SOFTWARE ENGINEERING LABORATORY (SEL) RELATIONSHIPS, MODELS, AND MANAGEMENT RULES

FEBRUARY 1991

(NASA-TM-105513) SOFTWARE ENGINEERING LABORATORY (SEL) RELATIONSHIPS, MODELS, AND MANAGEMENT RULES (NASA) 94 p CSCL 09B

N92-18123

Unclas

G3/61 0068914

NASA

National Aeronautics and Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771
SOFTWARE ENGINEERING LABORATORY (SEL) RELATIONSHIPS, MODELS, AND MANAGEMENT RULES

FEBRUARY 1991
FOREWORD

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

NASA/GSFC, Systems Development Branch
The University of Maryland, Computer Science Department
Computer Sciences Corporation, Systems Development Operation

The goals of the SEL are (1) to understand the software development process in the GSFC environment; (2) to measure the effects 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 major contributors to this document are

William Decker (CSC)
Robert Hendrick (CSC)
Jon Valett (GSFC)

Single copies of this document can be obtained by writing to

Systems Development Branch
Code 552
Goddard Space Flight Center
Greenbelt, Maryland 20771
ABSTRACT

This document captures over 50 individual SEL research results, extracted from a review of published SEL documentation, that can be applied directly to managing software development projects. Four basic categories of results are defined and discussed—environment profiles, relationships, models, and management rules. In each category, research results are presented as a single page that summarizes the individual result, lists potential uses of the result by managers, and references the original SEL documentation where the result was found. The document serves as a concise reference summary of applicable research for SEL managers.
# Table of Contents

Section 1 — Introduction .................................................. 1-1  
Section 2 — Environment Profiles ........................................ 2-1  
Section 3 — Relationships .................................................. 3-1  
Section 4 — Models .......................................................... 4-1  
Section 5 — Management Rules ............................................. 5-1  
Glossary  
References  
Standard Bibliography of SEL Literature
List of Tables

Table

<table>
<thead>
<tr>
<th></th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>1-1</td>
<td>Index of Results</td>
<td>1-3</td>
</tr>
<tr>
<td>1-2</td>
<td>Research Results Grouped by Factor</td>
<td>1-5</td>
</tr>
</tbody>
</table>
SECTION 1—INTRODUCTION

The Software Engineering Laboratory (SEL) was established in 1977 to support research in the measurement and evaluation of the software development process (Reference 1). The SEL is a cooperative effort of the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC), Computer Sciences Corporation (CSC), and the University of Maryland. Under its sponsorship, numerous experiments have been designed and executed to study the effects of applying various tools, methodologies, and models to the software development efforts in flight dynamics applications.

The SEL has been researching and evaluating software development methodologies for over 13 years. This research has provided valuable insight into the software development process of one particular organization. By collecting detailed software development data and recording that data in a software engineering data base (References 2 and 3), the SEL has been able to characterize and understand the development process within that organization. Many of the research results that have been published by the SEL over the years can be applied directly to managing software development projects in the SEL environment.

To help promote successful software development practices in this environment, the SEL recognized the potential of providing the experience of previous projects—the data, research results, and knowledge of experienced software managers—to the managers of ongoing projects. Research efforts were undertaken to determine if and how SEL experience and data could be effectively incorporated into an automated tool to provide insight into a project. Reference 4 describes an experimental, automated tool, the Software Management Environment (SME), currently in development.

As part of the tool development, the SEL conducted a thorough review of published SEL research. This review revealed several categories of data and findings that are fundamental to the construction and use of a tool to support the management of software development projects. These results are also useful to managers outside of an automated tool; in fact, many of the results are simply formal presentations of data used by managers on a regular basis.

This document contains a representative set of the research results extracted during the review of SEL documentation that can be used by software development managers. As a result, the document can serve as a concise reference summary of applicable research for SEL managers. (Each research result lists one of the source documents, References 5 through 14, where additional detailed information may be found.)

Although this research applies to projects within the SEL, other organizations may use the results as a starting point in their efforts to understand their own environments. The results provide a fundamental set of knowledge that illustrates the types of data an organization should assemble and suggests how to use the data.
The remainder of this document describes SEL research results organized by section into four basic categories: environment profiles, relationships, models, and management rules. Section 2 describes profiles containing typical values that characterize software development projects in the SEL environment. Section 3 describes relationships between various software development measures that are essential to successful planning and estimation activities. Section 4 describes models that capture the expected behavior of software development measures over the course of the development life cycle. Section 5 describes management rules that can help in diagnosing problems and evaluating the status of software development projects.

Each section first defines and describes the result type and then presents a series of representative samples of the result type. The results appear as single-page displays that include the following information: the result type, a text description of the result, a list of management activities to which the result can be applied, the SEL document reference, and a tabular and/or graphic representation of the result.

Table 1-1 presents a complete list of all single-page results presented in the document by title with the page number of the result. Table 1-2 contains a cross-reference of the research results grouped by specific factors of interest to managers. Note that individual results may appear several times in the list if the result relates to more than one factor.
## Table 1-1. Index of Results (1 of 2)

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>TITLE OF RESULT</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result 2-1</td>
<td>PROCESS AND PRODUCT CHARACTERISTICS</td>
<td>2-2</td>
</tr>
<tr>
<td>Result 2-2</td>
<td>PERFORMANCE AND QUALITY CHARACTERISTICS</td>
<td>2-3</td>
</tr>
<tr>
<td>Result 2-3</td>
<td>EFFORT DISTRIBUTION BY STAFF CATEGORY</td>
<td>2-4</td>
</tr>
<tr>
<td>Result 2-4</td>
<td>EFFORT DISTRIBUTION BY ACTIVITY</td>
<td>2-5</td>
</tr>
<tr>
<td>Result 2-5</td>
<td>ERROR DISTRIBUTION BY ERROR CLASS</td>
<td>2-6</td>
</tr>
<tr>
<td>Result 2-6</td>
<td>ERROR DISTRIBUTION BY EFFORT TO CORRECT</td>
<td>2-7</td>
</tr>
<tr>
<td>Result 2-7</td>
<td>PAGE DISTRIBUTION BY DOCUMENT TYPE</td>
<td>2-8</td>
</tr>
<tr>
<td>Result 2-8</td>
<td>COST OF USER DOCUMENTATION</td>
<td>2-9</td>
</tr>
<tr>
<td>Result 2-9</td>
<td>COST OF REUSE</td>
<td>2-10</td>
</tr>
<tr>
<td>Result 3-1</td>
<td>EFFORT VS. LINES OF CODE</td>
<td>3-3</td>
</tr>
<tr>
<td>Result 3-2</td>
<td>EFFORT VS. MODULES</td>
<td>3-4</td>
</tr>
<tr>
<td>Result 3-3</td>
<td>DURATION VS. LINES OF CODE</td>
<td>3-5</td>
</tr>
<tr>
<td>Result 3-4</td>
<td>DURATION VS. MODULES</td>
<td>3-6</td>
</tr>
<tr>
<td>Result 3-5</td>
<td>DURATION VS. EFFORT</td>
<td>3-7</td>
</tr>
<tr>
<td>Result 3-6</td>
<td>STAFF SIZE VS. EFFORT</td>
<td>3-8</td>
</tr>
<tr>
<td>Result 3-7</td>
<td>PRODUCTIVITY VS. CODE REUSE</td>
<td>3-9</td>
</tr>
<tr>
<td>Result 3-8</td>
<td>PRODUCTIVITY VS. MODULE REUSE</td>
<td>3-10</td>
</tr>
<tr>
<td>Result 3-9</td>
<td>COMPUTER RUNS VS. LINES OF CODE</td>
<td>3-11</td>
</tr>
<tr>
<td>Result 3-10</td>
<td>COMPUTER RUNS VS. EFFORT</td>
<td>3-12</td>
</tr>
<tr>
<td>Result 3-11</td>
<td>COMPUTER TIME VS. LINES OF CODE</td>
<td>3-13</td>
</tr>
<tr>
<td>Result 3-12</td>
<td>DOCUMENTATION PAGES VS. LINES OF CODE</td>
<td>3-14</td>
</tr>
<tr>
<td>Result 3-13</td>
<td>DOCUMENTATION PAGES VS. MODULES</td>
<td>3-15</td>
</tr>
<tr>
<td>Result 3-14</td>
<td>DOCUMENTATION PAGES VS. EFFORT</td>
<td>3-16</td>
</tr>
<tr>
<td>Result 3-15</td>
<td>SIZE, EFFORT, AND SCHEDULE BY PHASE</td>
<td>3-17</td>
</tr>
<tr>
<td>Result 3-16</td>
<td>EFFORT ADJUSTMENT FOR COMPLEXITY</td>
<td>3-18</td>
</tr>
<tr>
<td>Result 3-17</td>
<td>EFFORT ADJUSTMENT FOR TEAM EXPERIENCE</td>
<td>3-19</td>
</tr>
<tr>
<td>Result 3-18</td>
<td>EFFORT ADJUSTMENT FOR SCHEDULE TYPE</td>
<td>3-20</td>
</tr>
<tr>
<td>Result 4-1</td>
<td>SCHEDULE BY PHASE</td>
<td>4-3</td>
</tr>
<tr>
<td>Result 4-2</td>
<td>EFFORT BY PHASE</td>
<td>4-4</td>
</tr>
<tr>
<td>Result 4-3</td>
<td>EFFORT ACTIVITY BY PHASE</td>
<td>4-5</td>
</tr>
<tr>
<td>Result 4-4</td>
<td>LINES OF CODE BY PHASE</td>
<td>4-6</td>
</tr>
<tr>
<td>Result 4-5</td>
<td>COMPUTER USE BY PHASE</td>
<td>4-7</td>
</tr>
<tr>
<td>Result 4-6</td>
<td>ERROR RATE IN EACH PHASE</td>
<td>4-8</td>
</tr>
<tr>
<td>Result 4-7</td>
<td>PROGRAMMER HOURS PER LINE OF CODE BY PHASE</td>
<td>4-9</td>
</tr>
<tr>
<td>Result 4-8</td>
<td>COMPUTER RUNS PER LINE OF CODE BY PHASE</td>
<td>4-10</td>
</tr>
<tr>
<td>Result 4-9</td>
<td>SOFTWARE CHANGES PER LINE OF CODE BY PHASE</td>
<td>4-11</td>
</tr>
<tr>
<td>Result 4-10</td>
<td>COMPUTER TIME PER LINE OF CODE BY PHASE</td>
<td>4-12</td>
</tr>
<tr>
<td>Result 4-11</td>
<td>PROGRAMMER HOURS PER COMPUTER RUN BY PHASE</td>
<td>4-13</td>
</tr>
<tr>
<td>Result 4-12</td>
<td>SOFTWARE CHANGES PER COMPUTER RUN BY PHASE</td>
<td>4-14</td>
</tr>
<tr>
<td>Result 4-13</td>
<td>COMPUTER TIME PER COMPUTER RUN BY PHASE</td>
<td>4-15</td>
</tr>
<tr>
<td>Result 4-14</td>
<td>PROGRAMMER HOURS PER SOFTWARE CHANGE BY PHASE</td>
<td>4-16</td>
</tr>
<tr>
<td>Result 4-15</td>
<td>COMPUTER TIME PER SOFTWARE CHANGE BY PHASE</td>
<td>4-17</td>
</tr>
<tr>
<td>Result 4-16</td>
<td>UNCERTAINTY IN EFFORT AND SIZE ESTIMATES BY PHASE</td>
<td>4-18</td>
</tr>
<tr>
<td>Result 4-17</td>
<td>TRENDS IN EFFORT TO CHANGE BY PHASE</td>
<td>4-19</td>
</tr>
<tr>
<td>Result 4-18</td>
<td>TRENDS IN CHANGES PER COMPUTER RUN BY PHASE</td>
<td>4-20</td>
</tr>
<tr>
<td>Result 4-19</td>
<td>TRENDS IN COMPUTER TIME PER CHANGE BY PHASE</td>
<td>4-21</td>
</tr>
</tbody>
</table>
### Table 1-1.  Index of Results (2 of 2)

