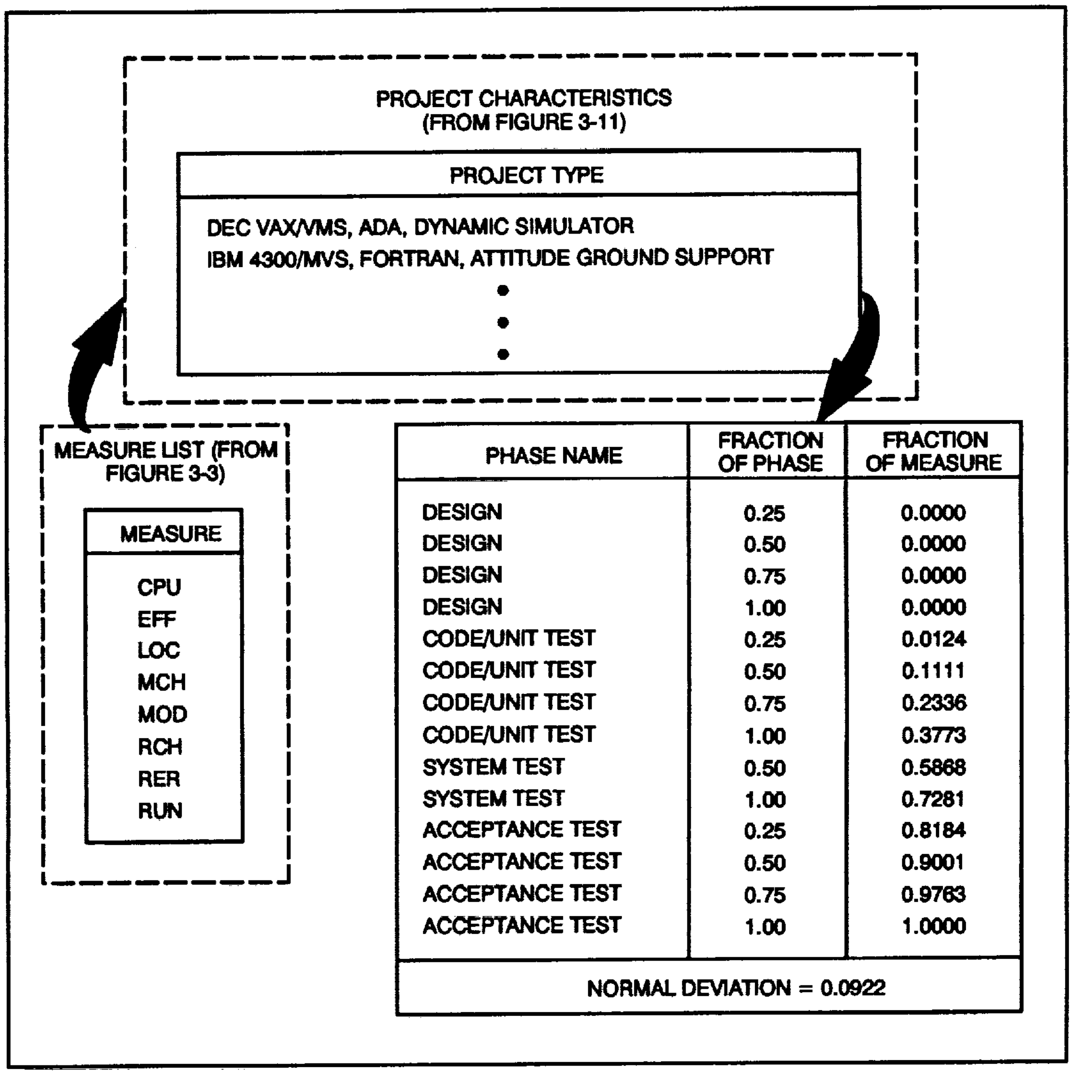
*Used by:* Comparison, prediction, trend analysis, overall assessment

*Source:* Research

*Structure:* Table with three columns (phase name, fraction of phase, fraction of measure), scalar value (magnitude of normal deviation)

*Instances:* There is one table for each measure type and project type.

*Example:* Figure 3-12 illustrates the dependence of measure models on the project type derived from the project characteristics data.


PROJECT CHARACTERISTICS (FROM FIGURE 3-11)

| PROJECT TYPE |
|---|
| DEC VAX/VMS, ADA, DYNAMIC SIMULATOR |
| IBM 4300/MVS, FORTRAN, ATTITUDE GROUND SUPPORT |
| • |
| • |
| • |

MEASURE LIST (FROM FIGURE 3-3)

| MEASURE |
|---|
| CPU |
| EFF |
| LOC |
| MCH |
| MOD |
| RCH |
| RER |
| RUN |

| PHASE NAME | FRACTION OF PHASE | FRACTION OF MEASURE |
|---|---|---|
| DESIGN | 0.25 | 0.0000 |
| DESIGN | 0.50 | 0.0000 |
| DESIGN | 0.75 | 0.0000 |
| DESIGN | 1.00 | 0.0000 |
| CODE/UNIT TEST | 0.25 | 0.0124 |
| CODE/UNIT TEST | 0.50 | 0.1111 |
| CODE/UNIT TEST | 0.75 | 0.2336 |
| CODE/UNIT TEST | 1.00 | 0.3773 |
| SYSTEM TEST | 0.50 | 0.5868 |
| SYSTEM TEST | 1.00 | 0.7281 |
| ACCEPTANCE TEST | 0.25 | 0.8184 |
| ACCEPTANCE TEST | 0.50 | 0.9001 |
| ACCEPTANCE TEST | 0.75 | 0.9763 |
| ACCEPTANCE TEST | 1.00 | 1.0000 |
| NORMAL DEVIATION = 0.0922 | | |

**Figure 3-12. Tabular Models—Measures**

### 3.1.2.2 TABULAR MODELS—PROFILES

Tabular models of profile data describe the expected growth of the cumulative values of each component specified for the profile.

Profile models are tabulations of the profile associated with a given measure as a function of development phase. As with measure models, profile values are expressed as a fraction starting at 0 at the start of the first phase. At the end of the last phase, the sum of the values of the profile components reaches 1. Values are tabulated for each component at the end of each phase and at some intermediate fractions of a phase.

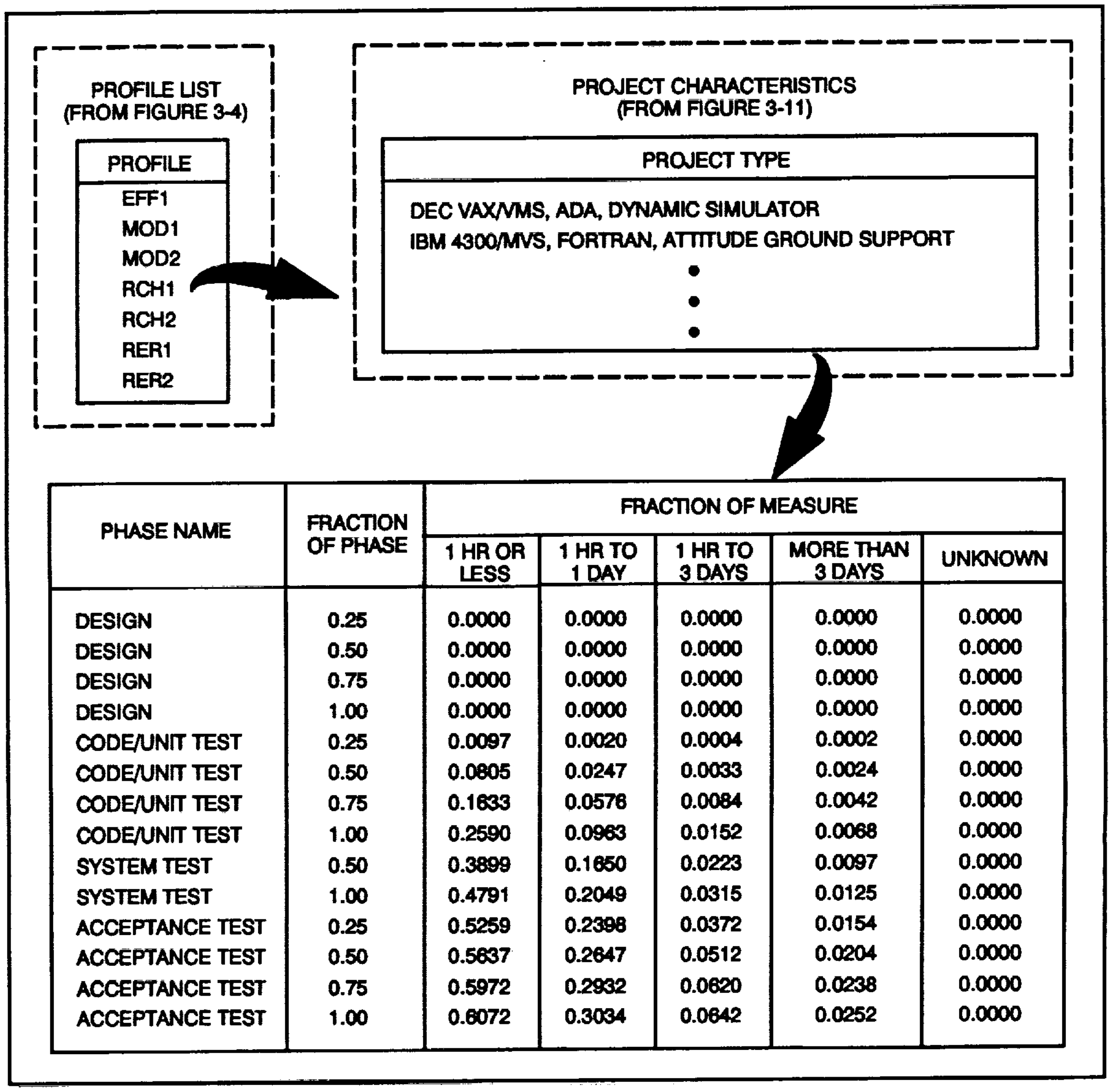
*Used by:* Profile analysis, overall assessment

*Source:* Research

*Structure:* Table with N+2 columns (phase name, fraction of phase, fraction of measure for each component), where N is the number of profile components

*Instances:* There is one table for each profile type and project type.

*Example:* Figure 3-13 illustrates the dependence of profile models on the project type derived from the project characteristics table.



PROFILE LIST
(FROM FIGURE 3-4)

| PROFILE |
|---|
| EFF1 |
| MOD1 |
| MOD2 |
| RCH1 |
| RCH2 |
| RER1 |
| RER2 |

PROJECT CHARACTERISTICS
(FROM FIGURE 3-11)

| PROJECT TYPE |
|---|
| DEC VAX/VMS, ADA, DYNAMIC SIMULATOR |
| IBM 4300/MVS, FORTRAN, ATTITUDE GROUND SUPPORT |
| • |
| • |
| • |

| PHASE NAME | FRACTION OF PHASE | FRACTION OF MEASURE | | | | |
|---|---|---|---|---|---|---|
| | | 1 HR OR LESS | 1 HR TO 1 DAY | 1 HR TO 3 DAYS | MORE THAN 3 DAYS | UNKNOWN |
| DESIGN | 0.25 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| DESIGN | 0.50 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| DESIGN | 0.75 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| DESIGN | 1.00 | 0.0000 | 0.0000 | 0.0000 | 0.0000 | 0.0000 |
| CODE/UNIT TEST | 0.25 | 0.0097 | 0.0020 | 0.0004 | 0.0002 | 0.0000 |
| CODE/UNIT TEST | 0.50 | 0.0805 | 0.0247 | 0.0033 | 0.0024 | 0.0000 |
| CODE/UNIT TEST | 0.75 | 0.1633 | 0.0576 | 0.0084 | 0.0042 | 0.0000 |
| CODE/UNIT TEST | 1.00 | 0.2590 | 0.0963 | 0.0152 | 0.0068 | 0.0000 |
| SYSTEM TEST | 0.50 | 0.3899 | 0.1650 | 0.0223 | 0.0097 | 0.0000 |
| SYSTEM TEST | 1.00 | 0.4791 | 0.2049 | 0.0315 | 0.0125 | 0.0000 |
| ACCEPTANCE TEST | 0.25 | 0.5259 | 0.2398 | 0.0372 | 0.0154 | 0.0000 |
| ACCEPTANCE TEST | 0.50 | 0.5637 | 0.2647 | 0.0512 | 0.0204 | 0.0000 |
| ACCEPTANCE TEST | 0.75 | 0.5972 | 0.2932 | 0.0620 | 0.0238 | 0.0000 |
| ACCEPTANCE TEST | 1.00 | 0.6072 | 0.3034 | 0.0642 | 0.0252 | 0.0000 |

**Figure 3-13. Tabular Models—Profiles**

### 3.1.2.3 TABULAR MODELS—SCHEDULES

Tabular models of schedules describe the amount of time that should be spent in each development phase.

