Naval Research Laboratory
Washington, DC 20375-5320

AD-A278 759

NRL/MR/8140--94-7479

Developing Software For Ease of Change: Metrics From Later in the TRALAB System Life Cycle

JAMES A. HAGER

HRB Systems, Inc.
State College, PA 16804

LOUIS J. CHMURA, JR.

Command Control Communication Computers & Intelligence
Naval Center for Space Technology

DTIC
MAY 0 4 1994

April 11, 1994

94-13230

Approved for public release; distribution unlimited.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (*Leave Blank*)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	April 11, 1994	Interim

4. TITLE AND SUBTITLE

Developing Software For Ease of Change: Metrics From Later in the TRALAB System Life Cycle

5. FUNDING NUMBERS

6. AUTHOR(S)

James A. Hager* and Louis J. Chmura, Jr.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Research Laboratory
Washington, DC 20375-5320

8. PERFORMING ORGANIZATION REPORT NUMBER

NRL/MR/8140--94-7479

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

*HRB Systems, Inc., State College, PA 16804

12a. DISTRIBUTION/AVAILABILITY STATEMENT

Approved for public release; distribution unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT (*Maximum 200 words*)

This paper is a summary of software change and defect data collected during extended development and post-deployment software support (PDSS) for the Training Laboratory (TRALAB), a computer-based training system developed starting in 1984 by the Naval Center for Space Technology (NCST) at the Naval Research Laboratory (NRL). Of note, the development contractor was required to apply software engineering technology developed previously at NRL as part of the Software Cost Reduction (SCR) project. Another important requirement was for the collection of project metric data that could be used to evaluate the effectiveness of the SCR approach. Data collection was accomplished by modifying customary project Software Problem Reports (SPRs).

Analysis of the data collected on SPR resolution work during extended development integration and test (I&T) during August 1988 through April 1989 indicates that the application of SCR technology enhanced software ease of change. This paper is a continuation of the earlier analysis to include PDSS SPR data collected during August 1990 through May 1991. The more recent analysis continues to suggest that identifying expected system changes during system definition stages and modularizing the system to encapsulate these changes yield life-cycle benefits. Modifications required by changes tend to be confined to a small number of design components, and it is easier to implement expected changes in comparison to arbitrary changes.

14. SUBJECT TERMS

Computer programs	Ease of change	Information hiding
Software	PDSS	SCR
Metrics	TRALAB	

15. NUMBER OF PAGES

28

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500 Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std 239-18
298-102

CONTENTS

Accession For
NTIS GRA&I ☑
DTIC TAB ☐
Unannounced ☐
Justification
By
Distribution/
Availability Codes
Dist | Avail and/or Special
A-1

DEVELOPING SOFTWARE FOR EASE OF CHANGE: METRICS FROM LATER IN THE TRALAB SYSTEM LIFE CYCLE

1.0 Introduction

This paper is a summary of software change and defect data collected during extended development and post-deployment software support (PDSS) for the Training Laboratory (TRALAB), a computer-based training system developed starting in 1984 by the Naval Center for Space Technology (NCST) at the Naval Research Laboratory (NRL). Hager [89] describes the TRALAB project in detail. Of note, the development contractor was required to apply software engineering technology developed previously at NRL as part of the Software Cost Reduction (SCR) project [CLEMENTS 86; HENINGER 80; PARNAS 84]. Another important requirement was for the collection of project metric data that could be used to evaluate the effectiveness of the SCR approach. Data collection was accomplished by modifying customary project Software Problem Reports (SPRs).

Analysis of the data collected on SPR resolution work during extended development integration and test (I&T) during August 1988 through April 1989 indicates that the application of SCR technology enhanced software ease of change [HAGER 91]. This paper is a continuation of the earlier analysis to include PDSS SPR data collected during August 1990 through May 1991. The more recent analysis continues to suggest that identifying expected system changes during system definition stages and modularizing the system to encapsulate these changes yield life-cycle benefits. Modifications required by changes tend to be confined to a small number of design components, and it is easier to implement expected changes in comparison to arbitrary changes.