<table>
<thead>
<tr>
<th>NUMBER</th>
<th>TITLE OF RESULT</th>
<th>PAGE</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result 5-1</td>
<td>VARIATIONS IN EFFORT TO CHANGE</td>
<td>5-3</td>
</tr>
<tr>
<td>Result 5-2</td>
<td>DEVIATIONS IN STAFFING PLAN</td>
<td>5-4</td>
</tr>
<tr>
<td>Result 5-3</td>
<td>VARIATIONS IN SIZE AND EFFORT ESTIMATES</td>
<td>5-5</td>
</tr>
<tr>
<td>Result 5-4</td>
<td>HIGH COMPUTER USE</td>
<td>5-6</td>
</tr>
<tr>
<td>Result 5-5</td>
<td>DEVIATIONS IN COMPUTER USE PER LINE OF CODE</td>
<td>5-7</td>
</tr>
<tr>
<td>Result 5-6</td>
<td>DEVIATIONS IN LINES OF CODE</td>
<td>5-8</td>
</tr>
<tr>
<td>Result 5-7</td>
<td>DEVIATIONS IN CHANGES PER LINE OF CODE</td>
<td>5-9</td>
</tr>
<tr>
<td>Result 5-8</td>
<td>DEVIATIONS IN LINES OF CODE PER STAFF HOUR</td>
<td>5-10</td>
</tr>
</tbody>
</table>
## Table 1-2. Research Results Grouped by Factor (1 of 3)

<table>
<thead>
<tr>
<th>FACTOR</th>
<th>TYPE OF RESULT</th>
<th>DESCRIPTION OF RESEARCH RESULT</th>
<th>RESULT NUMBER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Changes</td>
<td>Profile</td>
<td>CHANGE RATE DURING IMPLEMENTATION</td>
<td>Result 2-2</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>SOFTWARE CHANGES PER LINE OF CODE BY PHASE</td>
<td>Result 4-9</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>SOFTWARE CHANGES PER COMPUTER RUN BY PHASE</td>
<td>Result 4-12</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>PROGRAMMER HOURS PER SOFTWARE CHANGE BY PHASE</td>
<td>Result 4-14</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>COMPUTER TIME PER SOFTWARE CHANGE BY PHASE</td>
<td>Result 4-15</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>TRENDS IN EFFORT TO CHANGE BY PHASE</td>
<td>Result 4-17</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>TRENDS IN CHANGES PER COMPUTER RUN BY PHASE</td>
<td>Result 4-18</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>TRENDS IN COMPUTER TIME PER CHANGE BY PHASE</td>
<td>Result 4-19</td>
</tr>
<tr>
<td></td>
<td>Management Rule</td>
<td>DEVIATIONS IN CHANGES PER LINE OF CODE</td>
<td>Result 5-7</td>
</tr>
<tr>
<td>Computer Runs</td>
<td>Relationship</td>
<td>COMPUTER RUNS VS. LINES OF CODE</td>
<td>Result 3-9</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>COMPUTER RUNS VS. EFFORT</td>
<td>Result 3-10</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>COMPUTER RUNS PER LINE OF CODE BY PHASE</td>
<td>Result 4-8</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>PROGRAMMER HOURS PER COMPUTER RUN BY PHASE</td>
<td>Result 4-11</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>SOFTWARE CHANGES PER COMPUTER RUN BY PHASE</td>
<td>Result 4-12</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>COMPUTER TIME PER COMPUTER RUN BY PHASE</td>
<td>Result 4-13</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>TRENDS IN CHANGES PER COMPUTER RUN BY PHASE</td>
<td>Result 4-18</td>
</tr>
<tr>
<td></td>
<td>Management Rule</td>
<td>VARIATIONS IN EFFORT TO CHANGE</td>
<td>Result 5-1</td>
</tr>
<tr>
<td></td>
<td>Management Rule</td>
<td>DEVIATIONS IN COMPUTER USE PER LINE OF CODE</td>
<td>Result 5-4</td>
</tr>
<tr>
<td>Computer Time</td>
<td>Relationship</td>
<td>COMPUTER TIME VS. LINES OF CODE</td>
<td>Result 3-11</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>COMPUTER USE BY PHASE</td>
<td>Result 4-5</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>COMPUTER TIME PER LINE OF CODE BY PHASE</td>
<td>Result 4-10</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>COMPUTER TIMEPER COMPUTER RUN BY PHASE</td>
<td>Result 4-13</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>COMPUTER TIME PER SOFTWARE CHANGE BY PHASE</td>
<td>Result 4-15</td>
</tr>
<tr>
<td></td>
<td>Management Rule</td>
<td>VARIATIONS IN EFFORT TO CHANGE</td>
<td>Result 5-4</td>
</tr>
<tr>
<td></td>
<td>Management Rule</td>
<td>DEVIATIONS IN COMPUTER USE PER LINE OF CODE</td>
<td>Result 5-5</td>
</tr>
<tr>
<td>Documentation</td>
<td>Profile</td>
<td>PAGE DISTRIBUTION BY DOCUMENT TYPE</td>
<td>Result 2-7</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>COST OF USER DOCUMENTATION</td>
<td>Result 2-8</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>DOCUMENTATION PAGES VS. LINES OF CODE</td>
<td>Result 3-12</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>DOCUMENTATION PAGES VS. MODULES</td>
<td>Result 3-13</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>DOCUMENTATION PAGES VS. EFFORT</td>
<td>Result 3-14</td>
</tr>
<tr>
<td>Duration</td>
<td>Profile</td>
<td>AVERAGE PROJECT DURATION</td>
<td>Result 2-1</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>DURATION VS. LINES OF CODE</td>
<td>Result 3-3</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>DURATION VS. MODULES</td>
<td>Result 3-4</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>DURATION VS. EFFORT</td>
<td>Result 3-5</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>SIZE, EFFORT, AND SCHEDULE BY PHASE</td>
<td>Result 3-15</td>
</tr>
<tr>
<td>Effort</td>
<td>Profile</td>
<td>AVERAGE PROJECT EFFORT</td>
<td>Result 2-1</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>EFFORT DISTRIBUTION BY STAFF CATEGORY</td>
<td>Result 2-3</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>EFFORT DISTRIBUTION BY ACTIVITY</td>
<td>Result 2-4</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>ERROR DISTRIBUTION BY EFFORT TO CORRECT</td>
<td>Result 2-6</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>EFFORT VS. LINES OF CODE</td>
<td>Result 3-1</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>EFFORT VS. MODULES</td>
<td>Result 3-2</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>DURATION VS. EFFORT</td>
<td>Result 3-5</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>STAFF SIZE VS. EFFORT</td>
<td>Result 3-6</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>COMPUTER RUNS VS. EFFORT</td>
<td>Result 3-10</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>DOCUMENTATION PAGES VS. EFFORT</td>
<td>Result 3-14</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>SIZE, EFFORT, AND SCHEDULE BY PHASE</td>
<td>Result 3-15</td>
</tr>
<tr>
<td>FACTOR</td>
<td>TYPE OF RESULT</td>
<td>DESCRIPTION OF RESEARCH RESULT</td>
<td>RESULT NUMBER</td>
</tr>
<tr>
<td>--------------------</td>
<td>-------------------------</td>
<td>-------------------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Effort (Cont’d)</td>
<td>Relationship</td>
<td>EFFORT ADJUSTMENT FOR COMPLEXITY</td>
<td>Result 3-16</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>EFFORT ADJUSTMENT FOR TEAM EXPERIENCE</td>
<td>Result 3-17</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>EFFORT ADJUSTMENT FOR SCHEDULE TYPE</td>
<td>Result 3-18</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>EFFORT BY PHASE</td>
<td>Result 4-2</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>EFFORT ACTIVITY BY PHASE</td>
<td>Result 4-3</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>PROGRAMMER HOURS PER LINE OF CODE BY PHASE</td>
<td>Result 4-7</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>PROGRAMMER HOURS PER COMPUTER RUN BY PHASE</td>
<td>Result 4-11</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>PROGRAMMER HOURS PER SOFTWARE CHANGE BY PHASE</td>
<td>Result 4-14</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>UNCERTAINTY IN EFFORT AND SIZE ESTIMATES BY PHASE</td>
<td>Result 4-16</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>TRENDS IN EFFORT TO CHANGE BY PHASE</td>
<td>Result 4-17</td>
</tr>
<tr>
<td></td>
<td>Management Rule</td>
<td>VARIATIONS IN EFFORT TO CHANGE</td>
<td>Result 5-1</td>
</tr>
<tr>
<td></td>
<td>Management Rule</td>
<td>VARIATIONS IN SIZE AND EFFORT ESTIMATES</td>
<td>Result 5-3</td>
</tr>
<tr>
<td></td>
<td>Management Rule</td>
<td>DEVIATIONS IN LINES OF CODE PER STAFF HOUR</td>
<td>Result 5-8</td>
</tr>
<tr>
<td>Errors</td>
<td>Profile</td>
<td>ERROR RATE DURING IMPLEMENTATION</td>
<td>Result 2-2</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>ERROR RATE DURING SYSTEM TESTING</td>
<td>Result 2-2</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>ERROR RATE DURING ACCEPTANCE TESTING</td>
<td>Result 2-2</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>ERROR RATE DURING MAINTENANCE/OPERATIONS</td>
<td>Result 2-2</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>ERROR DISTRIBUTION BY ERROR CLASS</td>
<td>Result 2-5</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>ERROR DISTRIBUTION BY EFFORT TO CORRECT</td>
<td>Result 2-6</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>ERROR RATE IN EACH PHASE</td>
<td>Result 2-6</td>
</tr>
<tr>
<td>Experience</td>
<td>Profile</td>
<td>AVERAGE APPLICATION EXPERIENCE OF MANAGERS</td>
<td>Result 2-1</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>AVERAGE OVERALL EXPERIENCE OF MANAGERS</td>
<td>Result 2-1</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>AVERAGE APPLICATION EXPERIENCE OF TECHNICAL STAFF</td>
<td>Result 2-1</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>AVERAGE OVERALL EXPERIENCE OF TECHNICAL STAFF</td>
<td>Result 2-1</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>EFFORT ADJUSTMENT FOR TEAM EXPERIENCE</td>
<td>Result 3-17</td>
</tr>
<tr>
<td>Process</td>
<td>Profile</td>
<td>AVERAGE CODING RATE</td>
<td>Result 2-2</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>NOMINAL MAINTAINABILITY INDICATORS</td>
<td>Result 2-2</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>NOMINAL RELIABILITY INDICATORS</td>
<td>Result 2-2</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>PRODUCTIVITY VS. CODE REUSE</td>
<td>Result 3-7</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>PRODUCTIVITY VS. MODULE REUSE</td>
<td>Result 3-8</td>
</tr>
<tr>
<td>Product Size</td>
<td>Profile</td>
<td>AVERAGE LINES OF CODE DELIVERED</td>
<td>Result 2-1</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>EFFORT VS. LINES OF CODE</td>
<td>Result 3-1</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>EFFORT VS. MODULES</td>
<td>Result 3-2</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>DURATION VS. LINES OF CODE</td>
<td>Result 3-3</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>DURATION VS. MODULES</td>
<td>Result 3-4</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>COMPUTER RUNS VS. LINES OF CODE</td>
<td>Result 3-9</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>COMPUTER TIME VS. LINES OF CODE</td>
<td>Result 3-11</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>DOCUMENTATION PAGES VS. LINES OF CODE</td>
<td>Result 3-12</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>DOCUMENTATION PAGES VS. MODULES</td>
<td>Result 3-13</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>SIZE, EFFORT, AND SCHEDULE BY PHASE</td>
<td>Result 3-15</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>LINES OF CODE BY PHASE</td>
<td>Result 4-4</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>PROGRAMMER HOURS PER LINE OF CODE BY PHASE</td>
<td>Result 4-7</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>COMPUTER RUNS PER LINE OF CODE BY PHASE</td>
<td>Result 4-8</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>SOFTWARE CHANGES PER LINE OF CODE BY PHASE</td>
<td>Result 4-9</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>COMPUTER TIME PER LINE OF CODE BY PHASE</td>
<td>Result 4-10</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>UNCERTAINTY IN EFFORT AND SIZE ESTIMATES BY PHASE</td>
<td>Result 4-16</td>
</tr>
<tr>
<td>FACTOR</td>
<td>TYPE OF RESULT</td>
<td>DESCRIPTION OF RESEARCH RESULT</td>
<td>RESULT NUMBER</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------------</td>
<td>----------------------------------------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>Product Size</td>
<td>Management Rule</td>
<td>VARIATIONS IN SIZE AND EFFORT ESTIMATES</td>
<td>Result 5-3</td>
</tr>
<tr>
<td>(Cont’d)</td>
<td>Management Rule</td>
<td>DEVIATIONS IN COMPUTER USE PER LINE OF CODE</td>
<td>Result 5-5</td>
</tr>
<tr>
<td></td>
<td>Management Rule</td>
<td>DEVIATIONS IN LINES OF CODE</td>
<td>Result 5-6</td>
</tr>
<tr>
<td></td>
<td>Management Rule</td>
<td>DEVIATIONS IN CHANGES PER LINE OF CODE</td>
<td>Result 5-7</td>
</tr>
<tr>
<td></td>
<td>Management Rule</td>
<td>DEVIATIONS IN LINES OF CODE PER STAFF HOUR</td>
<td>Result 5-8</td>
</tr>
<tr>
<td>Reuse</td>
<td>Profile</td>
<td>AVERAGE REUSE PERCENTAGE</td>
<td>Result 2-2</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>COST OF REUSE</td>
<td>Result 2-9</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>PRODUCTIVITY VS. CODE REUSE</td>
<td>Result 3-7</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>PRODUCTIVITY VS. MODULE REUSE</td>
<td>Result 3-8</td>
</tr>
<tr>
<td>Schedule</td>
<td>Relationship</td>
<td>EFFORT ADJUSTMENT FOR SCHEDULE TYPE</td>
<td>Result 3-18</td>
</tr>
<tr>
<td></td>
<td>Model</td>
<td>SCHEDULE BY PHASE</td>
<td>Result 4-1</td>
</tr>
<tr>
<td>Staffing</td>
<td>Profile</td>
<td>AVERAGE STAFF SIZE (FTE)</td>
<td>Result 2-1</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>PEAK STAFF SIZE</td>
<td>Result 2-1</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>TOTAL STAFF SIZE (INDIVIDUALS)</td>
<td>Result 2-1</td>
</tr>
<tr>
<td></td>
<td>Profile</td>
<td>EFFORT DISTRIBUTION BY STAFF CATEGORY</td>
<td>Result 2-3</td>
</tr>
<tr>
<td></td>
<td>Relationship</td>
<td>STAFF SIZE VS. EFFORT</td>
<td>Result 3-6</td>
</tr>
<tr>
<td></td>
<td>Management Rule</td>
<td>DEVIATIONS IN STAFFING PLAN</td>
<td>Result 5-2</td>
</tr>
</tbody>
</table>
The term “environment profile” refers to a SEL research result containing typical values that characterize software development in the SEL environment. These profiles describe what constitutes a normal project under typical conditions. They include a wide range of factors relevant to this environment and serve as a basis for better understanding the SEL development process.