The schedule template is a tabulation of the fraction of the project duration that should be allocated to each development phase when creating a project schedule. The schedule template is also used as a model of the schedule when performing prediction, guideline comparison, trend analysis, and overall assessment.

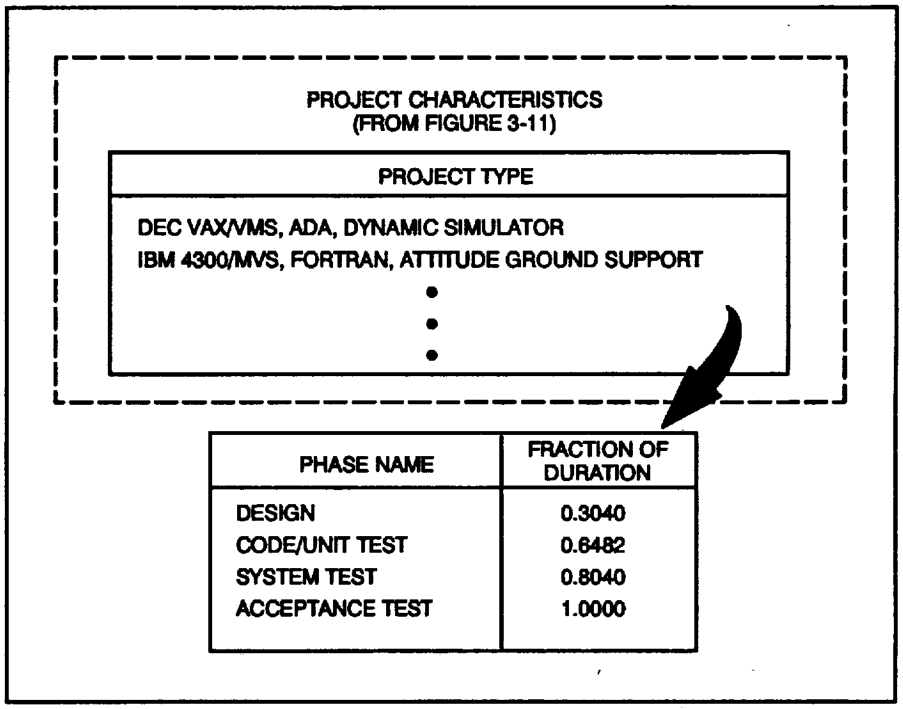
*Used by:* All SME functions

*Source:* Research

*Structure:* Table with two columns (phase name, fraction of duration)

*Instances:* There is one table for each project type.

*Example:* Figure 3-14 illustrates the dependence of the schedule model on the project type derived from the project characteristics data.



| PROJECT TYPE |
|---|
| DEC VAX/VMS, ADA, DYNAMIC SIMULATOR |
| IBM 4300/MVS, FORTRAN, ATTITUDE GROUND SUPPORT |
| • |
| • |
| • |

| PHASE NAME | FRACTION OF DURATION |
|---|---|
| DESIGN | 0.3040 |
| CODE/UNIT TEST | 0.6482 |
| SYSTEM TEST | 0.8040 |
| ACCEPTANCE TEST | 1.0000 |

**Figure 3-14. Tabular Models—Schedules**

## 3.1.2.4 ANALYTICAL MODELS—ESTIMATE SETS

Analytical models of estimate sets describe the relationships that exist among completion values of measures.

The estimate set models are a tabulation of normalized completion values for each measure defined in the measure list. Developed by analyzing data for past projects, the completion values in the set are normalized to 1000 lines of code.

These models implicitly capture the set of linear relationships that exist between each pair of measures. As a result, software routines that are functional in nature can be developed to access the models and return a wide range of useful information. A set of these routines is used by the SME to (1) produce a full set of completion estimates for all measures given the expected completion value for any one measure and (2) return the ratio of estimated completion values for any two specified measures.

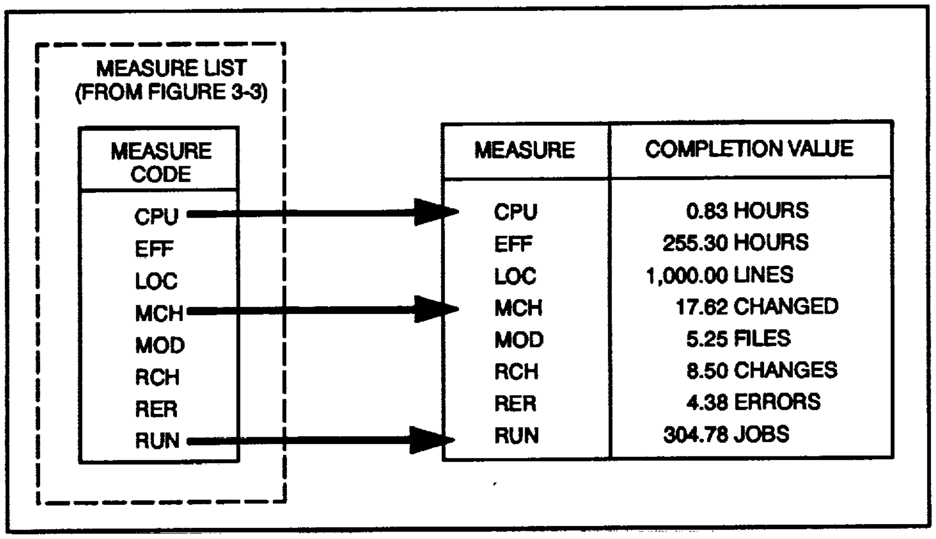
*Used by:* Planning

*Source:* Research

*Structure:* Table with two columns (measure, completion value)

*Instances*: There is one table for each project type.

*Example:* Figure 3-15 illustrates the concept of an estimate set model and its dependence on the measures defined in the measure list.



| MEASURE | COMPLETION VALUE |
|---|---|
| CPU | 0.83 HOURS |
| EFF | 255.30 HOURS |
| LOC | 1,000.00 LINES |
| MCH | 17.62 CHANGED |
| MOD | 5.25 FILES |
| RCH | 8.50 CHANGES |
| RER | 4.38 ERRORS |
| RUN | 304.78 JOBS |

**Figure 3-15. Analytical Models—Estimate Sets**

### 3.1.2.5 ANALYTICAL MODELS—ESTIMATING RELATIONSHIPS

Analytical models of estimating relationships consist of a software library containing the functional relationships among completion values of measures.

The analytical models are used by the SME to produce a full set of completion estimates for measures given a minimum of information about the size of the project. An analytical model for total project duration is also included in this group.

Although similar to analytical models of estimate sets (Section 3.1.2.4), these models use a software library that can support complex relationships that are not linear.

While these models are functional in nature, they are discussed here along with the data architecture because they are the results of research into a particular environment. Other relationships may apply to other environments or to other project types in the same environment. The relationships may also change with time as new methodologies and languages are introduced or as experience is gained with the application area. This suggests that the relationships may change often and their functional definitions should be kept separated from the SME functional architecture.

*Used by:* Planning

*Source:* Research

*Structure:* Software library

*Instances:* There is one software library of analytical models for each project type.

*Example:* Figure 3-16 illustrates the concept of a software library of analytical models.

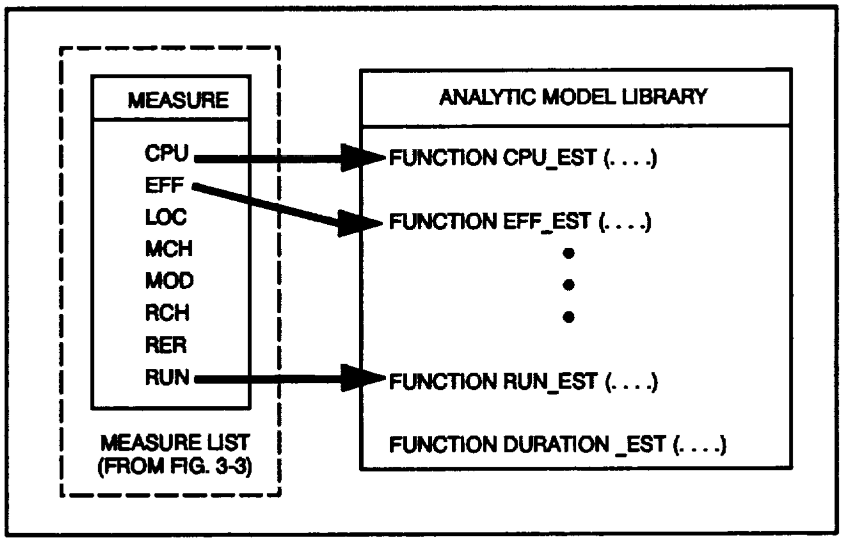

MEASURE
CPU
EFF
LOC
MCH
MOD
RCH
RER
RUN
MEASURE LIST (FROM FIG. 3-3)
ANALYTIC MODEL LIBRARY
FUNCTION CPU_EST (. . . .)
FUNCTION EFF_EST (. . . .)
FUNCTION RUN_EST (. . . .)
FUNCTION DURATION _EST (. . . .)

**Figure 3-16. Analytical Models—Estimating Relationships**

### 3.1.2.6 ATTRIBUTE DEFINITIONS

The list of attribute definitions describes how objective data are used to perform overall quality assessment for a project.

The SME defines overall quality in terms of product attributes such as correctability, maintainability, and reliability. Relative ratings can be calculated and assigned to these attributes by applying functions to various types of objective data. For example, a rating for the maintainability attribute might be derived by calculating the percentage of the total number of changes that were isolated and implemented in less than 1 day.

The attribute definitions list decomposes each attribute into one or more weighted factors and further defines each weighted factor as a function. These functions then evaluate a project's objective measurement data to produce a relative rating for each quality attribute.

*Used by:* Overall assessment

*Source:* Research

*Structure:* Table containing attribute names, factors associated with each attribute, factor weights, factor definition functions, and acceptable ranges

*Instances:* There is one list of attribute definitions.

*Example:* Figure 3-17 shows how the attribute definitions relate attributes to factors and in turn relate factors to the historical measure and profile data.

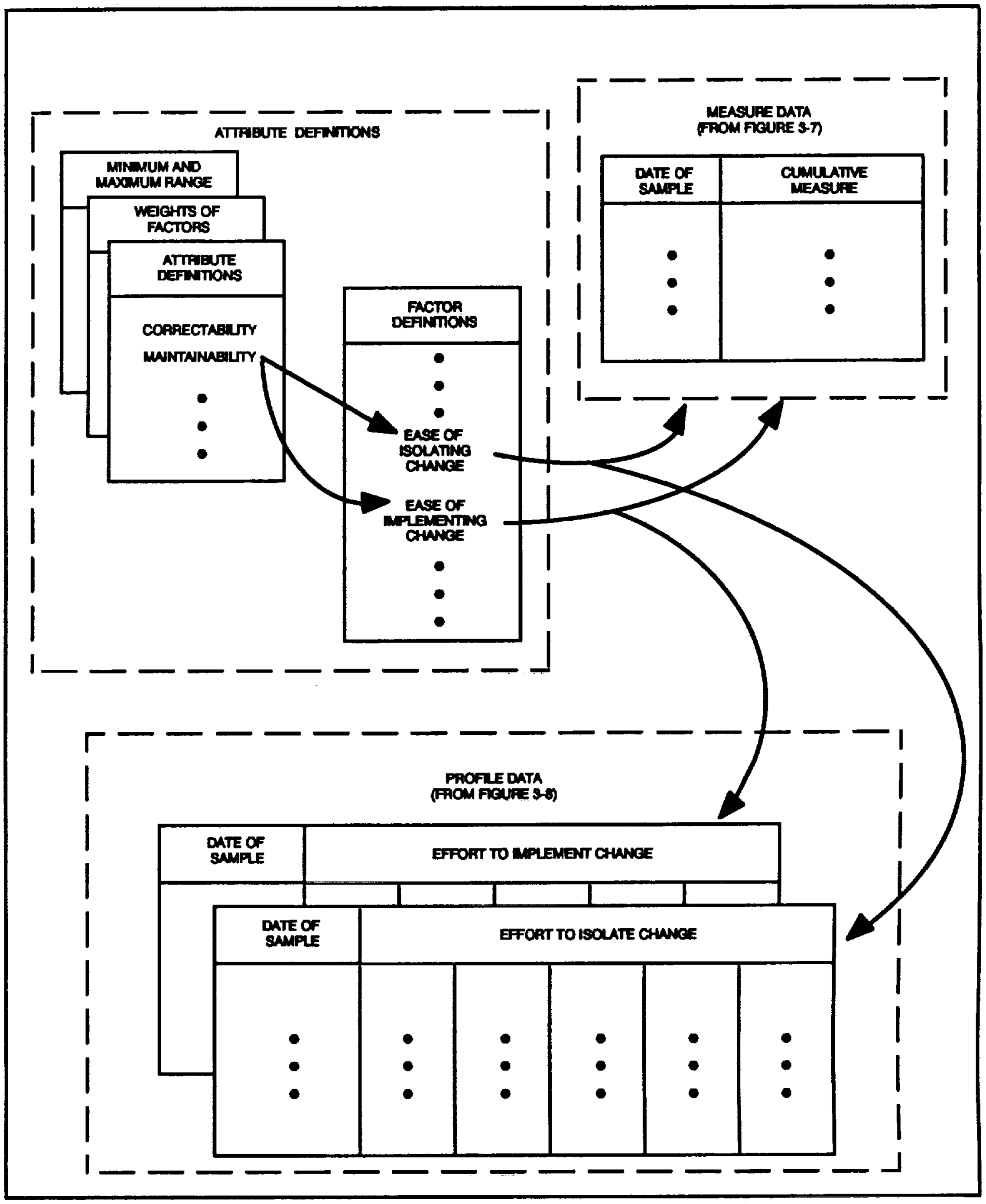


**Figure 3-17. Attribute Definitions**

### 3.1.3 Rules of Managing Software Development

The SME uses softwâre management rules that have been gathered by the SEL in various ways. Some research efforts have worked directly with experienced managers to discover the rules experts use in practical management of software development. Other rules result from studies that have investigated trends and relationships within the SEL database.