The remainder of this paper comprises five sections. Section 2 is a summary of NRL's earlier SCR project, fundamental SCR design and documentation concepts, and the TRALAB project and architecture. Section 3 contains analyses of the SPR data. Section 4 provides acknowledgments; section 5, references. Appendix A provides the Software Problem Report (SPR) and Software Modification Transmittal (SMT) forms used to collect and report the TRALAB change and problem data.

2.0 Background

2.1 Problem

The difficulty of generating software that is easily maintained is evident when full software life-cycle costs are examined. Typically, PDSS efforts account for 60% or more of the total life-cycle costs [BOEHM 73]. One reason for such a situation may be that ease of maintenance is not a natural by-product of many currently used development approaches and methodologies. For example:

- maintainability requirements are omitted or not clearly identified in system requirements specifications,
- verification of maintainability requirements is imprecise at best,
- documentation structures do not provide enough visibility for maintainability concerns, and
- design approaches do not address directly how to ensure that modifications required by eventual likely changes will not ripple throughout the design and implementation.

2.2 Software Cost Reduction Approach

Controlling software maintenance costs would seem to require changes in both software design approaches [CLEMENTS 86; LAMB 88; CLEMENTS 83; WALLACE 87; PARNAS 72] and supporting documentation structures [HESTER 81]. In 1978, the SCR program was initiated by NRL and the Naval Weapons Center to develop and illustrate software technology that could be

Manuscript approved March 3, 1994.

used to control software life-cycle costs. Analyses of early SCR project metrics showed patterns that suggest that SCR technology can help to control such costs [CHMURA 90].

A fundamental aspect of the SCR approach was the principle of information hiding, a software design concept first introduced by Parnas [72]. With information hiding, a software "module" is treated both as a unit of work and as a unit of rework. Accordingly, software designers should consider likely future changes when identifying software modules to prevent unpredictable and possibly wide-ranging rework in the future. The first step in effectively applying the information hiding principle is to identify expected changes early in the requirements specification process. Once the expected changes are agreed upon, they are prioritized based on their likelihood of occurrence. The expected changes are then factored explicitly into the design of the system. The general approach is to identify and specify modules that limit the rework required of future changes by trying to confine or "hide" each expected change in separate modules, which are referred to as information-hiding modules. The information that will be hidden behind an "abstract" programming interface to an information-hiding module is referred to as the "secret" of the module. The TRALAB design architecture described in section 2.4 below is an information-hiding module structure.

2.3 TRALAB Program

The TRALAB program started in September 1984 with the purpose of developing a computer-based training system to support several data collection and processing systems developed and maintained by the NCST at NRL [HAGER 89]. Following a successful System Definition phase, NRL redirected the effort to incorporate SCR concerns. For example, a list of expected TRALAB changes was generated and refined over the course of the initial development, and an information-hiding design approach was used. The rationale for the redirection was (1) to reduce the risks associated with the history of frequent and significant upgrades experienced with the NCST data collection and processing systems, and (2) to experiment with the SCR technology in the development of a new NCST system.

A TRALAB system comprises 26 Zenith 286 personal computers, networked via Ethernet. System users consist of Students, Courseware Authors, Instructors, and Administrators. Instructors have the ability to interact and monitor students engaged in training on TRALAB computers.

A TRALAB system consists of approximately 49,000 lines of non-commented logical source lines (NCLSLOC) [BAUMERT 92] of Pascal code, 25,000 database records, and an additional 10,000 scenario script records. Logical source statements have a defined beginning and ending, independent of any relationship to the physical lines on which it is recorded or printed. Descriptions of user-interface menus and forms, as well as messages and displays characteristic of a "target system" for which TRALAB would offer computer-based training are stored as database entries to facilitate easy capture and change. Scenarios are script files used to control the flow of the training simulations. Scenarios are created and maintained by trainers using utilities generated as part of the development and, by design, do not require programming skills to develop or maintain.

TRALAB modules are implemented as Pascal packages (non-ANSI standard implementation). Pascal packages are similar to the Ada package construct and provide support for information hiding concerns through separately compilable interfaces (visible to a programmer) and implementations (hidden).