The projects studied by the SEL generally involve the development of scientific, ground-based, interactive graphics software with moderate reliability and response requirements. Representative applications provide spacecraft support in the areas of attitude determination, attitude control, maneuver planning, orbit adjustment, and mission analysis.

An environment profile captures values that characterize one or more relevant aspects of the development process in the SEL environment. These profiles serve as a basic piece of management data by providing a standard for managers to use in evaluating and planning new projects. Managers also use profiles as a baseline for assessing the effects of improvement initiatives.

The following paragraphs describe three groups of environment profiles identified through SEL research efforts.

- **Characteristic values** describe nominal SEL values related to product, process, performance, and quality factors. These profiles provide a standard for determining if a project is representative of the environment with respect to overall project-specific features such as effort, size, duration, or staffing levels. They additionally provide a means for evaluating a project’s performance and quality with regard to factors such as productivity, reliability, and maintainability.

- **Distribution profiles** describe the typical allocation of a key software development measure or resource classified according to its type. Examples of these profiles include distributions for effort by reported activity and for errors by effort to correct.

- **Cost profiles** describe heuristics that express typical values for costs that may reduce or increase normal expenditures from the basic development costs. Examples of these profiles include the cost of reuse and cost of user documentation.

The following pages present a selection of representative environment profiles published by the SEL.
**TYPE:** Characteristic Values

**DESCRIPTION:**

The profile contains typical values related to process and product factors that are characteristic of software development in the SEL environment.

**APPLICATION:**

Comparison — Permits overall project-specific factors to be compared with past projects to determine if a project is representative of the environment.

Assessment — Serves as a basis for judging how typical or atypical a project is with respect to factors used in diagnosing problems (e.g., team is more experienced than normal, project size is smaller than normal).

**SOURCE:**


**REPRESENTATION:**

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Low</th>
<th>Average</th>
<th>High</th>
</tr>
</thead>
<tbody>
<tr>
<td>Duration</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Time (months)</td>
<td>19</td>
<td>24</td>
<td>43</td>
</tr>
<tr>
<td>Cost</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Effort (staff-years)</td>
<td>3</td>
<td>14</td>
<td>32</td>
</tr>
<tr>
<td>Size</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Delivered code (KSLOC)</td>
<td>31</td>
<td>107</td>
<td>246</td>
</tr>
<tr>
<td>Staff</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Average staff (FTE)</td>
<td>4</td>
<td>8</td>
<td>15</td>
</tr>
<tr>
<td>Peak staff (FTE)</td>
<td>5</td>
<td>13</td>
<td>30</td>
</tr>
<tr>
<td>Individuals (total)</td>
<td>6</td>
<td>22</td>
<td>44</td>
</tr>
<tr>
<td>Application Experience</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Managers (years)</td>
<td>4</td>
<td>9</td>
<td>15</td>
</tr>
<tr>
<td>Technical staff (years)</td>
<td>2</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>Overall Experience</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Managers (years)</td>
<td>10</td>
<td>14</td>
<td>19</td>
</tr>
<tr>
<td>Technical staff (years)</td>
<td>4</td>
<td>6</td>
<td>9</td>
</tr>
</tbody>
</table>

**NOTES:**

Type of software: Scientific, ground-based, interactive graphic Machines: IBM 4341 and DEC VAX 780, 8600, and 8810
Sample: 10 FORTRAN (with 15% in Assembler) and 3 Ada projects
Staff-Year = 1664 effort hours
PERFORMANCE AND QUALITY CHARACTERISTICS

TYPE: Characteristic Values

DESCRIPTION:
The profile contains typical values related to performance and quality factors that are characteristic of software development in the SEL environment.

APPLICATION:
Planning – Provides guidance in anticipating values for performance and quality measures that can be expected on a project.
Comparison – Permits software quality and performance factors to be compared with previous projects to measure the relative effect of improvement initiatives.
Assessment – Serves as a basis for evaluating an ongoing project with respect to performance and quality factors that relate to productivity, reliability, and maintainability.
Assessment – Provides indicators to help detect problems in the quality of software development efforts.

SOURCE:
Manager's Handbook for Software Development (Revision 1), SEL-84-101, p. 3-3, Table 3-2, p. 6-8
Measures and Metrics for Software Development, SEL-83-002, pp. 4-12 through 4-22, Table 4-3


REPRESENTATION:

<table>
<thead>
<tr>
<th>Characteristics</th>
<th>Nominal Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Productivity</td>
<td></td>
</tr>
<tr>
<td>Coding rate</td>
<td>3.3 developed SLOC per hour</td>
</tr>
<tr>
<td>Reuse percentage</td>
<td>30% of code is &quot;reused&quot;</td>
</tr>
<tr>
<td>Reliability</td>
<td></td>
</tr>
<tr>
<td>Error rate</td>
<td>4 errors per developed KSLOC during implementation</td>
</tr>
<tr>
<td>Error rate</td>
<td>2 errors per developed KSLOC during system testing</td>
</tr>
<tr>
<td>Error rate</td>
<td>1 error per developed KSLOC during acceptance testing</td>
</tr>
<tr>
<td>Error rate</td>
<td>0.5 errors per developed KSLOC during maintenance/ operations</td>
</tr>
<tr>
<td>Change rate</td>
<td>14 changes to components per developed KSLOC during implementation</td>
</tr>
<tr>
<td>Maintainability</td>
<td></td>
</tr>
<tr>
<td>Effort to change</td>
<td>85% of all changes classified as &quot;easy&quot; or &quot;very easy&quot;</td>
</tr>
<tr>
<td>Effort to repair</td>
<td>85% of all repairs classified as &quot;easy&quot; or &quot;very easy&quot;</td>
</tr>
</tbody>
</table>

NOTES: Type of software: Scientific, ground-based, interactive graphic Systems
Machines: IBM 4341 and DEC VAX 780, 8600, and 8810
Language: FORTRAN (with 15% in Assembler)
EFFECT DISTRIBUTION BY STAFF CATEGORY

TYPE: Distribution Profile

DESCRIPTION:
The profile contains a distribution of total effort for the development process by staff category.

APPLICATION:
Planning — Provides a high-level cut at a staffing plan to identify typical management and support resources required.
Comparison — Permits overall staffing mix to be compared with past projects to determine if a project is representative of the environment.
Observation — When applied to an estimate of total effort, provides estimated completion values for tracking the expenditure of effort by staff category.

SOURCE:
An Approach to Software Cost Estimation, SEL-83-001, pp. 4-10, A.4

<table>
<thead>
<tr>
<th>Staffing Category</th>
<th>% of Total Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Programmers</td>
<td>84</td>
</tr>
<tr>
<td>Managers</td>
<td>10</td>
</tr>
<tr>
<td>Support Staff</td>
<td>6</td>
</tr>
</tbody>
</table>

NOTE: Support staff consists of administrative and publications support personnel.

Effort Distribution by Staff Category

Programmers (84.0%)
Managers (10.0%)
Support Staff (6.0%)
**Effort Distribution by Activity**

**TYPE:** Distribution Profile

**DESCRIPTION:**
The profile contains a distribution of the total hours of effort expended by reported activity for both FORTRAN and Ada projects.

**APPLICATION:**
- Planning — Provides a high-level understanding of where programmers can be expected to expend effort.
- Observation — When applied to an estimate of total effort, provides anticipated completion values for tracking the expenditure of effort by reported activity.

**SOURCE:**
Manager's Handbook for Software Development (Revision 1), SEL-84-101, p. 6-4

**Representation:**

<table>
<thead>
<tr>
<th>Reported Activity</th>
<th>FORTRAN %</th>
<th>Ada %</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design</td>
<td>23</td>
<td>19</td>
</tr>
<tr>
<td>Code</td>
<td>21</td>
<td>16</td>
</tr>
<tr>
<td>Test</td>
<td>30</td>
<td>35</td>
</tr>
<tr>
<td>Other</td>
<td>26</td>
<td>30</td>
</tr>
</tbody>
</table>

**NOTE:** Total hours expended are based on programmers providing weekly data attributing their time to the activity that they felt they were actually doing (no matter what development life-cycle phase they were in at the time).

![](image-url)
**ERROR DISTRIBUTION BY ERROR CLASS**

**TYPE:** Distribution Profile

**DESCRIPTION:**
The profile contains a distribution of errors classified by type of reported error for both FORTRAN and Ada projects.

**APPLICATION:**

*Planning* — Provides a high-level understanding of the proportional number of errors expected to be encountered by reported class.

*Observation* — When applied to an estimate of total errors, provides anticipated completion values for tracking the number of reported errors uncovered to date in each error class.

*Assessment* — Detects problems by observing deviations from the standard distribution in errors uncovered by reported class. (This usage assumes a distribution that is relatively constant over the project life-cycle.)

**SOURCE:**


---

<table>
<thead>
<tr>
<th>Error Class</th>
<th>% of Reported Errors</th>
</tr>
</thead>
<tbody>
<tr>
<td>FORTRAN</td>
<td>Ada</td>
</tr>
<tr>
<td>Initialization</td>
<td>15</td>
</tr>
<tr>
<td>Logic/Control</td>
<td>16</td>
</tr>
<tr>
<td>Interfaces</td>
<td>24</td>
</tr>
<tr>
<td>Data</td>
<td>30</td>
</tr>
<tr>
<td>Computational</td>
<td>15</td>
</tr>
</tbody>
</table>

---

**Error Distribution by Error Class**

**FORTRAN Projects**

- Computational (15.0%)
- Logic/Control (15.0%)
- Interfaces (24.0%)
- Data (30.0%)

**Ada Projects**

- Computational (15.0%)
- Logic/Control (22.0%)
- Interfaces (17.0%)
- Data (31.0%)
ERROR DISTRIBUTION BY EFFORT TO CORRECT

TYPE: Distribution Profile

DESCRIPTION:
The profile contains a distribution of errors classified by the effort required to correct the error.

APPLICATION:
Planning – Provides a high-level understanding of the amount of effort that can be expected to be required to correct errors.

Observation – When applied to an estimate of total errors, provides estimated completion values for tracking the number of reported errors by effort to correct.

Assessment – Provides an overall indicator for projecting the quality of the product being produced with respect to reliability, maintainability, and correctability.

Assessment – Detects problems by observing deviations from the standard distribution to judge relative difficulty encountered in correcting errors. (This usage assumes a distribution that is relatively constant over the project life-cycle.)

SOURCE:
"What Have We Learned in 6 Years?", F. E. McGary, Proceedings of the Seventh Annual Software Engineering Workshop, SEL-82-007, pp. 6, 24

<table>
<thead>
<tr>
<th>Effort to Correct</th>
<th>% of Total</th>
</tr>
</thead>
<tbody>
<tr>
<td>Very Easy</td>
<td>48</td>
</tr>
<tr>
<td>Easy</td>
<td>37</td>
</tr>
<tr>
<td>Medium</td>
<td>9</td>
</tr>
<tr>
<td>Hard</td>
<td>7</td>
</tr>
</tbody>
</table>

NOTE: The classifications defined for “effort to correct” reflect the actual effort expended, as reported by the programmer, when making a change. They do not reflect the subjective judgment of the programmer.