Since 1984, the SEL has been performing research in the area of capturing the knowledge and experience of managers. By obtaining managers' "rules-of-thumb" and experience at evaluating projects, the SME should be able to provide expert assistance at diagnosing project problems and suggesting actions to correct those problems.

The SME uses a set of rules that captures the experience of the software development manager. These rules have been developed through the examination of past research results and through ongoing research in this area.

One past research effort involved collecting knowledge from software development managers by two different methods and combining the results (Reference 10). The first method for collecting the manager's rules was a top-down approach. In this approach, the managers examined a set of potential problems, such as "productivity is lower than expected," and they provided lists of possible reasons for each deviation. The second method was a bottom-up approach. Here the managers were given a set of problem causes and asked to list the deviations that would be observed if these things were happening on a project. From these two data collection approaches, a set of consistent software management rules was developed.

As an example, one rule that came out of this research was the following:

| PROBLEM OR DEVIATION | REASON OR EXPLANATION |
|---|---|
| ABOVE NORMAL SOFTWARE CHANGES PER LINE OF CODE NEAR THE END OF THE CODING PHASE | GOOD QUALITY TESTING OR GOOD QUALITY TEST PLAN |

Such rules are valuable in determining the causes for deviations from a typical project within an environment.

A second research study within the SEL (Reference 9) took a different approach to capturing software managers' knowledge. In this study, the researchers attempted to capture the knowledge managers use in evaluating the quality of a development effort. The strategy was to query managers on what factors directly influenced specific quality indicators and to what degree. The answers were combined into a set of factor-based rules for evaluating the quality of a development effort, as in the following example:

| FACTORS THAT AFFECT PROJECT STABILITY | WEIGHT |
|---|---|
| SPECIFICATIONS STABILITY | 0.50 |
| TEAM STABILITY | 0.30 |
| DESIGN STABILITY | 0.20 |

This rule shows three factors that affect the overall stability of a software project. The weight represents the relative importance of the factor in determining the overall stability. Of course, other rules might need to be evaluated to determine the rating of each of these three factors. Thus, a network of software management rules can be used to determine the ratings of a set of important software project quality factors.

The current research in the SME has built on these research efforts. Rules of the types described above have been combined with the empirical results of examining the data in the SEL database to develop a stable set of rules for the SME. This set of rules constitutes a knowledge base that captures the reasoning needed by the SME to analyze a project.

As an example, one of the manager's rules of the first type used by SME is as follows:

| PROBLEM OR DEVIATION | REASON OR EXPLANATION |
|---|---|
| ABOVE NORMAL SOFTWARE CHANGES | THE DEVELOPMENT TEAM IS INEXPERIENCED |

(Note that this is only one of several explanations tested by the SME as a result of this deviation.) To determine the accuracy of the conclusion, the SME uses a rule of the second type, as follows:

| FACTORS THAT AFFECT TEAM EXPERIENCE | WEIGHT |
|---|---|
| EXPERIENCE WITH APPLICATION | 0.25 |
| EXPERIENCE WITH LANGUAGE | 0.25 |
| EXPERIENCE WITH ENVIRONMENT | 0.25 |
| EXPERIENCE WITH TOOLS | 0.25 |

The ratings of the underlying factors can be determined by using subjective data supplied by the manager (Section 3.1.4.4) or by examining objective measurement data from the SEL database.

Additional ongoing research in the SME has focused on investigating an alternative data structure for capturing the management rules collected as part of past research studies. This approach views the set of rules as a series of conditions and associated explanations that constitute a rule base. By evaluating each rule in the rule base and accumulating a list of valid explanations, the SME can use the rule base to analyze a project.

For example, one of the manager's rules interpreted for use with the rule base by the SME is as follows:

| CONDITION | REASON OR EXPLANATION | WEIGHT |
|---|---|---|
| IF LATE IN CODING PHASE AND SOFTWARE CHANGES PER LINE OF CODE ARE ABOVE NORMAL | GOOD QUALITY TEST PLAN | 0.25 |
| | OR | |
| | UNSTABLE SPECIFICATIONS | 0.25 |
| | OR | |
| | ERROR-PRONE CODE | 0.50 |

The SME incorporates the knowledge base and the rule base as two independent approaches for providing expert assistance to software development managers. At the present time, the two methods for capturing management rules within the SME appear to be equally useful and valid.

The following subsections describe the structure of the knowledge base and the rule base.

### 3.1.3.1 KNOWLEDGE BASE

The knowledge base is a description of the relationships that link objective and subjective data about a project to a set of possible assessments of the project status.

The knowledge base consists of a set of tables used to evaluate and display software development rules. The primary table (the reason table) links the observed deviation of some measure to a possible cause for the deviation. The factor table is used to locate the information needed to evaluate the truth of the reason (e.g., is this project actually working on a complex problem?). The explanation table contains the message that is displayed when a reason is true.

The ratings of factors are evaluated according to several methods; the factor table is used to link the factor to its evaluation method. Some factors obtain ratings directly from the manager's subjective data (Section 3.1.4.4); other factor ratings are the result of calculations using measure data (Section 3.1.1.6). A third type of factor obtains its rating from combining the weighted ratings of other, influencing factors.

*Used by:* Trend analysis, guidance

*Source:* Research

*Structure:* Set of three tables: a factor table with two columns (factor name, location of data), a reason table with two columns (deviation, reason), and an explanation table with two columns (reason, text of explanation)

*Instances:* There is one set of knowledge base tables.

*Example:* Figure 3-18 shows the knowledge base tables with two typical links between tables. The link from the reason table to the factor table shows how the SME locates the method for evaluating the "team experience" factor. In this case the factor table indicates that other factors must be evaluated and combined to evaluate the team experience factor. (To evaluate "problem complexity," the SME would look in the subjective data for the manager's rating; to evaluate "coding productivity," the SME would use measure data.) The link from the reason table to the explanation table shows how the SME locates the message to display if a factor evaluation indicates that a reason is true.

#### 3.1.3.2 RULE BASE

The rule base identifies a set of rules that link the conditional evaluation of objective data about a project to a set of possible assessments of the project status.

The rule base consists of two related tables used to evaluate and display software development rules. The primary table (the rule table) contains a series of conditions with each condition associated to one or more possible reasons. The explanation table contains the message that is to be displayed when a reason is considered valid.

Each condition in the rule base is evaluated based on the present life cycle phase and current measure data for the project. If a condition is deemed true, the associated weighted reasons are considered valid and added to an assertion list. Attempts to duplicate a reason in the assertion list result in one entry weighted to reflect both conditions. Upon completion, the reasons in the list can be translated for display using the explanation table.

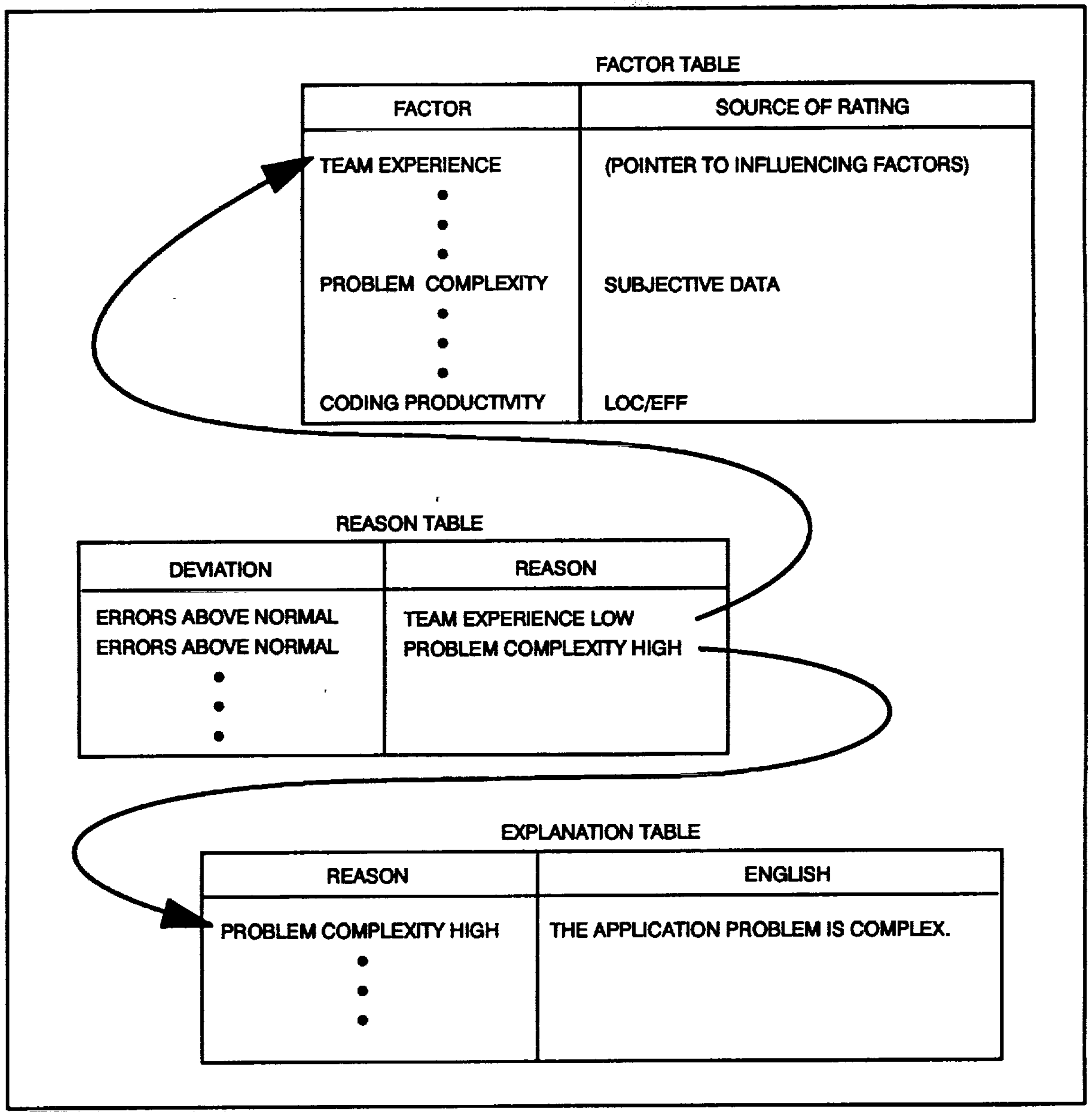
*Used by:* Trend analysis

*Source:* Research

*Structure:* Set of two tables: a rule table with two columns (condition, weighted reasons) and an explanation table with two columns (reason, text of explanation)

*Instances:* There is one rule base.

*Example:* Figure 3-19 shows how the rule base links the conditional evaluation of reasons with an English translation. Note that for clarity in the example, only one rule appears in the rule table.