Table 2.3-1 is a list of the latest sizes of each of the third-level modules in terms of NCLSLOC. The NCLSLOC count for Simulation Activity is an aggregate. The Simulation Activity module comprises three fourth-level target-system-specific modules.

Table 2.3-1 Third-Level Module Size

MODULE	SOURCE LINES OF CODE (NCLSLOC)
HARDWARE-HIDING	
VIRTUAL OPERATING SYSTEM	2183
VIRTUAL NETWORK	1071
VIRTUAL TERMINAL	2869
BEHAVIOR-HIDING	
SCENARIO MAINTENANCE	3455
DATA BASE EDITOR	1675
SYSTEM ADMINISTRATION	1740
SIMULATION ACTIVITY	12254
TRAINING EVALUATION	1724
STUDENT OBSERVATION	946
MENU PROCESSING	4599
CONTROL	718
SOFTWARE DECISION-HIDING	
SCENARIO TRANSLATION	2563
SIMULATION DELIVERY	3417
VIRTUAL DATABASE	602
SYSTEM DATA	2622
TARGET DATA	2556
SYSTEM ADMINISTRATION DATA	1946
OPERATIONAL DATA	1631

2.4 TRALAB Module Structure

The TRALAB module structure is hierarchical where a submodule of a module hides one or more of the secrets of the parent module. The first two levels of the design hierarchy provide an organizational road map for identifying what collections of software modules/components might need to be modified in response to a change. These upper levels of the information-hiding design are somewhat generic and may apply to many systems. Modules at level 3 and below tend to be objects that will be implemented in code (e.g., as Pascal or ADA packages). There are four levels in the TRALAB hierarchy, the first level of which consisted of the following:

- Hardware-Hiding module
- Behavior-Hiding module
- Software Decision-Hiding module

The designers of the TRALAB module structure did not adjust the design decomposition based on NCLSLOC size or complexity of a module implementation. The decomposition strategy relied entirely on encapsulating expected changes within distinct components of the system. This philosophy was a departure from traditional development approaches that limit module size and complexity to support span of control and testability concerns.

2.4.1 Hardware-Hiding Module

The Hardware-Hiding module comprises modules that would need to be modified when system or interface hardware is replaced or modified. The Hardware-Hiding module includes (i.e., is decomposed into) the Extended Computer and the Device Interface modules.

The Extended Computer module hides those characteristics of the computing platform that are likely to change if the computer or its operating system is modified or replaced. It offers a virtual computer for the remaining TRALAB software. The major submodule of the Extended Computer module is the Virtual Operating System (VOS) module that hides some of the peculiarities and changeable features of the OS version used initially for TRALAB. It should be noted that the VOS offers features similar to the initial OS; in other words, TRALAB software engineers were careful not to commit to design a new portable OS. One area where this philosophy influenced the development approach was the TRALAB executive module (TCN). A lack of tasking primitives in the underlying OS led to a polling approach to event notification and processing. No attempt was made to extend existing OS functionality to support real-time tasking concerns by adding new tasking primitives.

The Device Interface module comprises two third-level submodules - the Virtual Network Interface (VNI) and the Virtual Terminal (VTM) modules. The VNI module hides the commercial network software used. The abstract interface specifications written for this module identify abstract network primitives for establishing a circuit, sending a message, and receiving a message. If the technology for establishing a circuit or sending a message changes, other modules using the VNI service routines do not have to be modified.

The Virtual Terminal module insulates the system from changes to the terminal by providing abstract primitives for screen display and keyboard drivers. The display output device is managed as a set of windows, each with characteristics to simulate portions of target screen displays. The Virtual Terminal Interface hides the physical characteristics of the display device, locations of the devices, and windowing mechanisms. The virtual interface provides the capability to change physical screen characteristics without impacting existing software.

Figure 2.4.1-1 illustrates the composition of the Hardware-Hiding Module.