Error Distribution by Effort to Correct
TYPE: Distribution Profile

DESCRIPTION:
The profile contains a distribution of page counts for project documentation classified by the type of document.

APPLICATION:
Planning – Provides a high-level understanding of the proportional number of pages expected to be needed for project documentation based on the type of document.

SOURCE:
An Approach to Software Cost Estimation, SEL-63-001, p. 3-12

<table>
<thead>
<tr>
<th>Document Type</th>
<th>% of Total Effort</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design Description</td>
<td>33</td>
</tr>
<tr>
<td>Test Plans</td>
<td>7</td>
</tr>
<tr>
<td>User Documents</td>
<td>41</td>
</tr>
<tr>
<td>Component Prologs</td>
<td>16</td>
</tr>
<tr>
<td>Development/Management Plan</td>
<td>3</td>
</tr>
</tbody>
</table>

Page Distribution by Document Type
COST OF USER DOCUMENTATION

TYPE: Cost Profile

DESCRIPTION:
The profile describes additional documentation costs (over the basic development cost) based on the level of user documentation required.

NOTE: This result does not imply that documentation is unnecessary. The profile simply illustrates the relative cost associated with various documentation alternatives.

APPLICATION:
Planning — Provides guidance for estimating the cost associated with producing user documentation.

SOURCE:
An Approach to Software Cost Estimation, SEL-83-001, p. 3-12

REPRESENTATION:

<table>
<thead>
<tr>
<th>Document Level</th>
<th>Additional Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>No User Documents</td>
<td>0%</td>
</tr>
<tr>
<td>Informal User Docs</td>
<td>5%</td>
</tr>
<tr>
<td>Formal User Docs</td>
<td>16%</td>
</tr>
</tbody>
</table>

NOTE: Additional cost is expressed as a percentage of the basic development cost.
**TYPE:** Cost Profile

**DESCRIPTION:**

The profile describes the cost of reusing a software module compared to the cost of developing a new module. This cost varies as a function of the extent of the modifications required to reuse the module.

**APPLICATION:**

*Planning* – Provides guidance for estimating the cost reduction that can be expected when software modules are reused with little or no change. (The number of developed modules, typically used as a basis for estimating costs, may be expressed as \( N + 0.20 \times R \), with \( N \) being new or extensively modified modules and \( R \) being slightly modified or old modules.)

**SOURCE:**

Manager’s Handbook for Software Development (Revision1), SEL-84-101, p. 3-8, Table 3-8

An Approach to Software Cost Estimation, SEL-83-001, p. 3-17, Table 3-7

**REPRESENTATION:**

<table>
<thead>
<tr>
<th>Module Classification</th>
<th>Percent of Code Modified or Added</th>
<th>Relative Cost</th>
</tr>
</thead>
<tbody>
<tr>
<td>New</td>
<td>100</td>
<td>100</td>
</tr>
<tr>
<td>Extensively Modified</td>
<td>&gt; 25</td>
<td>100</td>
</tr>
<tr>
<td>Slightly Modified</td>
<td>1–25</td>
<td>20</td>
</tr>
<tr>
<td>Old</td>
<td>0</td>
<td>20</td>
</tr>
</tbody>
</table>

where

“Percent of Code” is the percentage of the module’s code that must be modified or added for reuse.

“Relative Cost” is expressed as a percentage of the cost of developing a new module.

**Cost of Reusing a Module**

![Cost of Reusing a Module Diagram]
SECTION 3—RELATIONSHIPS

The term “relationship” refers to a SEL research result that describes the correlation between various software development measures at a specific point in the project life cycle. These relationships provide a method for projecting the values of unknown development measures and costs from information that is more readily available or more accurately known.

Extensive SEL research has been conducted to identify key software development measures in this environment and to quantify the relationships that exist between these measures. Since accurate estimation and planning are essential in successfully managing the development process, many SEL studies focus on estimating project completion values such as total effort, product size, and expected duration. These studies generally apply statistical analysis to historical project data to obtain useful relationships.

Managers employ these results to facilitate and standardize the planning and estimation process. Without a set of relationships derived from an understanding of the environment and from historical data, planning and estimation become largely guesswork. As a result, relationships between key development measures or costs are considered basic and necessary management data.

The following paragraphs describe three distinct groups of relationships identified through SEL research efforts.

- **General relationships** are time-independent equations that describe the correlation between key software development measures at project completion. These relationships provide a method for projecting a desired value 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 (LOC) may be used to estimate the effort required for a project of a given size.

  When planning a project, experienced managers (or estimators) follow an established procedure that employs these relationships. Generally, this process involves (1) estimating the size of the software product, (2) converting the size estimate to an estimate of total effort, and (3) determining an expected duration and practical staffing level for completing the project.

  Since a set of relationships should be inherently consistent for the environment, a manager's use of relationships in this process implicitly detects and helps correct potential planning problems. For example, a relationship that expresses duration as a function of LOC may identify the impracticality of targeting the software delivery for a specified date without a reduction in scope or size.
• **Phase-dependent relationships** are time-independent equations, typically captured as a set of equations in a table, that apply to specific life-cycle phases during the project. They provide a method of reestimating the completion values of key software development measures for ongoing projects based on the most accurate and latest information available.

As prescribed in the SEL environment, managers reestimate the values for key development measures at the end of each life-cycle phase. The basic values to be reestimated consist of effort, schedule duration, and project size. SEL research has identified an optimum set of relationships for each phase to accomplish this reestimation.

These relationships comprise a system intended for use with specific data that are available when the relevant phase completes. For example, the most accurate measure of project size available through the requirements analysis phase may be the number of subsystems. After detailed design, when the system has been decomposed to a finer level, the most accurate measure of project size may be the number of modules. The discrete sets of relationships reflect the increasing granularity and decreasing uncertainty that occurs as the life-cycle progresses.

• **Adjustment factors** describe relationships expressing guidelines for determining multiplicative factors that may be applied to refine estimates. These factors account for differences in the problem, process, or environment that vary significantly from typical development conditions and that will increase or decrease nominal project expenditures.

These relationships are represented in a tabular format that can be used to calculate the appropriate adjustment factor when atypical conditions arise. Estimates from other relationships, derived for nominal projects, are multiplied by the computed factor to revise the estimate upward or downward as needed.

Examples of these relationships include guidelines for adjusting effort estimates to reflect problem complexity or development team experience.

The following pages present a selection of representative relationships published by the SEL.
TYPE: General Relationship

DESCRIPTION:
The relationship describes the correlation between total effort and source lines of code.

NOTE: This result may be counterintuitive since many popular equations used outside of the SEL environment have an exponent greater than 1.

APPLICATION:
Planning — Estimating the total effort required for a project based on a measure of the project size.

Planning — Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.

Observation — Providing a target completion value for scaling effort models used in monitoring the expenditure of effort over the development life-cycle.

SOURCE:
The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 13

RELATED RESULTS:
1,2-Finding Relationships Between Effort and Other Variables in the SEL* V. R. Basil and N. M. Panillio-Yap, Collected Software Engineering Papers: Volume III, SEL-65-003, pp. 4-4, 4-5

1 Describes alternative relationships, E = 4.372 + 1.430 * devlines and E = 5.497 + 1.500 * newlines, that relate total effort to source lines of code in developed KSLOC and new KSLOC, respectively.


REPRESENTATION:

\[ E = 1.266 \times L^{0.986} \]

where

- \( E \) is total effort in staff-months (with 172 hours per staff-month)
- \( L \) is total delivered lines of code in KSLOC

NOTE: uncertainty = 0.456

\[ E_{\text{upper bound}} = E \times (1.0 + \text{uncertainty}) \]

\[ E_{\text{lower bound}} = E / (1.0 + \text{uncertainty}) \]

EFFORT vs. LINES OF CODE

![Graph showing the relationship between effort and lines of code]

- [Graph X-axis: Delivered Lines of Code (KSLOC) Y-axis: Effort (Staff-month)]

- [Graph showing different lines representing upper and lower bounds]

- [Graph legend indicating the equation and bounds]

- [Graph data points and trend lines]

- [Graph annotations and axis labels]
**TYPE:** General Relationship

**DESCRIPTION:**
The relationship describes the correlation between total effort and module count.

**APPLICATION:**
- **Planning** – Estimating the total effort required for a project based on a measure of the project size.
- **Planning** – Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.
- **Observation** – Providing a target completion value for scaling effort models used in monitoring the expenditure of effort over the development life-cycle.

**SOURCE:**
*The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 15*

**RELATED RESULTS:**
1. *The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 19*

**REPRESENTATION:**

\[ E = 0.029 \times M^{1.319} \]

where

- \( E \) is total effort in staff-months (with 172 hours per staff-month)
- \( M \) is total number of modules in project (defined as any separately compilable entity)

**NOTE:**
- uncertainty = 0.431
- \( E_{\text{upper bound}} = E \times (1.0 + \text{uncertainty}) \)
- \( E_{\text{lower bound}} = E / (1.0 + \text{uncertainty}) \)

---

1 Describes an alternative relationship, \( E = 0.052 \times DM^{1.277} \), that relates total effort to the number of developed modules.
TYPE: General Relationship

DESCRIPTION:

The relationship describes the correlation between project duration and source lines of code.

APPLICATION:

Planning – Estimating the total time required to complete a project based on a measure of the project size.

Planning – Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.

Observation – Providing a time scale for schedule models used to monitor software development measures such as effort and lines of code.

SOURCE:

The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 32

RELATED RESULTS:

1 The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 36

1 Describes an alternative relationship, \( D = 4.836 \times L^{0.257} \), that relates duration to source lines of code in developed KSLOC.

REPRESENTATION:

\[
D = 5.450 \times L^{0.203}
\]

where

- \( D \) is project duration in months (from start of project to end of acceptance testing)
- \( L \) is total delivered lines of code in KSLOC

NOTE: uncertainty = 0.367

\[ D_{\text{upper bound}} = D \times (1.0 + \text{uncertainty}) \]

\[ D_{\text{lower bound}} = D / (1.0 + \text{uncertainty}) \]
TYPE: General Relationship

DESCRIPTION:
The relationship describes the correlation between project duration and module count.

APPLICATION:
Planning – Estimating the total time required to complete a project based on a measure of the project size.
Planning – Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.
Observation – Providing a time scale for schedule models used to monitor software development measures such as effort and lines of code.

SOURCE:
The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 32

RELATED RESULTS:
1The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 36

REPRESENTATION:

\[ D = 2.453 \times M^{0.275} \]

where

- \( D \) is project duration in months (from start of project to end of acceptance testing)
- \( M \) is total number of modules in project (defined as any separately compilable entity)

NOTE: uncertainty = 0.361

\[
\begin{align*}
D_{\text{upper bound}} &= D \times (1.0 + \text{uncertainty}) \\
D_{\text{lower bound}} &= D / (1.0 + \text{uncertainty})
\end{align*}
\]

1 Describes an alternative relationship, \( D = 1.835 \times D^{0.348} \), that relates duration to the number of developed modules.
TYPE: General Relationship

DESCRIPTION:
The relationship describes the correlation between project duration and total effort.

APPLICATION:
Planning — Estimating the total time required to complete a project based on a measure of total effort.
Planning — Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.
Observation — Providing a time scale for schedule models used to monitor software development measures such as effort and lines of code.

SOURCE:
The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 41

REPRESENTATION:

\[ D = 5.104 \cdot E^{0.210} \]

where

- \( D \) is project duration in months (from start of project to end of acceptance testing)
- \( E \) is total effort in staff-months (with 172 hours per staff-month)

NOTE: uncertainty = 0.346

\[ D_{\text{upper bound}} = D \cdot (1.0 + \text{uncertainty}) \]
\[ D_{\text{lower bound}} = D / (1.0 + \text{uncertainty}) \]
TYPE: General Relationship

DESCRIPTION:
The relationship describes the correlation between average staff size and total effort.

APPLICATION:
Planning – Estimating the average staff size required for a project based on a measure of total effort.
Planning – Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.

SOURCE:
The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 41

REPRESENTATION:

\[ S = 0.198 \times E^{0.788} \]

where

\[ S \] is average staff size in staff-months per month (defined as total effort divided by duration or E/D)

\[ E \] is total effort in staff-months (with 172 hours per staff-month)

NOTE: uncertainty = 0.347
\[ S_{\text{upper bound}} = S \times (1.0 + \text{uncertainty}) \]
\[ S_{\text{lower bound}} = S / (1.0 + \text{uncertainty}) \]
TYPE: General Relationship

DESCRIPTION:
The relationship describes the correlation between productivity and code reuse.