FACTOR TABLE

| FACTOR | SOURCE OF RATING |
|---|---|
| TEAM EXPERIENCE | (POINTER TO INFLUENCING FACTORS) |
| • | |
| • | |
| • | |
| PROBLEM COMPLEXITY | SUBJECTIVE DATA |
| • | |
| • | |
| • | |
| CODING PRODUCTIVITY | LOC/EFF |

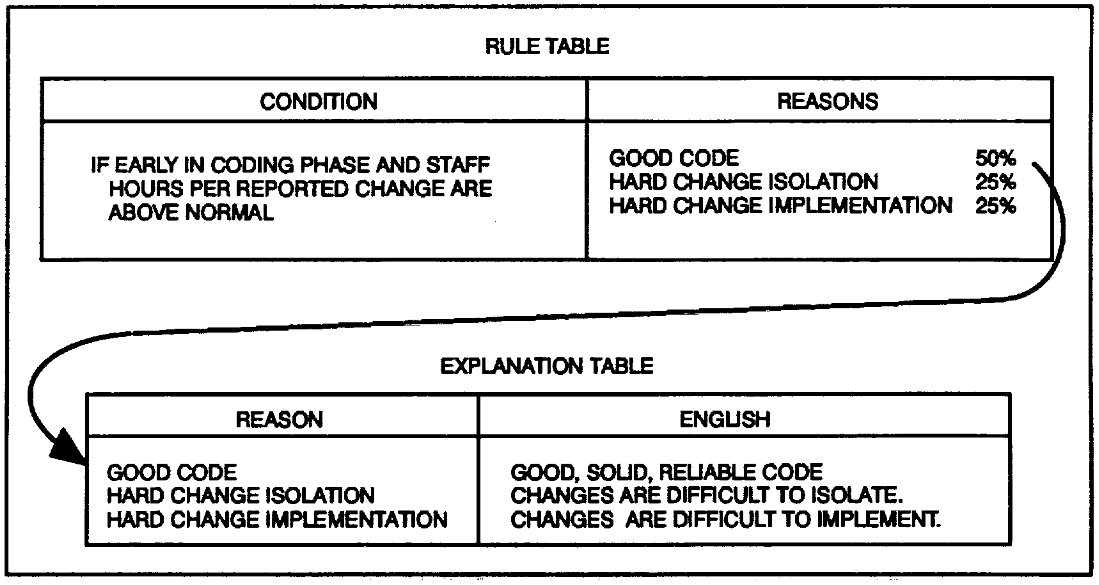
REASON TABLE

| DEVIATION | REASON |
|---|---|
| ERRORS ABOVE NORMAL | TEAM EXPERIENCE LOW |
| ERRORS ABOVE NORMAL | PROBLEM COMPLEXITY HIGH |
| • | |
| • | |
| • | |

EXPLANATION TABLE

| REASON | ENGLISH |
|---|---|
| PROBLEM COMPLEXITY HIGH | THE APPLICATION PROBLEM IS COMPLEX. |
| • | |
| • | |
| • | |

**Figure 3-18. Knowledge Base**



RULE TABLE

| CONDITION | REASONS | |
|---|---|---|
| IF EARLY IN CODING PHASE AND STAFF HOURS PER REPORTED CHANGE ARE ABOVE NORMAL | GOOD CODE | 50% |
| | HARD CHANGE ISOLATION | 25% |
| | HARD CHANGE IMPLEMENTATION | 25% |

EXPLANATION TABLE

| REASON | ENGLISH |
|---|---|
| GOOD CODE | GOOD, SOLID, RELIABLE CODE |
| HARD CHANGE ISOLATION | CHANGES ARE DIFFICULT TO ISOLATE. |
| HARD CHANGE IMPLEMENTATION | CHANGES ARE DIFFICULT TO IMPLEMENT. |

**Figure 3-19. Rule Base**

## 3.1.4 Data From the Manager

The SME is an interactive management tool that uses the SEL database for much of its input data. The information in the database is used by several applications; to ensure the integrity of the data, none of the applications, including the SME, may change data in the database. This results in a conflict when the SME requires true interactive functions in which the manager enters new or modified data for the SME to analyze.

The solution is to use some external data structures that parallel data available from the database. A look back at the data elements discussed so far reveals two elements (current schedule and estimates) that the manager might wish to modify based on new information about the project.

The manager supplies the original schedule and estimate information to the SEL database on forms. The alternative schedule and estimates discussed here provide a way for the manager to quickly modify a copy of this information and to use the SME to analyze "what if" the manager changed these project parameters.

To modify the current schedule and estimate data in the SEL database, the manager would be required to submit revised information on SEL forms. This ensures that all information in the SEL database is entered under the database quality assurance and validation procedures.

The two other types of data presented here, phase estimates and subjective data, have no counterparts in the SEL database. These two types of data have no default values; all data are entered by the manager.

### 3.1.4.1 ALTERNATIVE SCHEDULES

Alternative schedules refer to a project's schedule.

An alternative schedule has the same format and usage as the current schedule. Alternative schedules are created and modified by the manager (using the SME planning function) to investigate the effects of changing schedules. This type of schedule "belongs" to the manager, and, to eliminate the chance of confusion, each manager's alternative schedules are kept in separate areas.

*Used by:* All SME functions

*Source:* Manager

*Structure:* Table with three columns (phase name, start date, end date)

*Instances:* There can be several tables, each associated with a specific project and manager. A manager may have more than one table for a specific project.

*Example:* Figure 3-20 shows that managers may have more than one alternative schedule.

### 3.1.4.2 ALTERNATIVE ESTIMATES

Alternative estimates refer to a set of estimated completion values for a project.

An alternative set of estimates has the same format and usage as the current estimate set. Alternative estimate sets are created and modified by the manager (using the SME planning function) to investigate the effects of changing estimates. This type of estimate set "belongs" to the manager, and, to eliminate the chance of confusion, each manager's alternative estimate sets are kept in separate areas.

*Used by:* All SME functions

*Source:* Manager

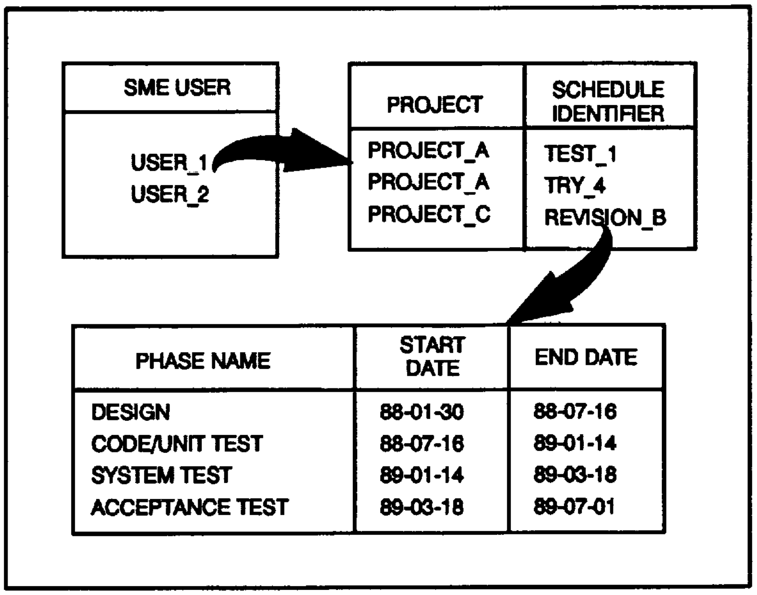
*Structure:* Table with two columns (measure name, estimated value at project completion)

*Instances:* There can be several tables, each associated with a specific project and manager. A manager may have more than one table for a specific project.

*Example:* Figure 3-21 shows that managers may have more than one alternative estimate set.

### 3.1.4.3 PHASE ESTIMATES

Phase estimates represent an historical record of all estimates of the completed fraction of a project's current development phase.

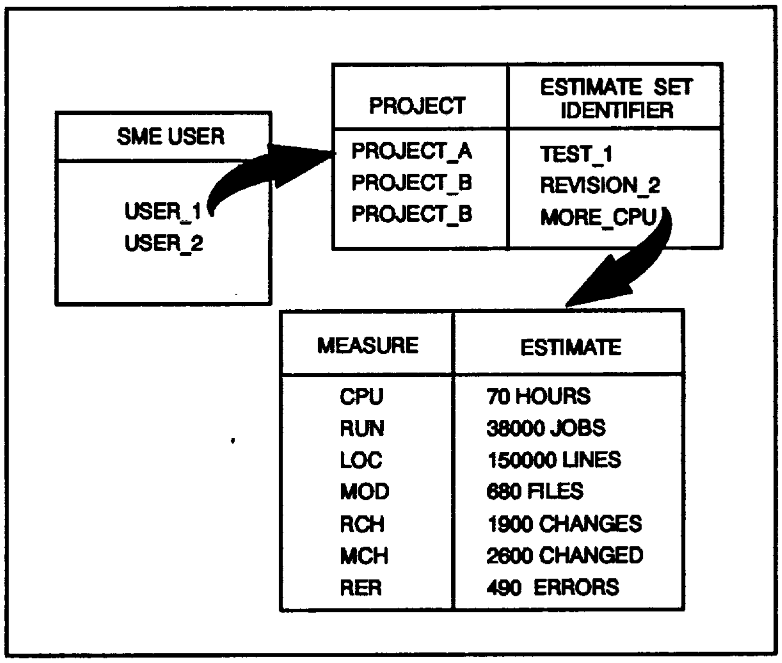


| SME USER |
|---|
| USER_1 |
| USER_2 |

| PROJECT | SCHEDULE IDENTIFIER |
|---|---|
| PROJECT_A | TEST_1 |
| PROJECT_A | TRY_4 |
| PROJECT_C | REVISION_B |

| PHASE NAME | START DATE | END DATE |
|---|---|---|
| DESIGN | 88-01-30 | 88-07-16 |
| CODE/UNIT TEST | 88-07-16 | 89-01-14 |
| SYSTEM TEST | 89-01-14 | 89-03-18 |
| ACCEPTANCE TEST | 89-03-18 | 89-07-01 |

**Figure 3-20. Alternative Schedules**



| SME USER |
|---|
| USER_1 |
| USER_2 |

| PROJECT | ESTIMATE SET IDENTIFIER |
|---|---|
| PROJECT_A | TEST_1 |
| PROJECT_B | REVISION_2 |
| PROJECT_B | MORE_CPU |

| MEASURE | ESTIMATE |
|---|---|
| CPU | 70 HOURS |
| RUN | 38000 JOBS |
| LOC | 150000 LINES |
| MOD | 680 FILES |
| RCH | 1900 CHANGES |
| MCH | 2600 CHANGED |
| RER | 490 ERRORS |

**Figure 3-21. Alternative Estimates**

A phase estimate indicates where a project is in the development life cycle on a given date. Each estimate contains the current date, the current phase, and the completed fraction of that phase. The SME obtains phase estimates as a basis for making predictions either from the manager or from an internal calculation called phase analysis. The estimates are saved by the SME to be used in later sessions to display any trend in the history of predicted values. The estimates "belong" to the manager, and, to eliminate the chance of confusion, each manager's phase estimates are kept in separate areas.

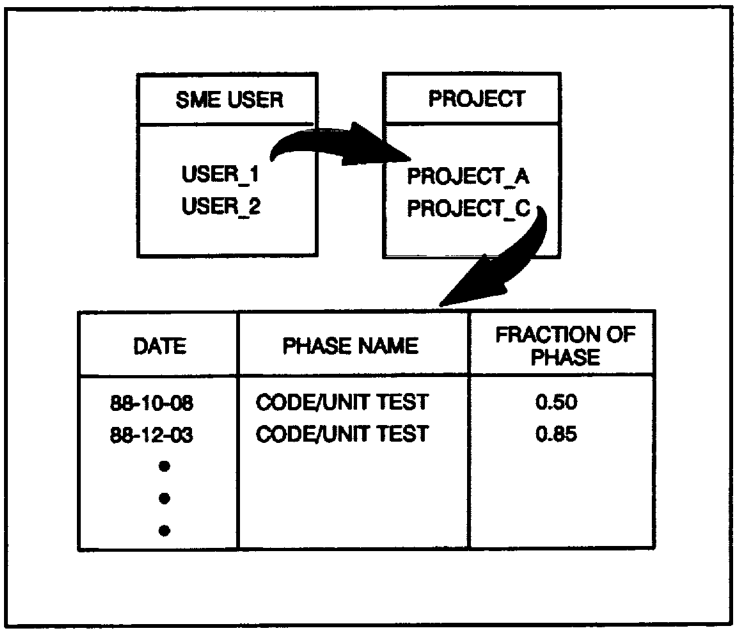
*Used by:* Prediction

*Source:* Manager

*Structure:* Table with three columns (date, phase name, fraction)

*Instances:* There can be several tables, each associated with a specific project and manager. There is only one table per project per manager.

*Example:* Figure 3-22 shows a sample list of phase estimates.



| DATE | PHASE NAME | FRACTION OF PHASE |
|---|---|---|
| 88-10-08 | CODE/UNIT TEST | 0.50 |
| 88-12-03 | CODE/UNIT TEST | 0.85 |
| • | | |
| • | | |
| • | | |

**Figure 3-22. Phase Estimates**

### 3.1.4.4 SUBJECTIVE DATA

Subjective data represent the manager's ratings of software development factors for a project.

Subjective data are collected from the manager during sessions with the trend analysis function. The data consist of the ratings (high, normal, low, unknown) of factors that affect the development process. Examples of these development factors are development team experience, problem complexity, tool usage, and computer responsiveness. The ratings are used by the expert system software in the SME according to the relationships contained in the knowledge base.

*Used by:* Trend analysis, guidance

*Source:* Manager

*Structure:* Table with two columns (factor name, rating)

*Instances:* There is one subjective data table per project.

*Example:* Figure 3-23 shows samples of subjective data tables.

## 3.2 FUNCTIONAL ARCHITECTURE

The functional architecture is a description of the software structure. The discussion includes, for each major function, a brief description of its purpose and a brief presentation of the processing involved. The functions are summarized with a list of services provided and a list of the information required by each.

Figure 3-24 shows the structure of the SME software. This structure results from the functional grouping presented in Figure 2-10. Each function is discussed separately below. The discussions refer to the data described in the presentation of the data architecture in Section 3.1.

### 3.2.1 The SME Executive

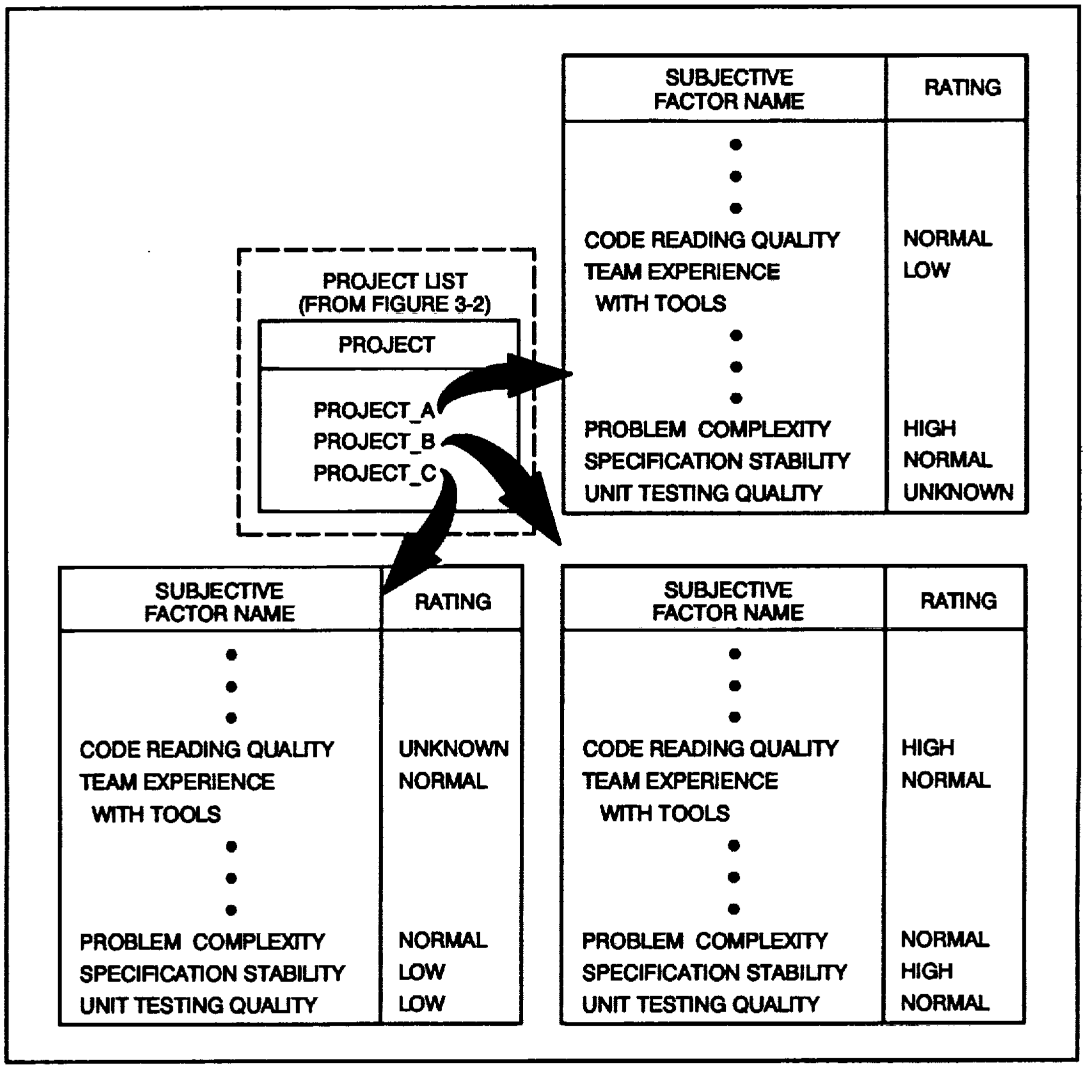
Project management typically involves focusing the manager's attention on a single project, although it may also involve comparison with previous projects and other ongoing projects. For this reason, the SME performs all of its functions within the context of a single project.

Services provided:

- Opens and closes the SME session
- Provides a list of projects from which the manager selects a project of interest
- Passes control to the monitoring, overall assessment, planning, or guidance function as requested by the manager

Information required:

- List of projects available to the SME



| SUBJECTIVE FACTOR NAME | RATING |
|---|---|
| ● | |
| ● | |
| ● | |
| CODE READING QUALITY | NORMAL |
| TEAM EXPERIENCE WITH TOOLS | LOW |
| ● | |
| ● | |
| ● | |
| PROBLEM COMPLEXITY | HIGH |
| SPECIFICATION STABILITY | NORMAL |
| UNIT TESTING QUALITY | UNKNOWN |

| SUBJECTIVE FACTOR NAME | RATING |
|---|---|
| ● | |
| ● | |
| ● | |
| CODE READING QUALITY | UNKNOWN |
| TEAM EXPERIENCE WITH TOOLS | NORMAL |
| ● | |
| ● | |
| ● | |
| PROBLEM COMPLEXITY | NORMAL |
| SPECIFICATION STABILITY | LOW |
| UNIT TESTING QUALITY | LOW |

| SUBJECTIVE FACTOR NAME | RATING |
|---|---|
| ● | |
| ● | |
| ● | |
| CODE READING QUALITY | HIGH |
| TEAM EXPERIENCE WITH TOOLS | NORMAL |
| ● | |
| ● | |
| ● | |
| PROBLEM COMPLEXITY | NORMAL |
| SPECIFICATION STABILITY | HIGH |
| UNIT TESTING QUALITY | NORMAL |

**Figure 3-23. Subjective Data**

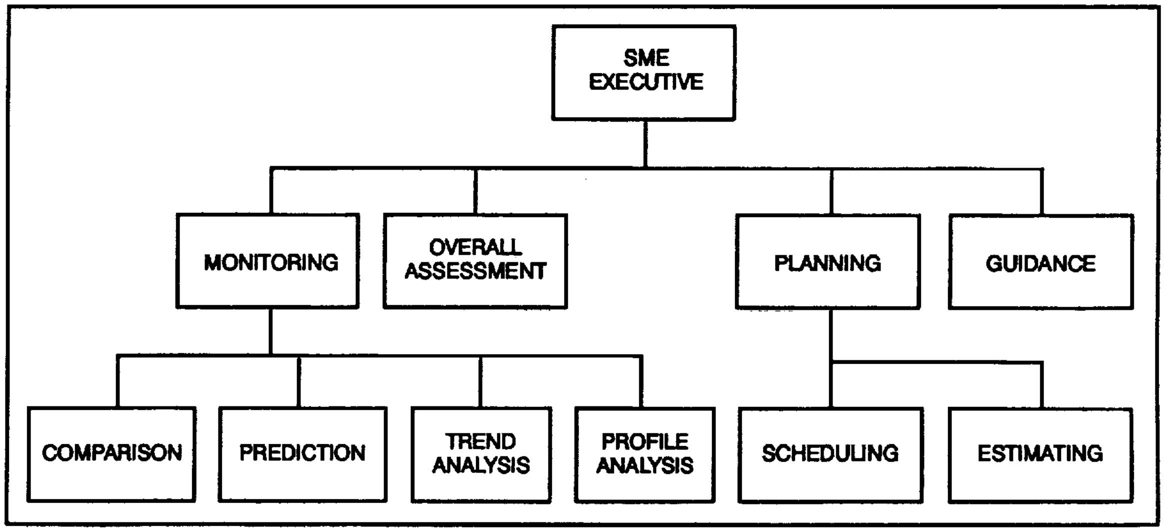


**Figure 3-24. SME Functional Architecture**

## 3.2.2 Monitoring

The SME monitoring function provides a graphic display area in which the manager views a project's measurement data. The manager selects a measure of interest and is presented with a plot of the cumulative growth of that measure. The measure of interest may be either a single measure or the ratio of two measures. The width of the plotting area is scaled to show the project schedule from start to end of the planned duration, and the height is scaled by the expected completion value of the measure of interest.

The monitoring function requires only limited keyboard input from the manager. All monitoring activities are selected from Lotus-like function menus, and the manager indicates choices such as measurement type or comparison project by selecting an item from prepared lists.

The monitor function displays only one type of measurement data at a time (such as total staff effort or lines of code per hour) in the display area. The manager can superimpose guidelines, data from comparable projects, predictions of the future behavior of the measure, and probable causes for the deviations of the measure onto the display area.

Services provided:

- Provides a list of measures that are available for the project of interest from which the manager selects a measure of interest
- Displays the cumulative values of the measure of interest for the project as a function of planned schedule
- Passes control to the comparison, prediction, trend analysis, or profile analysis function as requested by the manager

Information required:

- List of measures available for the project of interest
- Schedule for the project of interest
- Expected completion value for the measure of interest
- Measure data for the measure of interest

### 3.2.2.1 COMPARISON

The comparison function adds several types of plots to the monitor's graphic display area. The manager can request a display of the guidelines for the measure of interest or a display of data from other projects.

Data that are added to the display area are always scaled to match the measurement data for the project of interest. This is done by linearly scaling each new plot to force it to start at the project of interest's start and to end at the expected completion value for the project of interest.

The guideline curve represents the expected or normal growth path for the historical measure. The model for the measure of interest is used to generate this curve.

The scaling performed on data from another project uses the expected completion value for the comparison project.

Services provided:

- Displays guidelines for the measure of interest
- Provides a list of projects from which the manager selects a comparison project
- Scales and displays data from other projects

Information required:

- Model for the measure of interest
- List of projects with data available for the measure of interest
- Schedule for the comparison project
- Expected completion value for the measure of interest for the comparison project
- Measurement data for the measure of interest for the comparison project

### 3.2.2.2 PREDICTION

The prediction function adds a plot of the probable future behavior of the measure of interest to the display area.

The prediction plot is produced by displaying the model of the measure of interest. Unlike the comparison function, where plots are scaled to the manager's completion estimate, the model is forced to pass through the current value of the measure of interest. The scaling factor is calculated from an estimate of the current life-cycle phase of the project. For example, if the phase estimate is "halfway through the coding phase" and the model shows that one-third of the measure is normally observed at that time, the displayed plot of the model is scaled to reach three times the current value of the measure of interest at project completion.

The estimate of the project's current life-cycle phase is obtained from the manager or from a subfunction called phase analysis. Phase analysis calculates the phase at which a measure usually attains its current value. The average phase determined by applying phase analysis to all available measures is an alternative to a manager's estimate.

Services provided:

- Obtains an estimate of the current life-cycle phase of the project of interest
- Displays the predicted path for the growth curve from the current observed value to the predicted completion value; rescales the display area if the predicted value falls outside the current display area limits

Information required:

- Model for the measure of interest
- Current value of the measure of interest
- Estimate of the life-cycle phase (from the manager or from phase analysis)
- Model of each measure (for phase analysis only)
- Current value of each measure (for phase analysis only)
- Expected completion value for each measure (for phase analysis only)

#### 3.2.2.3 TREND ANALYSIS

The trend analysis function analyzes the trend of the measure of interest. If the current value is not on the expected growth path toward the expected completion value, a list of the probable causes of the deviation is presented. The manager can request a more detailed analysis of how the list was determined.

Trend analysis uses all subjective, characteristic, and measured information available for the project. Although the function lists only the causes for the deviation in the measure of interest, the trends of all measures are considered in the analysis.

The analysis compares the current value of a measure to the model of the measure and determines if the value is within an acceptable range of its expected value. A deviation (outside of the acceptable range) that is detected in the measure of interest causes the trend analysis function to use the knowledge base (Section 3.1.3.1) or the rule base (Section 3.1.3.2) to generate a list of possible reasons for the deviation.

The trend analysis function also helps the manager understand how the displayed list of possible reasons was determined. The ratings of the factors that were considered by the function can be displayed. If the knowledge base was used and the displayed factor gets its rating from the manager's subjective data (Section 3.1.4.4), the manager has an opportunity to supply or modify its rating.