APPLICATION:
Planning — Estimating the productivity required for a project based on an expected value for code reuse.

Planning — Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.

SOURCE:
The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 21

REPRESENTATION:

\[ P = 678.311 \times (DL/L)^{-0.730} \]

where:

- \( P \) is productivity in delivered SLOC per staff-month (defined as total delivered SLOC divided by total effort, or L/E, with 172 hours per staff-month)
- \( DL \) is total developed lines of code in KSLOC (excludes reused code)
- \( L \) is total delivered lines of code in KSLOC

NOTE:
- uncertainty = 0.235
- \( P_{\text{upper bound}} = P \times (1.0 + \text{uncertainty}) \)
- \( P_{\text{lower bound}} = P / (1.0 + \text{uncertainty}) \)
TYPE: General Relationship

DESCRIPTION:
The relationship describes the correlation between productivity and module reuse.

APPLICATION:
Planning — Estimating the productivity required for a project based on an expected value for module reuse.

Planning — Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.

SOURCE:
The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 21

\[ P = 677.760 \times (D/M)^{-0.708} \]

where

- \( P \) is productivity in delivered SLOC per staff-month (defined as total delivered SLOC divided by total labor effort, or L/E, with 172 hours per staff-month)
- \( D \) is total number of developed modules in project (excludes reused modules)
- \( M \) is total number of modules in project

NOTE: uncertainty = 0.258

\[ P_{\text{upper bound}} = P \times (1.0 + \text{uncertainty}) \]
\[ P_{\text{lower bound}} = P / (1.0 + \text{uncertainty}) \]
**TYPE:** General Relationship

**DESCRIPTION:**
The relationship describes the correlation between number of computer runs and source lines of code.

**APPLICATION:**
*Planning*—Estimating the computer runs required for a project based on a measure of total effort.

*Planning*—Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.

*Observation*—Providing a target completion value for scaling measure models used in monitoring the expenditure of computer resources over the development life-cycle.

**SOURCE:**
*Manager's Handbook for Software Development (Revision 1)*, SEL-84-101, p. 3-5

**RELATED RESULTS:**
1,2-Finding Relationships Between Effort and Other Variables in the SEL, *V. R. Basili and N. M. Panllio-Yap, Collected Software Engineering Papers: Volume III*, SEL-85-003, p. 4-6

---

1 Describes an alternative relationship, numruns = -108.274 + 150.879 * devlines, that relates the number of computer runs to source lines of code in developed KSLOC.

TYPE: General Relationship

DESCRIPTION:
The relationship describes the correlation between number of computer runs and total effort.

APPLICATION:
Planning - Estimating the computer runs required for a project based on a measure of total effort.
Planning - Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.
Observation - Providing a target completion value for scaling measure models used in monitoring the expenditure of computer resources over the development life-cycle.

SOURCE:


REPRESENTATION:

\[ E = 9.951 + 0.008 \times \text{numruns} \]

where

- \( E \) is total effort in staff-months (with 160 hours per staff-month)
- \( \text{numruns} \) is total number of computer runs (includes every batch job submitted and every interactive terminal sign-on)
TYPE: General Relationship

DESCRIPTION:
The relationship describes the correlation between computer CPU time and source lines of code.

APPLICATION:
Planning – Estimating the computer CPU time required for a project based on a measure of the project size.

Planning – Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.

Observation – Providing a target completion value for scaling measure models used in monitoring the expenditure of computer resources over the development life-cycle.

SOURCE:
Manager's Handbook for Software Development (Revision 1), SEL-84-101, p. 3-5

REPRESENTATION:

\[ H = 0.0008 \times L \]

where

- \( H \) is total hours of computer CPU time
- \( L \) is total delivered lines of code in SLOC

COMPUTER TIME vs. LINES OF CODE
### Documentation PAGES vs. Lines of Code

**Type:** General Relationship

**DESCRIPTION:** The relationship describes the correlation between pages of documentation and source lines of code.

<table>
<thead>
<tr>
<th>P</th>
<th>= 120 + 0.028 * L</th>
</tr>
</thead>
</table>

where

- **P** is the total pages of documentation (includes the requirements analysis report, design documents, system description, and user's guide)
- **L** is the total delivered lines of code in SLOC

**APPLICATION:**
- **Planning:** Estimating pages of documentation for a project based on a measure of the project size.
- **Planning:** Validating values in a set of project estimates for consistency and reasonable-ness when used in conjunction with other relationships.

**SOURCE:**
- Manager's Handbook for Software Development (Revision 1, SEL-84-101, P. 3-7)
- An Approach to Software Cost Estimation, SEL-83-001, P. 3-11

**Related Results:**

1. Describes an alternative relationship, P = 0.04 * L, that relates pages of documentation to source lines of code in developed SLOC.

---

**Diagram:**

- **Title:** DOCUMENTATION VS. LINES OF CODE
- **Graph:**
  - x-axis: Number of Pages
  - y-axis: Delivered Lines of Code (SLOC)
  - Graph shows a linear trend with points plotted for various numbers of pages and lines of code.

---

3-14

6199
TYPE: General Relationship

DESCRIPTION:
The relationship describes the correlation between pages of documentation and module count.

APPLICATION:
Planning – Estimating pages of documentation for a project based on a measure of the project size.
Planning – Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.

SOURCE:
The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 27

RELATED RESULTS:
1 The Software Engineering Laboratory: Relationship Equations, SEL-79-002, p. 27

REPRESENTATION:

\[ \text{DOC} = 33.585 \times M^{0.662} \]

where

\[ \text{DOC} \] is total pages of documentation (includes program design, test plans, user's guides, system description, and module description)

\[ M \] is total number of modules in project (defined as any separately compilable entity)

NOTE: uncertainty = 0.227
\[ \text{DOC}_{\text{upper bound}} = \text{DOC} \times (1.0 + \text{uncertainty}) \]
\[ \text{DOC}_{\text{lower bound}} = \frac{\text{DOC}}{1.0 + \text{uncertainty}} \]

1 Describes an alternative relationship, \[ \text{DOC} = 359.300 \times \text{DM}^{0.279} \], that relates pages of documentation to the total number of developed modules.
TYPE: General Relationship

DESCRIPTION:
The relationship describes the correlation between pages of documentation and total effort.

APPLICATION:
Planning – Estimating pages of documentation for a project based on a measure of total effort.

Planning – Validating values in a set of project estimates for consistency and reasonableness when used in conjunction with other relationships.

SOURCE:

\[ E = 0.581 + 0.045 \times \text{docpages} \]

where

- \( E \) is total effort in staff-months (with 160 hours per staff-month)
- \( \text{docpages} \) is total number of pages of documentation (includes program design document, development plan, test plan, system description, module descriptions, and user's guide)

TYPE: Phase-Dependent Relationships

DESCRIPTION:

The relationships describe a set of equations captured in a table that serve as a system for reestimating completion values for key software development measures at the end of each life-cycle phase.

APPLICATION:

Planning – Reestimating product size, total effort, and project duration based on specific data at recommended points in the life-cycle of ongoing projects.

SOURCE:

Manager’s Handbook for Software Development (Revision 1), SEL-84-101 p. 3-3, Table 3-2

<table>
<thead>
<tr>
<th>End of Phase</th>
<th>Relationship</th>
<th>Uncertainty</th>
</tr>
</thead>
</table>
| Requirements Analysis| L = 11000 * SS  
                         E = 3000 * SS  
                         D = 83 * SS     | 0.75          |
| Preliminary Design   | L = 190 * M   
                         E = 52 * M      
                         D = 1.45 * M    | 0.40          |
| Detailed Design      | DL = 200 * DM  
                         E = 0.31 * DL   
                         D = 0.0087 * DL | 0.25          |
| Implementation       | L = 1.26 * LCur  
                         E = 1.43 * ECur  
                         D = 1.54 * DCur  | 0.10          |
| System Testing       | E = 1.11 * ECur  
                         D = 1.18 * DCur  | 0.05          |

where

L is total size in SLOC
E is total effort in staff-hours
D is total duration in weeks per staff member (based on a full-time employee’s average work week with 1664 hours annually)
SS is total number of subsystems in project
M is total number of modules in project
DM is total number of developed modules in project (defined as N + 0.2*R, with N being new modules and R being reused modules)
DL is developed lines of code in SLOC
LCur is size in SLOC through the current phase
ECur is effort expended in staff-hours through the current phase
DCur is schedule expended in calendar weeks through the current phase

NOTE: Uncertainty applies to effort or size estimates as follows:
Estimate_{upper bound} = Estimate * (1.0 + uncertainty)
Estimate_{lower bound} = Estimate / (1.0 + uncertainty)
**EFFORT ADJUSTMENT FOR COMPLEXITY**

**TYPE:** Adjustment Factor

**DESCRIPTION:**
The relationship provides a multiplicative factor for adjusting effort estimates based on complexity guidelines.

**APPLICATION:**
Planning—Permits an estimate of the total effort required for a project to be adjusted to account for complexity as determined by indicators of past experience with the problem and environment.

Planning—Classifies a project with respect to complexity based on the development team's experience with the application and the computing environment.

**SOURCE:**
Manager's Handbook for Software Development (Revision 1), SEL-84-101, p. 3-4, Table 3-3

An Approach to Software Cost Estimation, SEL-83-001, p. 4-5, Table 4-2

**REPRESENTATION:**

<table>
<thead>
<tr>
<th>Project</th>
<th>Environment</th>
<th>Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>OLD</td>
<td>OLD</td>
<td>1.0</td>
</tr>
<tr>
<td>OLD</td>
<td>NEW</td>
<td>1.4</td>
</tr>
<tr>
<td>NEW</td>
<td>OLD</td>
<td>1.4</td>
</tr>
<tr>
<td>NEW</td>
<td>NEW</td>
<td>2.3</td>
</tr>
</tbody>
</table>

where

- **Project Type** is considered OLD when the development team has more than 2 years experience with the application area (e.g., orbit determination, simulator)
- **Environment Type** is considered OLD when the development team has more than 2 years experience with the computing environment (e.g., IBM 4341, VAX 8610)
**TYPE:** Adjustment Factor

**DESCRIPTION:**

The relationship provides a multiplicative factor for adjusting effort estimates based on years of team experience.

**APPLICATION:**

Planning – Permits an estimate of the total effort required for a project to be adjusted to account for years of team experience with the application.

**SOURCE:**

Manager’s Handbook for Software Development (Revision 1), SEL-84-101, p. 3-4, Table 3-4

An Approach to Software Cost Estimation, SEL-83-001, p. 4-5, Table 4-3

<table>
<thead>
<tr>
<th>Team Years of Application Experience</th>
<th>Effort Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>0.5</td>
</tr>
<tr>
<td>8</td>
<td>0.6</td>
</tr>
<tr>
<td>6</td>
<td>0.8</td>
</tr>
<tr>
<td>4</td>
<td>1.0</td>
</tr>
<tr>
<td>2</td>
<td>1.4</td>
</tr>
<tr>
<td>1</td>
<td>2.6</td>
</tr>
</tbody>
</table>

where

Team Years is the average of all team member’s years of application experience weighted by the member’s participation on the team with

Application experience is defined as prior work on similar applications (e.g., attitude and orbit determination).

Member’s participation is defined as time spent working on the project as a proportion of total project effort.
TYPE: Adjustment Factor

DESCRIPTION:
The relationship provides a multiplicative factor for adjusting effort estimates based on schedule type.

APPLICATION:
Planning - Permits an estimate of the total effort required for a project to be adjusted to account for a faster or slower schedule than normally used in the environment.

Planning - Characterizes the type of schedule for a project based on an expected work rate and complexity classification.

SOURCE:
An Approach to Software Cost Estimation, SEL-83-001, pp. 4-4, 4-7, Tables 4-4 and 4-5

RESULT:

<table>
<thead>
<tr>
<th>Schedule Characterization</th>
<th>Effort Multiplier</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fast</td>
<td>1.15</td>
</tr>
<tr>
<td>Average</td>
<td>1.00</td>
</tr>
<tr>
<td>Slow</td>
<td>0.85</td>
</tr>
</tbody>
</table>

where Schedule Characterization is determined from the following table:

<table>
<thead>
<tr>
<th>Complexity Class</th>
<th>Derived Schedule Type for Work Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Fast</td>
</tr>
<tr>
<td>OLD/OLD</td>
<td>&gt; 0.24</td>
</tr>
<tr>
<td>OLD/NEW</td>
<td>&gt; 0.17</td>
</tr>
<tr>
<td>NEW/OLD</td>
<td>&gt; 0.17</td>
</tr>
<tr>
<td>NEW/NEW</td>
<td>&gt; 0.11</td>
</tr>
</tbody>
</table>

with

Work Rate calculated in developed KSLOC of executable code per calendar week

Complexity Class expressed as a project/environment type pair (e.g., OLD/NEW)

(Project Type is OLD when the development team has more than 2 years experience with the application area; Environment Type is OLD when the development team has more than 2 years experience with the computing environment.)
SECTION 4—MODELS

The term “model” refers to a SEL research result that describes the expected behavior of a software development measure as a function of time. For example, a research result containing a tabular listing of the fraction of errors detected during each development phase can be considered a “model of error detection.” This type of research result has been described and applied in many SEL studies.

SEL research studies employ models to represent a “typical” project. The study results are commonly based on analyzing the comparison of some project of interest to the model. In almost all cases, the models have been obtained by averaging the behavior of the measure over several projects.

A model is a basic piece of management data. Models provide the standard or guidelines that managers use to judge the status of a project. By comparing the evolution of a measure to its expected behavior, the manager can assess a project’s health and predict the measure’s future behavior. Models of resource measures, such as effort or computer use, also can be used for planning.

The following paragraphs describe five groups of models.

- A **schedule model** describes how much time is allocated to each phase of the software development life cycle. In the SEL environment, most research results are obtained from projects in the requirements analysis through acceptance testing life-cycle phases.

  A schedule model is combined with each of the other types of models to provide a time scale for depicting the “typical” project in the environment.

- A **basic measure model** describes the behavior over time of a fundamental software development measure such as LOC, effort, or software errors.

  A manager uses basic measure models for tracking those software development measures that are directly related to the magnitude of the specific project being monitored. An estimate of the completion value of the measure for the project (see Section 3) is required since each model expresses a measure’s behavior on a normalized scale.

- An **environment model** describes the behavior over time of a ratio of two basic software development measures such as computer resource usage per job, LOC per staff hour, or errors per LOC.

  Environment models are useful when a manager tracks a project’s behavior as it compares to other projects in the environment. The effects of project size are minimized through the use of ratios. These models are characteristic of the environment and not a single project; therefore, they are not normalized but are expressed in absolute units.
• An uncertainty model describes the changing confidence interval for resource estimates over time. As a software development project evolves through specification, design, and implementation, more information becomes available about the eventual size of the completed project. As the project evolves, the manager's estimates of the resources required to complete it become more certain. The model describes the manner in which the uncertainty decreases.

• A subjective model is a description of specific features of a software development measure's probable behavior over time. The development measures are either basic measures or ratios of basic measures. SEL research describes these models with text and in some cases they are illustrated by a sketch. Subjective models are based on the experiences of managers in a particular software development environment. Subjective models are not quantitative; they describe the characteristics of a measure's behavior using words such as "constant," "increasing," or "starts to decrease."

Subjective models describe significant features of the behavior of measures that signal to the manager when a development process is working correctly or incorrectly. In SEL literature, subjective models are often paired with rules that indicate problems. For example, a subjective model that describes a measure's behavior as "constant in magnitude" is another view of a rule that states that a rapidly changing measure is an indicator of a problem.

The following pages describe a selection of models published by the SEL.
TYPE: Schedule Model

DESCRIPTION:
The model identifies the amount of time normally allocated to each life-cycle phase as a percentage of total project duration.

APPLICATION:
Planning — Provides guidance in selecting dates for major reviews and milestones.

Comparison — When used with an estimate of project duration, provides a time scale for depicting typical projects in the environment.

Prediction — Extrapolating the schedule of an ongoing project into the future given the time required to complete past life-cycle phases.

SOURCE:
Manager's Handbook for Software Development (Revision 1), SEL-84-101, p. 3-1

<table>
<thead>
<tr>
<th>Life-Cycle Phase</th>
<th>Percent of Time in Phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Analysis</td>
<td>12</td>
</tr>
<tr>
<td>Preliminary Design</td>
<td>8</td>
</tr>
<tr>
<td>Detailed Design</td>
<td>15</td>
</tr>
<tr>
<td>Implementation</td>
<td>30</td>
</tr>
<tr>
<td>System Testing</td>
<td>20</td>
</tr>
<tr>
<td>Acceptance Testing</td>
<td>15</td>
</tr>
</tbody>
</table>

SCHEDULE BY PHASE

![Bar chart showing the percentage of time in each phase: Requirements Analysis 12%, Preliminary Design 8%, Detailed Design 15%, Implementation 30%, System Testing 20%, Acceptance Testing 15%.]
TYPE: Basic Measure Model

DESCRIPTION:
The model identifies the effort normally allocated in each life-cycle phase as a percentage of total effort.

APPLICATION:
Planning – Provides guidance in allocating effort to phases of the development life-cycle.

Prediction – Extrapolating the effort required in future phases based on the known effort expended in past phases.

Analysis – Estimating the fraction of the current phase that is elapsed based on the fraction of the total estimated effort that has been expended to date.

Assessment – Detecting problems by observing deviations of actual staff-hours from guidelines based on the model.

SOURCE:
Manager’s Handbook for Software Development (Revision 1), SEL-84-101, p. 3-1
EFFECT ACTIVITY BY PHASE

DESCRIPTION:
The model illustrates how effort expended on specific activities is distributed throughout the development life-cycle.

APPLICATION:
Planning – Providing guidance in allocating effort to phases of the development life-cycle.

Comparison – Providing a standard breakdown of effort by activity based on previous projects for comparison with current projects.

Assessment – Detecting problems by observing deviations from the standard activity breakdown.

SOURCE:
1"The Effectiveness of Software Prototyping: A Case Study," M. V. Zeikowitz, Collected Software Engineering Papers: Volume VI, SEL-88-002, pp. 2-6 through 2-8

---

TYPE: Basic Measure Model

DESCRIPTION:

The model identifies the growth pattern of source code produced by a development project.

APPLICATION:

Prediction - Extrapolating the code to be produced in future phases given the known size of code produced in past phases.

Analysis - Estimating the fraction of the current phase that is elapsed based on the fraction of the total estimated code size that has been produced to date.

Assessment - Detecting problems by observing deviations of actual code size from guidelines based on the model.

SOURCE:


<table>
<thead>
<tr>
<th>Phase</th>
<th>% of Phase</th>
<th>% of Total Lines</th>
</tr>
</thead>
<tbody>
<tr>
<td>Design (Start)</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>0.00</td>
</tr>
<tr>
<td>Code/Unit Test (Start)</td>
<td>0</td>
<td>0.00</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>6.86</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>36.05</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>53.99</td>
</tr>
<tr>
<td>System Test (Start)</td>
<td>0</td>
<td>76.28</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>86.82</td>
</tr>
<tr>
<td>Acceptance Test (Start)</td>
<td>0</td>
<td>94.88</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>95.09</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>98.14</td>
</tr>
<tr>
<td></td>
<td>75</td>
<td>99.58</td>
</tr>
<tr>
<td>End</td>
<td>100</td>
<td>100.00</td>
</tr>
</tbody>
</table>
**TYPE:** Basic Measure Model

**DESCRIPTION:**

The model illustrates how the demand for computer time varies over the development life-cycle for FORTRAN and Ada projects. The amount of computer utilization (CPU time) is expressed as a percentage of the average weekly use for the entire project.

**NOTE:** The maximum value attained should not exceed three times the weekly average.

**APPLICATION:**

*Planning*—Provides guidance in anticipating computer resources required at various points in the development life-cycle.

*Prediction*—Extrapolating the total computer time expected to be used from the known amount of computer use to date.

*Analysis*—Estimating the fraction of the current phase that is elapsed based on the fraction of the total estimated computer time that has been used to date.

*Assessment*—Detecting problems by observing deviations of actual computer time used from guidelines based on the model.

**SOURCE:**

Manager's Handbook for Software Development (Revision 1), SEL-84-101, p. 3-6
ERROR RATE IN EACH PHASE

TYPE: Environment Model

DESCRIPTION:
The model describes the behavior of the error rate that can be expected in each life-cycle phase. The error rate is expressed as a ratio of the number of errors detected in the phase to the total estimated size of the project in developed lines of source code.

APPLICATION:
Assessment — Provides a characteristic pattern of behavior for the error rate that can be compared to actual measurements to help judge the relative reliability of the software being produced.

Assessment — Detecting problems by observing deviations of the error rate from values given by the model. (Failure of the error rate to decline during testing may suggest that many undiscovered errors remain in the software.)

SOURCE:

Manager's Handbook for Software Development (Revision 1), SEL-84-101, p. 6-8

<table>
<thead>
<tr>
<th>Phase</th>
<th>Errors Detected Per Developed KSLOC</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code/Unit Test</td>
<td>4</td>
</tr>
<tr>
<td>System Test</td>
<td>2</td>
</tr>
<tr>
<td>Acceptance Test</td>
<td>1</td>
</tr>
<tr>
<td>Maintenance/Operations</td>
<td>0.5</td>
</tr>
</tbody>
</table>

NOTE: This implies that a 50% reduction in the error detection rate for a given phase is expected for each subsequent phase.
PROGRAMMER HOURS PER LINE OF CODE BY PHASE

TYPE: Environment Model

DESCRIPTION:
The model describes the behavior of the cumulative ratio of programmer hours per line of source code.

APPLICATION:
Prediction – Extrapolating the final value of the ratio based on the current observed value and the values given by the model.
Assessment – Detecting problems by observing deviations of the ratio from values given by the model.

SOURCE:
Monitoring Software Development Through Dynamic Variables (Revision 1), SEL-83-106, p. 47

<table>
<thead>
<tr>
<th>Phase</th>
<th>Hours (in tenths) per Line of Code</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start coding</td>
<td>4.340</td>
<td>4.037</td>
</tr>
<tr>
<td>20% coding</td>
<td>2.379</td>
<td>1.673</td>
</tr>
<tr>
<td>40% coding</td>
<td>1.718</td>
<td>1.155</td>
</tr>
<tr>
<td>50% coding</td>
<td>1.683</td>
<td>1.026</td>
</tr>
<tr>
<td>60% coding</td>
<td>1.426</td>
<td>0.664</td>
</tr>
<tr>
<td>80% coding</td>
<td>1.324</td>
<td>0.539</td>
</tr>
<tr>
<td>Start system testing</td>
<td>1.260</td>
<td>0.443</td>
</tr>
<tr>
<td>50% system testing</td>
<td>1.312</td>
<td>0.447</td>
</tr>
<tr>
<td>Start acceptance testing</td>
<td>1.456</td>
<td>0.459</td>
</tr>
<tr>
<td>End acceptance testing</td>
<td>1.531</td>
<td>0.505</td>
</tr>
</tbody>
</table>

Programmer Hours per Line of Code

![Graph showing programmer hours per line of code by phase]
COMPUTER RUNS PER LINE OF CODE BY PHASE

TYPE: Environment Model

DESCRIPTION:
The model describes the behavior of the cumulative ratio of computer runs per line of source code.

APPLICATION:
Prediction – Extrapolating the final value of the ratio based on the current observed value and the values given by the model.
Assessment – Detecting problems by observing deviations of the ratio from values given by the model.

SOURCE:
Monitoring Software Development Through Dynamic Variables (Revision 1), SEL-83-106, p. 48

<table>
<thead>
<tr>
<th>Phase</th>
<th>Computer Runs per Line of Code</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start coding</td>
<td>0.0401</td>
<td>0.0454</td>
</tr>
<tr>
<td>20% coding</td>
<td>0.0503</td>
<td>0.0402</td>
</tr>
<tr>
<td>40% coding</td>
<td>0.0575</td>
<td>0.0362</td>
</tr>
<tr>
<td>50% coding</td>
<td>0.0659</td>
<td>0.0392</td>
</tr>
<tr>
<td>60% coding</td>
<td>0.0622</td>
<td>0.0281</td>
</tr>
<tr>
<td>80% coding</td>
<td>0.0716</td>
<td>0.0298</td>
</tr>
<tr>
<td>Start system testing</td>
<td>0.0795</td>
<td>0.0350</td>
</tr>
<tr>
<td>50% system testing</td>
<td>0.0899</td>
<td>0.0393</td>
</tr>
<tr>
<td>Start acceptance testing</td>
<td>0.0972</td>
<td>0.0397</td>
</tr>
<tr>
<td>End acceptance testing</td>
<td>0.1050</td>
<td>0.0444</td>
</tr>
</tbody>
</table>

Computer Runs per Line of Code

[Graph showing computer runs per line of code across different phases]
SOFTWARE CHANGES PER LINE OF CODE BY PHASE

TYPE: Environment Model

DESCRIPTION:
The model describes the behavior of the cumulative ratio of software changes per line of source code.