Services provided:

- Determines whether the measure of interest is deviating from the expected behavior for the measure
- Presents a list of probable causes for an observed deviation
- Collects and processes subjective information from the manager about the project (for knowledge base only)
- Helps the manager explore the reasoning structure to explain why the selected causes are probable

Information required:

- Model for the measure of interest
- Expected completion value for the measure of interest
- Schedule for the project of interest
- Current value of the measure of interest
- Knowledge base
- Rule base
- Subjective data (for knowledge base only)
- Model of other measure types
- Current value of other measures
- Expected completion value of other measures

### 3.2.2.4 PROFILE ANALYSIS

The profile analysis function displays a detailed breakdown of the data for the measure of interest into discrete categories. The manager indicates the profile to use for viewing the measure by selecting an available profile defined for the measure from a list.

The profile display is depicted as a bar graph of the current value, expected value, and projected completion value in each category defined by the profile for the measure. The expected values and projected completion values for the profile categories are derived by applying the model of the profile to the expected completion value of the measure.

Analyzing the distribution of measure values via profiles can help the manager detect problems and identify improvement areas.

Services provided:

- Provides a list of profiles available for the measure of interest from which the manager selects a profile
- Displays a bar graph of the current value, expected value, and projected completion value in each category defined for the selected profile

Information required:

- List of profiles available for the measure of interest for the project
- Current data value for each category in the selected profile
- Model of the selected profile
- Expected completion value for the measure of interest

### 3.2.3 Overall Assessment

The SME overall assessment function provides an evaluation of the quality of the project. This function differs from trend analysis in that it does not concentrate on a deviation in a single measure of interest. Instead, a fixed list of overall project quality attributes is evaluated.

The evaluation includes ratings of quality attributes such as maintainability, correctability, and reliability. The evaluation uses objective measurement data collected for the project to produce a rating for each quality attribute. The ratings inform the manager of significant overall trends in the development process.

The manager requesting a more detailed analysis can investigate the reasons that the SME computed a particular attribute rating. The SME displays the underlying factors used to compute the quality rating and provides the manager with a look at the data that contributed to the rating. The reasons given for computing the rating can help the manager determine potential courses of action to improve project quality.

Services provided:

- Evaluates overall project quality attributes
- Helps the manager explore the reasoning structure to show why the attributes were rated as they were

Information required:

- Model of each measure type
- Current value of each measure
- Expected completion value of each measure
- Model of each profile type
- Current values for each profile
- Schedule for the project of interest

### 3.2.4 Planning

The SME planning function provides the manager with the facilities to select, create, and modify alternative plans. Each alternative plan "belongs" to the manager and may be stored for use in later SME sessions.

An alternative plan consists of a set of completion estimates and a schedule. The manager uses these plans in "what if" scenarios to explore the effects of altering the estimates of final project size or cost or of changing the schedule.

The selected alternative plan is used by the monitoring, overall assessment, and guidance functions. The manager sees the results of using the alternatives by reexecuting these functions.

The manager creates or modifies estimates and schedules by selecting the appropriate editor from this function.

Services provided:

- Provides lists of alternative plans from which the manager selects alternatives
- Passes control to the estimate editor or the schedule editor as requested by the manager
- Stores new or modified plans for subsequent use as requested by the manager

Information required:

- List of alternative plans available for the project

#### 3.2.4.1 ESTIMATE EDITOR

The estimate editor provides assistance with creating or modifying a set of completion estimates for the project of interest.

On entering the estimate editor, the manager is provided with default estimate values for each measure from the set of current estimates (Section 3.1.1.9).

To update the default values, the manager can simply edit as few or as many of the completion estimates as desired. The editor displays the original estimate values next to any modified values.

The manager can also select an editor option to create a new set of completion estimates based on models. With this method, the manager supplies a basic project parameter such as the expected number of subsystems, lines of code, or staff hours. The editor uses analytical models of relationships such as errors per line of code or staff hours per module to generate a set of new estimate values. If desired, the manager may adjust the new completion estimates by editing the individual values as described above.

Services provided:

- Creates an estimate set from the analytical models
- Modifies a copy of the current estimate set

Information required:

- Analytical models for the development environment
- Current estimate set

### 3.2.4.2 SCHEDULE EDITOR

The schedule editor provides assistance with creating or modifying the schedule for the project of interest.

On entering the schedule editor, the manager is provided with a default schedule containing phase dates initialized from the current schedule (Section 3.1.1.8).

To update the default schedule, the manager simply edits the phase dates as desired. The editor examines the edited schedule to ensure that only contiguous, nonoverlapping phases are specified.

The manager can also select an editor option to create a new schedule based on a model. With this method, the manager supplies the start and end dates for the project. The editor uses a schedule model as a template to determine phase transition dates for a new schedule that matches the manager's specified duration. If desired, the manager may adjust the schedule by editing the phase dates as described above.

Services provided:

- Creates a schedule using the standard schedule model
- Modifies the current schedule

Information required:

- Schedule model for development environment
- Current schedule

### 3.2.5 Guidance

The SME guidance function provides the manager with assistance in selecting the appropriate response to problems detected by the trend analysis and overall assessment functions. The guidance is based on past experience with development projects in the environment.

The guidance function starts with the results of trend analysis and overall assessment: the complete list of probable causes of deviations in the measures and the list of low-rated quality attributes, respectively. These observations are input to an expert system that attempts to find a common source for them.

The system "connects" diverse observations by finding common themes. For example, observations such as "low reliability," "not enough system testing," and "lines of code below normal" all have an inexperienced team as a common theme. The guidance function points out that the manager would benefit most from adding an experienced programmer to the team.

Services provided:

- Presents a list of factors common to a majority of the trend analysis and overall assessment observations and suggestions for changing the factors

Information required:

- Schedule for the project of interest
- Knowledge base
- Subjective data
- Model of each measure type
- Current value of each measure
- Expected completion value of each measure
- Model of each profile type
- Current values for each profile

## 3.3 HARDWARE ARCHITECTURE

The SME hardware architecture is a description of how the processing and data are distributed among the hardware elements available to the SME. There are few restrictions imposed on this distribution by the SME requirements. As a result, the configuration described here is one of several that could have been adopted.

The Systems Technology Laboratory (STL) environment consists of VAX minicomputers accessed by VT220 terminals or PC workstations emulating VT220 terminals. Figure 3-25 shows the hardware configuration adopted for the SME. The majority of the SME software and all of the SME data reside on an STL VAX. A communications and graphics program is employed as a user interface on the PC workstations.

This configuration provides all managers with common access to the data in the corporate memory (SEL database). It ensures that managers at all levels are using the same up-to-date data when examining a project. Placing some data on local storage at the PC workstations was considered but rejected for reasons of simplicity. For the same reason, the SME program also executes all SME functions on the VAX.

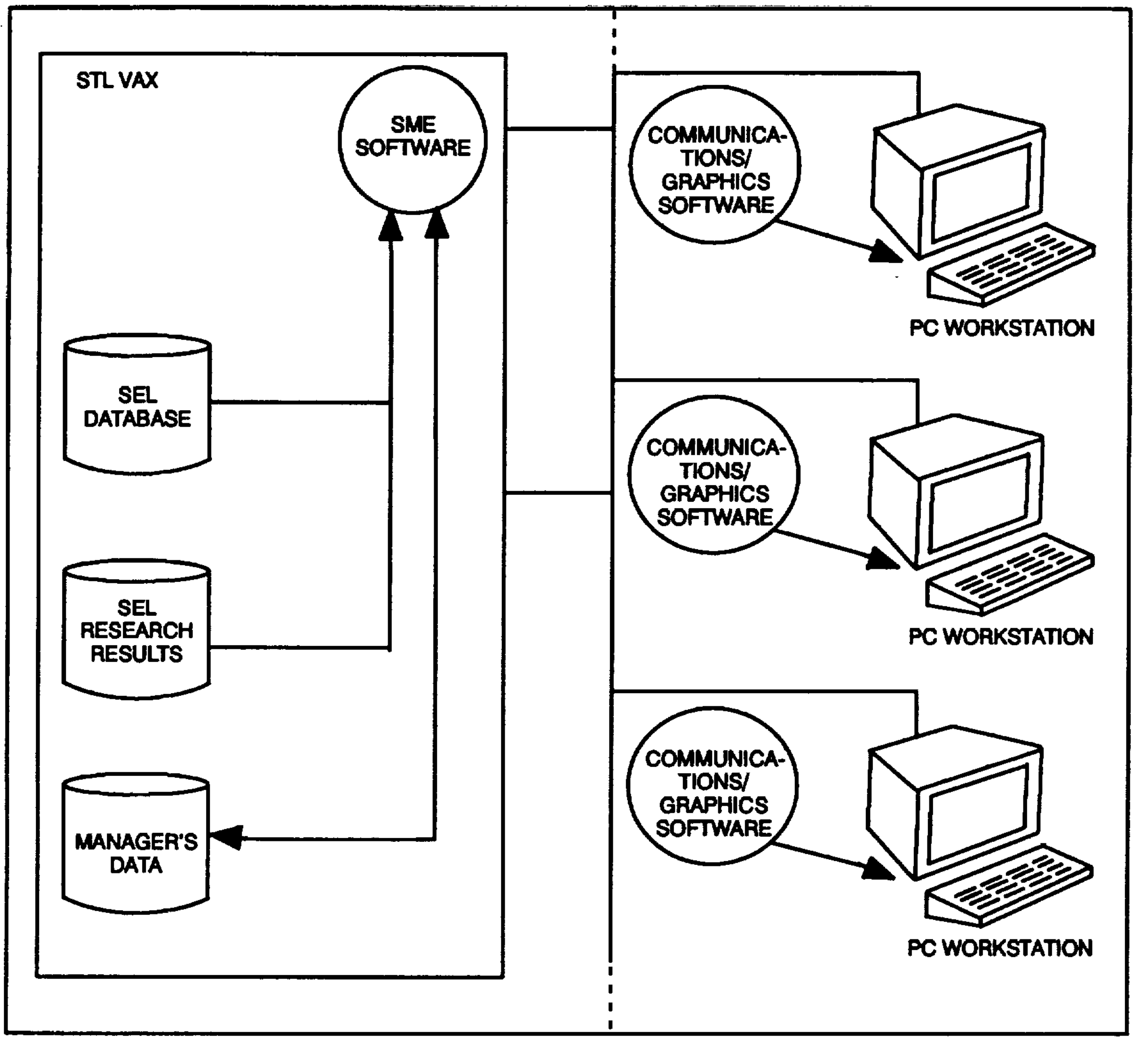


**Figure 3-25. SME Hardware Architecture**

Probably the key benefit of the PC workstation for the SME is accessibility: it sits right on the manager's desk. SME functions are substitutes for many of the manager's current office activities, so making the SME available to the manager in the office simply makes sense.

The hardware components used by the SME are not likely to change as the SME evolves; a common source of data and a remote access device with graphics capabilities are central to the concept of the SME.

## REFERENCES

1. SEL-81-104, *The Software Engineering Laboratory*, D. N. Card, F. E. McGarry, G. Page, et al., February 1982
2. V. Basili et al., "The Software Engineering Laboratory—An Operational Software Experience Factory," *Proceedings of the 14th International Conference on Software Engineering*. New York: IEEE Computer Society Press, May 1992
3. J. Valett, "The Dynamic Management Information Tool (DYNAMITE): Analysis of Prototype, Requirements, and Operational Scenarios," Master's Thesis, University of Maryland, May 1987
4. SEL-81-101, *Guide to Data Collection*, V. E. Church, D. N. Card, and F. E. McGarry, August 1982
5. SEL-92-002, *Data Collection Procedures for the Software Engineering Laboratory (SEL) Database*, G. Heller, J. Valett, and M. Wild, March 1992
6. SEL-91-001, *Software Engineering Laboratory (SEL) Relationships, Models, and Management Rules*, W. J. Decker, R. Hendrick, and J. Valett, February 1991
7. V. Basili and M. Zelkowitz, "Measuring Software Development Characteristics in the Local Environment," *Computers and Structures*, vol. 10, August 1978
8. R. Selby and V. Basili, "Comparing the Effectiveness of Software Testing Strategies," *IEEE Transactions on Software Engineering*, December 1987
9. J. Valett and A. Raskin, "DEASEL: An Expert System for Software Engineering," *Proceedings of the Tenth Annual Software Engineering Workshop*, SEL-85-006, December 1985
10. University of Maryland, Technical Report TR-1708, "An Evaluation of Expert Systems for Software Engineering Management," C. Ramsey and V. Basili, September 1986
11. SEL-89-101, *Software Engineering Laboratory (SEL) Database Organization and User's Guide (Revision 1)*, M. So, G. Heller, S. Steinberg, K. Pumphrey, and D. Spiegel, February 1990
12. V. Basili and M. V. Zelkowitz, "Designing a Software Measurement Experiment," *Proceedings of the Software Life Cycle Management Workshop*, September 1977
13. SEL-83-001, *An Approach to Cost Estimation*, F. E. McGarry, G. Page, D. N. Card, et al., February 1984
14. SEL-79-002, *The Software Engineering Laboratory: Relationship Equations*, K. Freburger and V. R. Basili, May 1979

# STANDARD BIBLIOGRAPHY OF SEL LITERATURE

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

## SEL-ORIGINATED DOCUMENTS

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

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

SEL-77-004, *A Demonstration of AXES for NAVPAK*, M. Hamilton and S. Zeldin, September 1977

SEL-77-005, *GSFC NAVPAK Design Specifications Languages Study*, P. A. Scheffer and C. E. Velez, October 1977

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

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

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

SEL-78-302, *FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 3)*, W. J. Decker, W. A. Taylor, et al., July 1986

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

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

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

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

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

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

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

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

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

SEL-80-008, *Tutorial on Models and Metrics for Software Management and Engineering*, V. R. Basili, 1980

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

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

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

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

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

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

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

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

SEL-81-107, *Software Engineering Laboratory (SEL) Compendium of Tools (Revision 1)*, W. J. Decker, W. A. Taylor, E. J. Smith, et al., February 1982

SEL-81-110, *Evaluation of an Independent Verification and Validation (IV&V) Methodology for Flight Dynamics*, G. Page, F. E. McGarry, and D. N. Card, June 1985

SEL-81-305, *Recommended Approach to Software Development*, L. Landis, F. E. McGarry, S. Waligora, et al., June 1992

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

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

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

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

SEL-82-102, *FORTRAN Static Source Code Analyzer Program (SAP) System Description (Revision 1)*, W. A. Taylor and W. J. Decker, April 1985

SEL-82-105, *Glossary of Software Engineering Laboratory Terms*, T. A. Babst, M. G. Rohleder, and F. E. McGarry, October 1983

SEL-82-1006, *Annotated Bibliography of Software Engineering Laboratory Literature*, L. Morusiewicz and J. Valett, November 1991

SEL-83-001, *An Approach to Software Cost Estimation*, F. E. McGarry, G. Page, D. N. Card, et al., February 1984

SEL-83-002, *Measures and Metrics for Software Development*, D. N. Card, F. E. McGarry, G. Page, et al., March 1984

SEL-83-003, *Collected Software Engineering Papers: Volume II*, November 1983

SEL-83-006, *Monitoring Software Development Through Dynamic Variables*, C. W. Doerflinger, November 1983

SEL-83-007, *Proceedings of the Eighth Annual Software Engineering Workshop*, November 1983

SEL-83-106, *Monitoring Software Development Through Dynamic Variables (Revision 1)*, C. W. Doerflinger, November 1989

SEL-84-003, *Investigation of Specification Measures for the Software Engineering Laboratory (SEL)*, W. W. Agresti, V. E. Church, and F. E. McGarry, December 1984

SEL-84-004, *Proceedings of the Ninth Annual Software Engineering Workshop*, November 1984

SEL-84-101, *Manager's Handbook for Software Development (Revision 1)*, L. Landis, F. E. McGarry, S. Waligora, et al., November 1990

SEL-85-001, *A Comparison of Software Verification Techniques*, D. N. Card, R. W. Selby, Jr., F. E. McGarry, et al., April 1985

SEL-85-002, *Ada Training Evaluation and Recommendations From the Gamma Ray Observatory Ada Development Team*, R. Murphy and M. Stark, October 1985

SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985

SEL-85-004, *Evaluations of Software Technologies: Testing, CLEANROOM, and Metrics*, R. W. Selby, Jr., and V. R. Basili, May 1985

SEL-85-005, *Software Verification and Testing*, D. N. Card, E. Edwards, F. McGarry, and C. Antle, December 1985

SEL-85-006, *Proceedings of the Tenth Annual Software Engineering Workshop*, December 1985

SEL-86-001, *Programmer's Handbook for Flight Dynamics Software Development*, R. Wood and E. Edwards, March 1986

SEL-86-002, *General Object-Oriented Software Development*, E. Seidewitz and M. Stark, August 1986

SEL-86-003, *Flight Dynamics System Software Development Environment (FDS/SDE) Tutorial*, J. Buell and P. Myers, July 1986

SEL-86-004, *Collected Software Engineering Papers: Volume IV*, November 1986

SEL-86-005, *Measuring Software Design*, D. N. Card et al., November 1986

SEL-86-006, *Proceedings of the Eleventh Annual Software Engineering Workshop*, December 1986

SEL-87-001, *Product Assurance Policies and Procedures for Flight Dynamics Software Development*, S. Perry et al., March 1987

SEL-87-002, *Ada® Style Guide (Version 1.1)*, E. Seidewitz et al., May 1987

SEL-87-003, *Guidelines for Applying the Composite Specification Model (CSM)*, W. W. Agresti, June 1987

SEL-87-004, *Assessing the Ada® Design Process and Its Implications: A Case Study*, S. Godfrey, C. Brophy, et al., July 1987

SEL-87-009, *Collected Software Engineering Papers: Volume V*, November 1987

SEL-87-010, *Proceedings of the Twelfth Annual Software Engineering Workshop*, December 1987

SEL-88-001, *System Testing of a Production Ada Project: The GRODY Study*, J. Seigle, L. Esker, and Y. Shi, November 1988

SEL-88-002, *Collected Software Engineering Papers: Volume VI*, November 1988

SEL-88-003, *Evolution of Ada Technology in the Flight Dynamics Area: Design Phase Analysis*, K. Quimby and L. Esker, December 1988

SEL-88-004, *Proceedings of the Thirteenth Annual Software Engineering Workshop*, November 1988

SEL-88-005, *Proceedings of the First NASA Ada User's Symposium*, December 1988

SEL-89-002, *Implementation of a Production Ada Project: The GRODY Study*, S. Godfrey and C. Brophy, September 1989

SEL-89-004, *Evolution of Ada Technology in the Flight Dynamics Area: Implementation/ Testing Phase Analysis*, K. Quimby, L. Esker, L. Smith, M. Stark, and F. McGarry, November 1989

SEL-89-005, *Lessons Learned in the Transition to Ada From FORTRAN at NASA/ Goddard*, C. Brophy, November 1989

SEL-89-006, *Collected Software Engineering Papers: Volume VII*, November 1989

SEL-89-007, *Proceedings of the Fourteenth Annual Software Engineering Workshop*, November 1989

SEL-89-008, *Proceedings of the Second NASA Ada Users' Symposium*, November 1989

SEL-89-101, *Software Engineering Laboratory (SEL) Database Organization and User's Guide (Revision 1)*, M. So, G. Heller, S. Steinberg, K. Pumphrey, and D. Spiegel, February 1990

SEL-89-103, *Software Management Environment (SME) Concepts and Architecture (Revision 1)*, R. Hendrick, D. Kistler, and J. Valett, September 1992

SEL-90-001, *Database Access Manager for the Software Engineering Laboratory (DAMSEL) User's Guide*, M. Buhler, K. Pumphrey, and D. Spiegel, March 1990

SEL-90-002, *The Cleanroom Case Study in the Software Engineering Laboratory: Project Description and Early Analysis*, S. Green et al., March 1990

SEL-90-003, *A Study of the Portability of an Ada System in the Software Engineering Laboratory (SEL)*, L. O. Jun and S. R. Valett, June 1990

SEL-90-004, *Gamma Ray Observatory Dynamics Simulator in Ada (GRODY) Experiment Summary*, T. McDermott and M. Stark, September 1990

SEL-90-005, *Collected Software Engineering Papers: Volume VIII*, November 1990

SEL-90-006, *Proceedings of the Fifteenth Annual Software Engineering Workshop*, November 1990

SEL-91-001, *Software Engineering Laboratory (SEL) Relationships, Models, and Management Rules*, W. Decker, R. Hendrick, and J. Valett, February 1991

SEL-91-003, *Software Engineering Laboratory (SEL) Ada Performance Study Report*, E. W. Booth and M. E. Stark, July 1991

SEL-91-004, *Software Engineering Laboratory (SEL) Cleanroom Process Model*, S. Green, November 1991

SEL-91-005, *Collected Software Engineering Papers: Volume IX*, November 1991

SEL-91-006, *Proceedings of the Sixteenth Annual Software Engineering Workshop*, December 1991

SEL-91-102, *Software Engineering Laboratory (SEL) Data and Information Policy (Revision 1)*, F. McGarry, August 1991

SEL-92-001, *Software Management Environment (SME) Installation Guide*, D. Kistler, January 1992

SEL-92-002, *Data Collection Procedures for the Software Engineering Laboratory (SEL) Database*, G. Heller, March 1992

## SEL-RELATED LITERATURE

[4]Agresti, W. W., V. E. Church, D. N. Card, and P. L. Lo, "Designing With Ada for Satellite Simulation: A Case Study," *Proceedings of the First International Symposium on Ada for the NASA Space Station*, June 1986

[2]Agresti, W. W., F. E. McGarry, D. N. Card, et al., "Measuring Software Technology," *Program Transformation and Programming Environments*. New York: Springer-Verlag, 1984

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

[8]Bailey, J. W., and V. R. Basili, "Software Reclamation: Improving Post-Development Reusability," *Proceedings of the Eighth Annual National Conference on Ada Technology*, March 1990

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

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

[3]Basili, V. R., "Quantitative Evaluation of Software Methodology," *Proceedings of the First Pan-Pacific Computer Conference*, September 1985

[7]Basili, V. R., *Maintenance = Reuse-Oriented Software Development*, University of Maryland, Technical Report TR-2244, May 1989

[7]Basili, V. R., *Software Development: A Paradigm for the Future*, University of Maryland, Technical Report TR-2263, June 1989

[8]Basili, V. R., "Viewing Maintenance of Reuse-Oriented Software Development," *IEEE Software*, January 1990

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

[9]Basili, V. R., and G. Caldiera, *A Reference Architecture for the Component Factory*, University of Maryland, Technical Report TR-2607, March 1991

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

[3]Basili, V. R., and N. M. Panlilio-Yap, "Finding Relationships Between Effort and Other Variables in the SEL," *Proceedings of the International Computer Software and Applications Conference*, October 1985

[4]Basili, V. R., and D. Patnaik, *A Study on Fault Prediction and Reliability Assessment in the SEL Environment*, University of Maryland, Technical Report TR-1699, August 1986

[2]Basili, V. R., and B. T. Perricone, "Software Errors and Complexity: An Empirical Investigation," *Communications of the ACM*, January 1984, vol. 27, no. 1

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

[3]Basili, V. R., and C. L. Ramsey, "ARROWSMITH-P—A Prototype Expert System for Software Engineering Management," *Proceedings of the IEEE/MITRE Expert Systems in Government Symposium*, October 1985

Basili, V. R., and J. Ramsey, *Structural Coverage of Functional Testing*, University of Maryland, Technical Report TR-1442, September 1984

Basili, V. R., and R. Reiter, "Evaluating Automatable Measures for Software Development," *Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity, and Cost*. New York: IEEE Computer Society Press, 1979

[5]Basili, V. R., and H. D. Rombach, "Tailoring the Software Process to Project Goals and Environments," *Proceedings of the 9th International Conference on Software Engineering*, March 1987

[5]Basili, V. R., and H. D. Rombach, "T A M E: Tailoring an Ada Measurement Environment," *Proceedings of the Joint Ada Conference*, March 1987

[5]Basili, V. R., and H. D. Rombach, "T A M E: Integrating Measurement Into Software Environments," University of Maryland, Technical Report TR-1764, June 1987

[6]Basili, V. R., and H. D. Rombach, "The TAME Project: Towards Improvement-Oriented Software Environments," *IEEE Transactions on Software Engineering*, June 1988

[7]Basili, V. R., and H. D. Rombach, *Towards A Comprehensive Framework for Reuse: A Reuse-Enabling Software Evolution Environment*, University of Maryland, Technical Report TR-2158, December 1988

[8]Basili, V. R., and H. D. Rombach, *Towards A Comprehensive Framework for Reuse: Model-Based Reuse Characterization Schemes*, University of Maryland, Technical Report TR-2446, April 1990

[9]Basili, V. R., and H. D. Rombach, *Support for Comprehensive Reuse*, University of Maryland, Technical Report TR-2606, February 1991

[3]Basili, V. R., and R. W. Selby, Jr., "Calculation and Use of an Environment's Characteristic Software Metric Set," *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985

Basili, V. R., and R. W. Selby, Jr., *Comparing the Effectiveness of Software Testing Strategies*, University of Maryland, Technical Report TR-1501, May 1985

[3]Basili, V. R., and R. W. Selby, Jr., "Four Applications of a Software Data Collection and Analysis Methodology," *Proceedings of the NATO Advanced Study Institute*, August 1985

[5]Basili, V. R., and R. Selby, "Comparing the Effectiveness of Software Testing Strategies," *IEEE Transactions on Software Engineering*, December 1987