APPLICATION:

Prediction – Extrapolating the final value of the ratio based on the current observed value and the values given by the model.

Assessment – Detecting problems by observing deviations of the ratio from values given by the model.

SOURCE:
Monitoring Software Development Through Dynamic Variables (Revision 1), SEL-83-106, p. 49

<table>
<thead>
<tr>
<th>Phase</th>
<th>Software Changes per Line of Code</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start coding</td>
<td>0.0076</td>
<td>0.0043</td>
</tr>
<tr>
<td>20% coding</td>
<td>0.0096</td>
<td>0.0046</td>
</tr>
<tr>
<td>40% coding</td>
<td>0.0072</td>
<td>0.0036</td>
</tr>
<tr>
<td>50% coding</td>
<td>0.0090</td>
<td>0.0033</td>
</tr>
<tr>
<td>60% coding</td>
<td>0.0089</td>
<td>0.0035</td>
</tr>
<tr>
<td>80% coding</td>
<td>0.0116</td>
<td>0.0048</td>
</tr>
<tr>
<td>Start system testing</td>
<td>0.0133</td>
<td>0.0050</td>
</tr>
<tr>
<td>50% system testing</td>
<td>0.0146</td>
<td>0.0057</td>
</tr>
<tr>
<td>Start acceptance testing</td>
<td>0.0153</td>
<td>0.0057</td>
</tr>
<tr>
<td>End acceptance testing</td>
<td>0.0201</td>
<td>0.0078</td>
</tr>
</tbody>
</table>

Software Changes per Line of Code

- Code/Link Test
- System Test
- Acceptance Test

Graph showing changes in software components per line of code across different phases.
TYPE: Environment Model

DESCRIPTION:
The model describes the behavior of the cumulative ratio of computer time per line of source code.

APPLICATION:

Prediction - Extrapolating the final value of the ratio based on the current observed value and the values given by the model.

Assessment - Detecting problems by observing deviations of the ratio from values given by the model.

SOURCE:
Monitoring Software Development Through Dynamic Variables (Revision 1), SEL-83-106, p. 50

<table>
<thead>
<tr>
<th>Phase</th>
<th>CPU Hours (tenths) per Line of Code</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start coding</td>
<td>0.0053</td>
<td>0.0054</td>
</tr>
<tr>
<td>20% coding</td>
<td>0.0077</td>
<td>0.0062</td>
</tr>
<tr>
<td>40% coding</td>
<td>0.0130</td>
<td>0.0071</td>
</tr>
<tr>
<td>50% coding</td>
<td>0.0178</td>
<td>0.0079</td>
</tr>
<tr>
<td>60% coding</td>
<td>0.0178</td>
<td>0.0068</td>
</tr>
<tr>
<td>80% coding</td>
<td>0.0220</td>
<td>0.0080</td>
</tr>
<tr>
<td>Start system testing</td>
<td>0.0287</td>
<td>0.0150</td>
</tr>
<tr>
<td>50% system testing</td>
<td>0.0357</td>
<td>0.0185</td>
</tr>
<tr>
<td>Start acceptance testing</td>
<td>0.0458</td>
<td>0.0226</td>
</tr>
<tr>
<td>End acceptance testing</td>
<td>0.0580</td>
<td>0.0275</td>
</tr>
</tbody>
</table>

Computer Time per Line of Code

Graph showing CPU hours per LOC over various phases and stages of software development.
TYPE: Environment Model

DESCRIPTION:
The model describes the behavior of the cumulative ratio of programmer hours per computer run.

APPLICATION:
Prediction – Extrapolating the final value of the ratio based on the current observed value and the values given by the model.
Assessment – Detecting problems by observing deviations of the ratio from values given by the model.

SOURCE:
Monitoring Software Development Through Dynamic Variables (Revision 1), SEL-83-106, p. 51

<table>
<thead>
<tr>
<th>Phase</th>
<th>Hours (in tenths) per Computer Run</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start coding</td>
<td>251.82</td>
<td>176.62</td>
</tr>
<tr>
<td>20% coding</td>
<td>75.26</td>
<td>40.18</td>
</tr>
<tr>
<td>40% coding</td>
<td>31.96</td>
<td>9.40</td>
</tr>
<tr>
<td>50% coding</td>
<td>26.71</td>
<td>7.97</td>
</tr>
<tr>
<td>60% coding</td>
<td>23.59</td>
<td>6.37</td>
</tr>
<tr>
<td>80% coding</td>
<td>19.35</td>
<td>5.07</td>
</tr>
<tr>
<td>Start system testing</td>
<td>17.46</td>
<td>4.32</td>
</tr>
<tr>
<td>50% system testing</td>
<td>16.41</td>
<td>3.94</td>
</tr>
<tr>
<td>Start acceptance testing</td>
<td>15.87</td>
<td>3.68</td>
</tr>
<tr>
<td>End acceptance testing</td>
<td>15.45</td>
<td>3.59</td>
</tr>
</tbody>
</table>
SOFTWARE CHANGES PER COMPUTER RUN BY PHASE

Result 4-12

TYPE: Environment Model

DESCRIPTION:
The model describes the behavior of the cumulative ratio of software changes per computer run.

APPLICATION:
Prediction — Extrapolating the final value of the ratio based on the current observed value and the values given by the model.
Assessment — Detecting problems by observing deviations of the ratio from values given by the model.

SOURCE:
Monitoring Software Development Through Dynamic Variables (Revision 1), SEL-83-106, p. 52

<table>
<thead>
<tr>
<th>Phase</th>
<th>Software Changes per Computer Run</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start coding</td>
<td>1.0240</td>
<td>1.2750</td>
</tr>
<tr>
<td>20% coding</td>
<td>0.3052</td>
<td>0.2073</td>
</tr>
<tr>
<td>40% coding</td>
<td>0.1714</td>
<td>0.1140</td>
</tr>
<tr>
<td>50% coding</td>
<td>0.1670</td>
<td>0.0999</td>
</tr>
<tr>
<td>60% coding</td>
<td>0.1583</td>
<td>0.0863</td>
</tr>
<tr>
<td>80% coding</td>
<td>0.1588</td>
<td>0.0691</td>
</tr>
<tr>
<td>Start system testing</td>
<td>0.1604</td>
<td>0.0561</td>
</tr>
<tr>
<td>50% system testing</td>
<td>0.1552</td>
<td>0.0492</td>
</tr>
<tr>
<td>Start acceptance testing</td>
<td>0.1563</td>
<td>0.0476</td>
</tr>
<tr>
<td>End acceptance testing</td>
<td>0.1611</td>
<td>0.0388</td>
</tr>
</tbody>
</table>
**TYPE:** Environment Model

**DESCRIPTION:**
The model describes the behavior of the cumulative ratio of computer time per computer run.

**APPLICATION:**
*Prediction* — Extrapolating the final value of the ratio based on the current observed value and the values given by the model.

*Assessment* — Detecting problems by observing deviations of the ratio from values given by the model.

**SOURCE:**
*Monitoring Software Development Through Dynamic Variables (Revision 1)*, SEL-83-106, p. 53

**COMPUTER TIME PER COMPUTER RUN BY PHASE**

<table>
<thead>
<tr>
<th>Phase</th>
<th>CPU Hours (tenths) per Computer Run</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start coding</td>
<td>0.1760</td>
<td>0.0498</td>
</tr>
<tr>
<td>20% coding</td>
<td>0.1944</td>
<td>0.0686</td>
</tr>
<tr>
<td>40% coding</td>
<td>0.2740</td>
<td>0.0740</td>
</tr>
<tr>
<td>50% coding</td>
<td>0.3197</td>
<td>0.0808</td>
</tr>
<tr>
<td>60% coding</td>
<td>0.3333</td>
<td>0.0935</td>
</tr>
<tr>
<td>80% coding</td>
<td>0.3581</td>
<td>0.1061</td>
</tr>
<tr>
<td>Start system testing</td>
<td>0.4056</td>
<td>0.0999</td>
</tr>
<tr>
<td>50% system testing</td>
<td>0.4453</td>
<td>0.0945</td>
</tr>
<tr>
<td>Start acceptance testing</td>
<td>0.4857</td>
<td>0.1029</td>
</tr>
<tr>
<td>End acceptance testing</td>
<td>0.5472</td>
<td>0.1371</td>
</tr>
</tbody>
</table>

**Computer Time per Computer Run**

[Graph showing CPU hours per computer run by phase with phases labeled as: Code/Unit Test, System Test, and Acceptance Test.]
TYPE: Environment Model

DESCRIPTION:
The model describes the behavior of the cumulative ratio of programmer hours per software change.

APPLICATION:
- Prediction — Extrapolating the final value of the ratio based on the current observed value and the values given by the model.
- Assessment — Detecting problems by observing deviations of the ratio from values given by the model.

SOURCE:
Monitoring Software Development Through Dynamic Variables (Revision 1), SEL-83-106, p. 54

<table>
<thead>
<tr>
<th>Phase</th>
<th>Hours (in tenths) per Software Change</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start coding</td>
<td>437.50</td>
<td>249.57</td>
</tr>
<tr>
<td>20% coding</td>
<td>297.29</td>
<td>204.09</td>
</tr>
<tr>
<td>40% coding</td>
<td>292.66</td>
<td>224.14</td>
</tr>
<tr>
<td>50% coding</td>
<td>235.36</td>
<td>166.41</td>
</tr>
<tr>
<td>60% coding</td>
<td>215.22</td>
<td>154.85</td>
</tr>
<tr>
<td>80% coding</td>
<td>141.77</td>
<td>75.62</td>
</tr>
<tr>
<td>Start system testing</td>
<td>118.81</td>
<td>58.36</td>
</tr>
<tr>
<td>50% system testing</td>
<td>110.54</td>
<td>47.61</td>
</tr>
<tr>
<td>Start acceptance testing</td>
<td>106.91</td>
<td>47.66</td>
</tr>
<tr>
<td>End acceptance testing</td>
<td>94.03</td>
<td>41.60</td>
</tr>
</tbody>
</table>
### COMPUTER TIME PER SOFTWARE CHANGE BY PHASE

**Representation:**

<table>
<thead>
<tr>
<th>Phase</th>
<th>CPU Hours per Software Change</th>
<th>Standard Deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Start coding</td>
<td>0.426</td>
<td>0.346</td>
</tr>
<tr>
<td>20% coding</td>
<td>0.871</td>
<td>0.616</td>
</tr>
<tr>
<td>40% coding</td>
<td>2.542</td>
<td>1.977</td>
</tr>
<tr>
<td>60% coding</td>
<td>3.058</td>
<td>2.962</td>
</tr>
<tr>
<td>80% coding</td>
<td>3.437</td>
<td>3.740</td>
</tr>
<tr>
<td>100% coding</td>
<td>2.802</td>
<td>2.112</td>
</tr>
<tr>
<td>Start system testing</td>
<td>2.804</td>
<td>1.643</td>
</tr>
<tr>
<td>50% system testing</td>
<td>2.950</td>
<td>1.404</td>
</tr>
<tr>
<td>Start acceptance testing</td>
<td>3.219</td>
<td>1.501</td>
</tr>
<tr>
<td>50% system testing</td>
<td>3.652</td>
<td>1.334</td>
</tr>
<tr>
<td>End acceptance testing</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**

- **X-axis:** CPU Hours (minutes) per Component Change
- **Y-axis:** Computer Time per Software Change

**Source:**

- Monitoring Software Development Through Dynamic Variables (Revision 1).
- SEI-88-106, p. 55
UNCERTAINTY IN EFFORT AND SIZE ESTIMATES BY PHASE

TYPE: Uncertainty Model

DESCRIPTION:
The model describes the relative uncertainty to expect in effort and size estimates made for a project at various times in the software development life-cycle.

NOTE: Values for effort and size should be reestimated at the end of each phase as additional information becomes available about the eventual size of the completed project. The model reflects the manner in which the uncertainty in estimates decrease as more accurate data are known.

APPLICATION:
Planning—Provides a range of confidence for an initial estimate of effort or size and for any reestimates made during subsequent phases of the software development life-cycle.