[9]Basili, V. R., and R. W. Selby, "Paradigms for Experimentation and Empirical Studies in Software Engineering," *Reliability Engineering and System Safety*, January 1991

[4]Basili, V. R., R. W. Selby, Jr., and D. H. Hutchens, "Experimentation in Software Engineering," *IEEE Transactions on Software Engineering*, July 1986

[2]Basili, V. R., R. W. Selby, and T. Phillips, "Metric Analysis and Data Validation Across FORTRAN Projects," *IEEE Transactions on Software Engineering*, November 1983

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

[3]Basili, V. R., and D. M. Weiss, "A Methodology for Collecting Valid Software Engineering Data," *IEEE Transactions on Software Engineering*, November 1984

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

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

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

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

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

[9]Booth, E. W., and M. E. Stark, "Designing Configurable Software: COMPASS Implementation Concepts," *Proceedings of Tri-Ada 1991*, October 1991

[9]Briand, L. C., V. R. Basili, and W. M. Thomas, *A Pattern Recognition Approach for Software Engineering Data Analysis*, University of Maryland, Technical Report TR-2672, May 1991

[5]Brophy, C. E., W. W. Agresti, and V. R. Basili, "Lessons Learned in Use of Ada-Oriented Design Methods," *Proceedings of the Joint Ada Conference*, March 1987

[6]Brophy, C. E., S. Godfrey, W. W. Agresti, and V. R. Basili, "Lessons Learned in the Implementation Phase of a Large Ada Project," *Proceedings of the Washington Ada Technical Conference*, March 1988

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

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

[3]Card, D. N., "A Software Technology Evaluation Program," *Annais do XVIII Congresso Nacional de Informatica*, October 1985

[5]Card, D. N., and W. W. Agresti, "Resolving the Software Science Anomaly," *The Journal of Systems and Software*, 1987

[6]Card, D. N., and W. W. Agresti, "Measuring Software Design Complexity," *The Journal of Systems and Software*, June 1988

[4]Card, D. N., V. E. Church, and W. W. Agresti, "An Empirical Study of Software Design Practices," *IEEE Transactions on Software Engineering*, February 1986

Card, D. N., V. E. Church, W. W. Agresti, and Q. L. Jordan, "A Software Engineering View of Flight Dynamics Analysis System," Parts I and II, Computer Sciences Corporation, Technical Memorandum, February 1984

Card, D. N., Q. L. Jordan, and V. E. Church, "Characteristics of FORTRAN Modules," Computer Sciences Corporation, Technical Memorandum, June 1984

[5]Card, D. N., F. E. McGarry, and G. T. Page, "Evaluating Software Engineering Technologies," *IEEE Transactions on Software Engineering*, July 1987

[3]Card, D. N., G. T. Page, and F. E. McGarry, "Criteria for Software Modularization," *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985

[1]Chen, E., and M. V. Zelkowitz, "Use of Cluster Analysis To Evaluate Software Engineering Methodologies," *Proceedings of the Fifth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1981

[4]Church, V. E., D. N. Card, W. W. Agresti, and Q. L. Jordan, "An Approach for Assessing Software Prototypes," *ACM Software Engineering Notes*, July 1986

[2]Doerflinger, C. W., and V. R. Basili, "Monitoring Software Development Through Dynamic Variables," *Proceedings of the Seventh International Computer Software and Applications Conference*. New York: IEEE Computer Society Press, 1983

Doubleday, D., *ASAP: An Ada Static Source Code Analyzer Program*, University of Maryland, Technical Report TR-1895, August 1987 (NOTE: 100 pages long)

[6]Godfrey, S., and C. Brophy, "Experiences in the Implementation of a Large Ada Project," *Proceedings of the 1988 Washington Ada Symposium*, June 1988

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

[5]Jeffery, D. R., and V. Basili, *Characterizing Resource Data: A Model for Logical Association of Software Data*, University of Maryland, Technical Report TR-1848, May 1987

[6]Jeffery, D. R., and V. R. Basili, "Validating the TAME Resource Data Model," *Proceedings of the Tenth International Conference on Software Engineering*, April 1988

[5]Mark, L., and H. D. Rombach, *A Meta Information Base for Software Engineering*, University of Maryland, Technical Report TR-1765, July 1987

[6]Mark, L., and H. D. Rombach, "Generating Customized Software Engineering Information Bases From Software Process and Product Specifications," *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, January 1989

[5]McGarry, F. E., and W. W. Agresti, "Measuring Ada for Software Development in the Software Engineering Laboratory (SEL)," *Proceedings of the 21st Annual Hawaii International Conference on System Sciences*, January 1988

[7]McGarry, F., L. Esker, and K. Quimby, "Evolution of Ada Technology in a Production Software Environment," *Proceedings of the Sixth Washington Ada Symposium (WADAS)*, June 1989

[3]McGarry, F. E., J. Valett, and D. Hall, "Measuring the Impact of Computer Resource Quality on the Software Development Process and Product," *Proceedings of the Hawaiian International Conference on System Sciences*, January 1985

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

[3]Page, G., F. E. McGarry, and D. N. Card, "A Practical Experience With Independent Verification and Validation," *Proceedings of the Eighth International Computer Software and Applications Conference*, November 1984

[5]Ramsey, C. L., and V. R. Basili, *An Evaluation of Expert Systems for Software Engineering Management*, University of Maryland, Technical Report TR-1708, September 1986

[3]Ramsey, J., and V. R. Basili, "Analyzing the Test Process Using Structural Coverage," *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985

[5]Rombach, H. D., "A Controlled Experiment on the Impact of Software Structure on Maintainability," *IEEE Transactions on Software Engineering*, March 1987

[8]Rombach, H. D., "Design Measurement: Some Lessons Learned," *IEEE Software*, March 1990

[9]Rombach, H. D., "Software Reuse: A Key to the Maintenance Problem," *Butterworth Journal of Information and Software Technology*, January/February 1991

[6]Rombach, H. D., and V. R. Basili, "Quantitative Assessment of Maintenance: An Industrial Case Study," *Proceedings From the Conference on Software Maintenance*, September 1987

[6]Rombach, H. D., and L. Mark, "Software Process and Product Specifications: A Basis for Generating Customized SE Information Bases," *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, January 1989

[7]Rombach, H. D., and B. T. Ulery, *Establishing a Measurement Based Maintenance Improvement Program: Lessons Learned in the SEL*, University of Maryland, Technical Report TR-2252, May 1989

[6]Seidewitz, E., "Object-Oriented Programming in Smalltalk and Ada," *Proceedings of the 1987 Conference on Object-Oriented Programming Systems, Languages, and Applications*, October 1987

[5]Seidewitz, E., "General Object-Oriented Software Development: Background and Experience," *Proceedings of the 21st Hawaii International Conference on System Sciences*, January 1988

[6]Seidewitz, E., "General Object-Oriented Software Development with Ada: A Life Cycle Approach," *Proceedings of the CASE Technology Conference*, April 1988

[9]Seidewitz, E., "Object-Oriented Programming Through Type Extension in Ada 9X," *Ada Letters*, March/April 1991

[4]Seidewitz, E., and M. Stark, "Towards a General Object-Oriented Software Development Methodology," *Proceedings of the First International Symposium on Ada for the NASA Space Station*, June 1986

[9]Seidewitz, E., and M. Stark, "An Object-Oriented Approach to Parameterized Software in Ada," *Proceedings of the Eighth Washington Ada Symposium*, June 1991

[8]Stark, M., "On Designing Parametrized Systems Using Ada," *Proceedings of the Seventh Washington Ada Symposium*, June 1990

[7]Stark, M. E. and E. W. Booth, "Using Ada to Maximize Verbatim Software Reuse," *Proceedings of TRI-Ada 1989*, October 1989

[5]Stark, M., and E. Seidewitz, "Towards a General Object-Oriented Ada Lifecycle," *Proceedings of the Joint Ada Conference*, March 1987

[8]Straub, P. A., and M. V. Zelkowitz, "PUC: A Functional Specification Language for Ada," *Proceedings of the Tenth International Conference of the Chilean Computer Science Society*, July 1990

[7]Sunazuka, T., and V. R. Basili, *Integrating Automated Support for a Software Management Cycle Into the TAME System*, University of Maryland, Technical Report TR-2289, July 1989

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

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

[5]Valett, J. D., and F. E. McGarry, "A Summary of Software Measurement Experiences in the Software Engineering Laboratory," *Proceedings of the 21st Annual Hawaii International Conference on System Sciences*, January 1988

[3]Weiss, D. M., and V. R. Basili, "Evaluating Software Development by Analysis of Changes: Some Data From the Software Engineering Laboratory," *IEEE Transactions on Software Engineering*, February 1985

[5]Wu, L., V. R. Basili, and K. Reed, "A Structure Coverage Tool for Ada Software Systems," *Proceedings of the Joint Ada Conference*, March 1987

[1]Zelkowitz, M. V., "Resource Estimation for Medium-Scale Software Projects," *Proceedings of the Twelfth Conference on the Interface of Statistics and Computer Science*. New York: IEEE Computer Society Press, 1979

[2]Zelkowitz, M. V., "Data Collection and Evaluation for Experimental Computer Science Research," *Empirical Foundations for Computer and Information Science* (Proceedings), November 1982

[6]Zelkowitz, M. V., "The Effectiveness of Software Prototyping: A Case Study," *Proceedings of the 26th Annual Technical Symposium of the Washington, D. C., Chapter of the ACM*, June 1987

[6]Zelkowitz, M. V., "Resource Utilization During Software Development," *Journal of Systems and Software*, 1988

[8]Zelkowitz, M. V., "Evolution Towards Specifications Environment: Experiences With Syntax Editors," *Information and Software Technology*, April 1990

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

## NOTES:

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

[2]This article also appears in SEL-83-003, *Collected Software Engineering Papers: Volume II*, November 1983.

[3]This article also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985.

[4]This article also appears in SEL-86-004, *Collected Software Engineering Papers: Volume IV*, November 1986.

[5]This article also appears in SEL-87-009, *Collected Software Engineering Papers: Volume V*, November 1987.

[6]This article also appears in SEL-88-002, *Collected Software Engineering Papers: Volume VI*, November 1988.

[7]This article also appears in SEL-89-006, *Collected Software Engineering Papers: Volume VII*, November 1989.

[8]This article also appears in SEL-90-005, *Collected Software Engineering Papers: Volume VIII*, November 1990.

[9]This article also appears in SEL-91-005, *Collected Software Engineering Papers: Volume IX*, November 1991.

[redacted]

# REPORT DOCUMENTATION PAGE

*Form Approved*
*OMB No. 0704-0188*

**1. AGENCY USE ONLY** *(Leave blank)*

**2. REPORT DATE**
SEPTEMBER 1992

**3. REPORT TYPE AND DATES COVERED**

**4. TITLE AND SUBTITLE**
SOFTWARE MANAGEMENT ENVIRONMENT (SME) CONCEPTS AND ARCHITECTURE REVISION

**5. FUNDING NUMBERS**
SEL 89 103

**6. AUTHOR(S)**
NASA-Jon Valett; Univ. of MD-David Kistler; CSC-Robert Hendrick

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS (ES)**
Same as #6

**8. PEFORMING ORGANIZATION REPORT NUMBER**
CR189293

**9. SPONSORING / MONITORING ADGENCY NAME(S) AND ADDRESS (ES)**
National Aeronautics and Space Administration
Goddard Space Flight Center, Greenbelt, MD 20771

**10. SPONSORING / MONITORING ADGENCY REPORT NUMBER**
CR189293

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION / AVAILABILITY STATMENT**
Single copies can be obtained from: Software Engineering Branch, Code 552, GSFC, Greenbelt, MD 20771.

**12b. DISTRIBUTION CODE**

**13. ABSTRACT** *(Maximum 200 words)*

This document presents the concepts and architecture of the Software Management Environment (SME), developed for the Software Enginerring Branch (Code 552).. of the Flight Dynamic Division (FDD) of the Goddard Space Flight Center (GSFC). The SME provides an integrated set of experience-based management tools that can assist software development managers in managing and planning flight dynamics software development projects. This document provides a high-level description of the types of information required to implement such an automated management tool, and it presents an architectural framework in which a set of management services can be provided.

This document is a major revision of SEL-89-003.

**14. SUBJECT TERMS**
SEL Management, Environnment/SME,, SME Functions; Architecture-/Data, Functional Architecture, Hardware Architecture

**15. NUMBER OF PAGES**
App 50

**16. PRICE CODE**

| | | | |
|---|---|---|---|
| **17. SECURITY CLASSIFICATION OF REPORT** | **18. SECURITY CLASSIFICATION OF THIS PAGE** | **19. SECURITY CLASSIFICATION OF ABSTRACT** | **20. LIMITATION OF ABSTRACT** |
| Unclassified | Unclassified | Unclassified | UL |

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39.18
298-102