SOURCE:
Manager’s Handbook for Software Development (Revision 1), SEL-84-101; pp. 3-2, 3-3

<table>
<thead>
<tr>
<th>End of Phase</th>
<th>Estimate Uncertainty</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Requirements Definition</td>
<td>1.00</td>
<td>0.50</td>
<td>2.00</td>
</tr>
<tr>
<td>Requirements Analysis</td>
<td>0.75</td>
<td>0.57</td>
<td>1.75</td>
</tr>
<tr>
<td>Preliminary Design</td>
<td>0.40</td>
<td>0.71</td>
<td>1.40</td>
</tr>
<tr>
<td>Detailed Design</td>
<td>0.25</td>
<td>0.80</td>
<td>1.25</td>
</tr>
<tr>
<td>Implementation</td>
<td>0.10</td>
<td>0.91</td>
<td>1.10</td>
</tr>
<tr>
<td>System Testing</td>
<td>0.05</td>
<td>0.95</td>
<td>1.05</td>
</tr>
</tbody>
</table>

where:
The uncertainty may be applied to effort or size estimates made at the end of each named life-cycle phase as follows:

Estimateupper bound = Estimate * (1.0 + uncertainty)
Estimatenower bound = Estimate / (1.0 + uncertainty)

Uncertainty of Estimates by Phase
**TYPE:** Subjective Model

**DESCRIPTION:**

The model describes the behavior of the measure of effort to make a software change.

**APPLICATION:**

Assessment – Providing a characteristic pattern of behavior for the effort to make a software change that can be compared to actual project behavior to detect problems.

**SOURCE:**


**REPRESENTATION:**

"The amount of manpower to make a change should increase toward the end of the project and be stable at the beginning."

---

1 Report originally prepared as a University of Maryland Technical Memorandum dated December 1982.
TRENDS IN CHANGES PER COMPUTER RUN BY PHASE

<table>
<thead>
<tr>
<th>TYPE: Subjective Model</th>
<th>REPRESENTATION:</th>
</tr>
</thead>
</table>
| DESCRIPTION:           | "The ratio of changes to computer runs should decrease as the project evolves."
| The model describes the behavior of the ratio of software changes to computer runs. | |

APPLICATION:
Assessment – Providing a characteristic pattern of behavior for the ratio of software changes to computer runs that can be compared to actual project behavior to detect problems.

SOURCE:

---

Software Changes per Run

<table>
<thead>
<tr>
<th>Implementation</th>
<th>System Test</th>
<th>Acc. Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Relative Changes/Run</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

Life-Cycle Phase

---

1 Report originally prepared as a University of Maryland Technical Memorandum dated December 1982.
TYPE: Subjective Model

DESCRIPTION:

The model describes the behavior of the measure of computer time used to make a change.

APPLICATION:

Assessment – Providing a characteristic pattern of behavior for the computer time used to make a change that can be compared to actual project behavior to detect problems.

SOURCE:


1 Report originally prepared as a University of Maryland Technical Memorandum dated December 1982.
SECTION 5—MANAGEMENT RULES

The term "management rule" refers to a SEL research result that specifies how to interpret the observed behavior of a software development measure. These rules are formal statements typically used in evaluating data collected during ongoing development efforts. Rules can stand alone or can be combined to work together to give more detailed conclusions.

Automating the use of management rules is a recent area of interest in the SEL, although many of the rules listed here were recorded in early SEL research. Managers have always applied informal rules to the process of assessing a project's status.

The following paragraphs describe three groups of rules.

- **Independent rules** stand alone and are applicable to a single specific situation. An observation is interpreted with no mention of when the observation is made during the life cycle nor is the measure compared to a model to aid in drawing the conclusion.

  Independent rules are typical of what managers think of as "rules of thumb" or "project management lore." They usually state something about the overall status of the project.

- **Model-dependent rules** require a standard, or model, to which a measure is first compared. Based on the comparison and an observed deviation from the standard, an interpretation is presented. This type of rule was implied by the "assessment" applications described for models in Section 4. This type of rule requires both a model of the measure and possibly an estimate of the completion value for the measure.

  During the life of a project, a manager charts software development data to monitor progress. By comparing the actual progress to a model, the manager can spot deviations earlier. Model-dependent rules describe the consequences of a particular deviation based on observations of previous projects.

- **Phase-dependent rules** are model-dependent rules that are made more elaborate by introducing the life-cycle phase into the rule. This allows different conclusions to be drawn based on the time at which the measure deviated from the standard.

Sets of rules (independent, model-dependent, and/or phase-dependent) can be created to provide more depth on which to draw conclusions. While each rule will probably not result in the same conclusion, the advantage of a set of rules comes from
“voting” the conclusions. The one conclusion that is most prevalent (or better yet, in the majority) has a good likelihood of being appropriate to the project’s situation. This is similar to managers “taking everything into account.”

The following pages describe a selection of rules published by the SEL.
TYPE: Independent Rule

DESCRIPTION:
The rule describes the type of behavior of the measure of effort to make a software change that indicates a poor quality development process.

APPLICATION:
Assessment – Detecting the lack of a quality development process by observing the described behavior.

SOURCE:

REPRESENTATION:
"Projects deemed less successful by subjective analysis have sharp changes in the amount of manpower spent per change."

\[
\text{If} \quad \text{the effort to make a change is varying sharply} \\
\text{then} \quad \text{the project will be less successful}
\]

---

1Report originally prepared as a University of Maryland Technical Memorandum dated December 1982.
TYPE: Independent Rule

DESCRIPTION:

The rule describes the type of behavior of the actual staffing level that indicates a poor quality development process.

APPLICATION:

Assessment — Detecting the lack of a quality development process by observing the described behavior.

SOURCE:

Measures and Metrics for Software Development, SEL-83-002, p. 4-22

REPRESENTATION:

"Departures from planned staffing usually indicate that production will depart from plan, too."

If

the staffing level is varying from the plan

then

production will not match the plan

---

Departure from Staffing Plan

<table>
<thead>
<tr>
<th>Design</th>
<th>Implementation</th>
<th>System Test</th>
<th>Acc. Test</th>
</tr>
</thead>
</table>

---

Hours/Week

Total SLOC

Schedule

---

Planned

Actual
VARIATIONS IN SIZE AND EFFORT ESTIMATES

TYPE: Independent Rule

DESCRIPTION:
The rule describes the type of behavior of estimates of system size and total effort that indicates a poor quality development process.

APPLICATION:
Assessment — Detecting the lack of a quality development process by observing the described behavior.

SOURCE:
Measures and Metrics for Software Development, SEL-83-002, p. 4-22

REPRESENTATION:
"Frequent changes in size and cost estimates can imply that the estimates are being adjusted to fit an inappropriate staffing level or that the skill mix of the development team is inappropriate to the task."

If size or effort estimates are changing frequently then the estimates are being adjusted to match the wrong staffing level or the development team has the wrong skill mix.

History of Size Estimates

<table>
<thead>
<tr>
<th></th>
<th>Design</th>
<th>Implementation</th>
<th>System Test</th>
<th>Acc. Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Estimated Size at Completion</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Frequent changes in the estimates may imply:
1. Wrong staffing level
2. Wrong skill mix

Point in Schedule Where Estimate Made

- Normal Range
- Unstable Project

5-3
**HIGH COMPUTER USE**

**TYPE:** Model-Dependent Rule

**DESCRIPTION:**
The rule describes the type of behavior of the measure of computer use that indicates a poor quality development process.

**APPLICATION:**
Assessment — Detecting the lack of a quality development process by observing the described behavior.

**SOURCE:**

**REPRESENTATION:**
"Extensive computer use is associated with low productivity."

If
- computer use is higher than expected
then
- productivity will be low

---

**DEVIATIONS IN COMPUTER USE PER LINE OF CODE**

**TYPE:** Phase-Dependent Rule

**DESCRIPTION:**

The rule describes the type of behavior of the measure of computer use that indicates a poor quality development process.

**APPLICATION:**

Assessment — Diagnosing the underlying reasons if the computer use per line of code is observed to depart from normal behavior.

**SOURCE:**

*Measures and Metrics for Software Development*, SEL-83-002, pp. 4-6, 4-9

---

**REPRESENTATION:**

If the phase is design and computer use is high then there is insufficient design effort being expended.

If the phase is implementation or test and computer use is low then there is insufficient testing being done or there is low productivity.

If the phase is test and computer use is high then the project is in an integration crunch.

---

**Computer Use Pattern**

![Computer Use Pattern Diagram](attachment:image.png)
DEVIANIONS IN LINES OF CODE

TYPE: Phase-Dependent Rule

DESCRIPTION:
The rule describes the type of behavior of the measure of software size that indicates a poor quality development process.

APPLICATION:
Assessment—Diagnosing the underlying reasons if the lines of code measure is observed to depart from normal behavior.

SOURCE:
Measures and Metrics for Software Development, SEL-83-002, pp. 4-7, 4-8

REPRESENTATION:

If the phase is design and lines of code is high
then there is inadequate design effort being expended

If the phase is implementation and lines of code is high
then there is inadequate unit testing being done

If the phase is implementation or test and lines of code is low
then the development effort is behind schedule

Software Production Pattern

[Diagram showing software production pattern with phases and points identifying issues like inadequate testing and development behind schedule]
DEVIATIONS IN CHANGES PER LINE OF CODE

TYPE: Phase-Dependent Rule

DESCRIPTION:
The rule describes the type of behavior of the measure of change rate that indicates a poor quality development process.

APPLICATION:
Assessment — Diagnosing the underlying reasons if the change rate measure is observed to depart from normal behavior.

SOURCE:
Measures and Metrics for Software Development, SEL-83-002, pp. 4-10 to 4-12

REPRESENTATION:

If the phase is implementation or test and changes per line of code is high then the software is unstable

If the phase is implementation or test and changes per line of code is low then there is insufficient testing being done

Software Change Pattern

<table>
<thead>
<tr>
<th>Implementation</th>
<th>System Test</th>
<th>Acc. Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Changes Per Total Lines of Code

Percent of Schedule Elapsed

- Software Instability
- Nominal Range
- Insufficient Testing
DEVIATIONS IN LINES OF CODE PER STAFF HOUR

RESULT 5-8

TYPE: Phase-Dependent Rule

DESCRIPTION:
The rule describes the type of behavior of a measure of productivity that indicates a poor quality development process.

APPLICATION:
Assessment—Diagnosing the underlying reasons if the productivity measure is observed to depart from normal behavior.

SOURCE:
Measures and Metrics for Software Development, SEL-83-002, pp. 4-20, 4-21

REPRESENTATION:

If the phase is implementation and lines of code per hour is high then there is extensive code being reused or there is inadequate unit testing being done

If the phase is implementation and lines of code per hour is low then the development effort is behind schedule

If the phase is test and lines of code per hour is high then the implementation is incomplete

If the phase is test and lines of code per hour is low then the development is in an integration crunch

Productivity Pattern

<table>
<thead>
<tr>
<th>Implementation</th>
<th>System Test</th>
<th>Acc. Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>Inadequate Testing</td>
<td>Implementation Incomplete</td>
<td></td>
</tr>
<tr>
<td>Extensive Reused Code</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Normal Productivity Range</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Developed Behind Schedule</td>
<td></td>
<td>Integration Crunch</td>
</tr>
</tbody>
</table>

- Total Lines of Code per Staff Hour
- Percent of Schedule Elapsed
# GLOSSARY

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>central processing unit</td>
</tr>
<tr>
<td>CSC</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>FTE</td>
<td>full-time equivalent</td>
</tr>
<tr>
<td>GSFC</td>
<td>Goddard Space Flight Center</td>
</tr>
<tr>
<td>KSLOC</td>
<td>source lines of code in thousands</td>
</tr>
<tr>
<td>LOC</td>
<td>lines of code</td>
</tr>
<tr>
<td>NASA</td>
<td>National Aeronautics and Space Administration</td>
</tr>
<tr>
<td>SEL</td>
<td>Software Engineering Laboratory</td>
</tr>
<tr>
<td>SLOC</td>
<td>source lines of code</td>
</tr>
<tr>
<td>SME</td>
<td>Software Management Environment</td>
</tr>
<tr>
<td>TBD</td>
<td>to be determined</td>
</tr>
</tbody>
</table>
REFERENCES

2. SEL-81-101, Guide to Data Collection, V. E. Church, D. N. Card, and F. E. McGarry, August 1982
3. SEL-87-008, Data Collection Procedures for the Rehosted SEL Database, G. Heller, October 1987
13. SEL-82-007, Proceedings of the Seventh Annual Software Engineering Workshop, December 1982
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-77-005, *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-79-005, *Proceedings From the Fourth Summer Software Engineering Workshop*, November 1979


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

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

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

SEL-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, F. E. McGarry, and M. G. Rohleder, October 1983

SEL-82-906, Annotated Bibliography of Software Engineering Laboratory Literature, P. Groves and J. Valett, November 1990


SEL-83-007, Proceedings From 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 From the Ninth Annual Software Engineering Workshop, November 1984


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


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


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

SEL-86-005, Measuring Software Design, D. N. Card, October 1986

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


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


SEL-87-008, Data Collection Procedures for the Rehosted SEL Database, G. Heller, October 1987


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

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


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


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


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


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


SEL-90-004, 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-RELATED LITERATURE**


BI-8


BI-9


BI-11


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.


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.


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