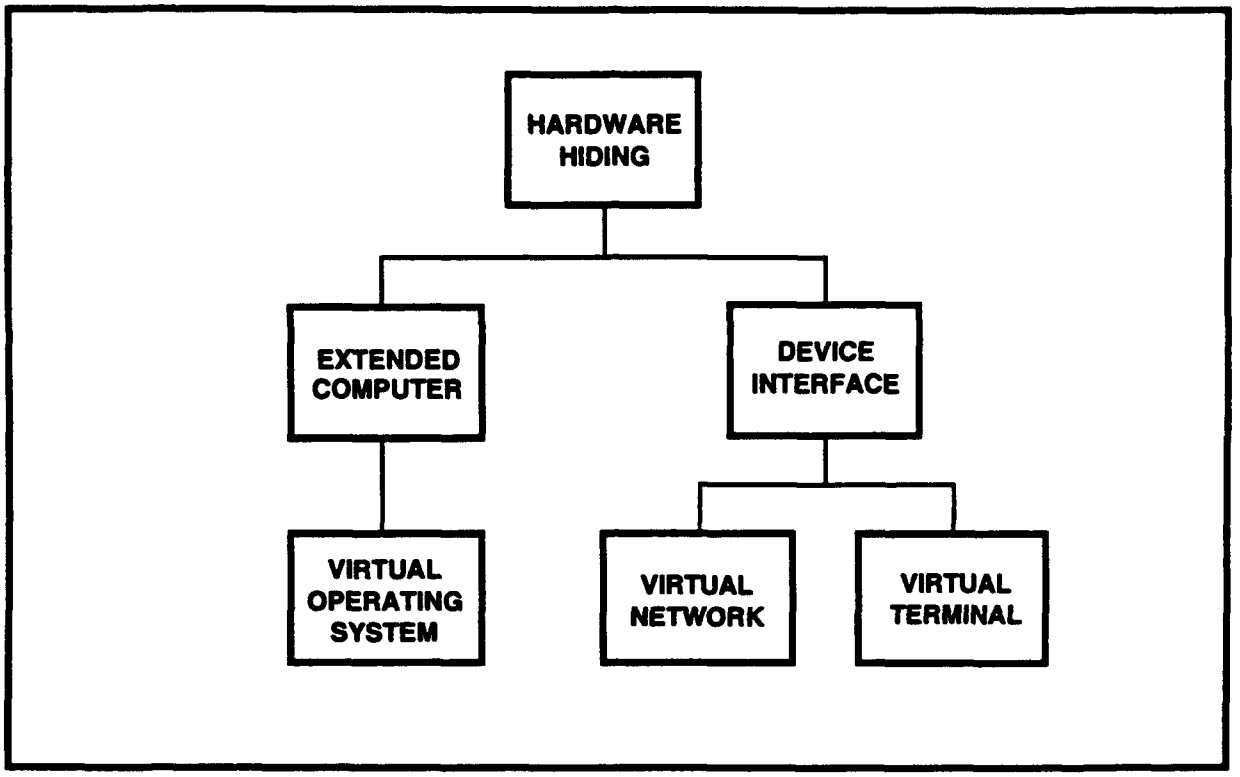

Figure 2.4.1-1. Hardware-Hiding Module and Submodules

2.4.2 Behavior-Hiding Module

The Behavior-Hiding module comprises modules that would need to be modified if there are changes to the required system behavior specified in the TRALAB System Technical Specification [HAGER 89]. The Behavior-Hiding module is decomposed into two second-level modules: the Application Driver module and the Shared Service module.

The Application Driver (AD) module is the sole controller of sets of closely related outputs. Each of its third-level submodules hides the rules determining the values of the outputs and the data structures and algorithms necessary to implement the outputs. Expected changes dealt with in the AD module include the authoring exchange necessary to create and maintain scenarios, the system administrator exchange necessary to maintain target system databases, the system administration classroom management policies, the processing unique to specific target system simulations, the student evaluation processing and criteria, and the student monitoring processing. A change in one of these areas should be confined to a specific third-level AD submodule.

The Shared Service module comprises software that controls required external behavior common to two or more AD modules. The Shared Service modules hide the characteristics of the shared behavior and the algorithms and data structures necessary to implement the shared behavior. Some of the secrets of the Shared Service module include AD module initialization, menu services, and control structures common to the AD modules. A change in any of these areas is isolated to

the Shared Service modules, even though the change may affect an external behavior shared by many application modules.

All required TRALAB system behavior is provided by the Behavior-Hiding modules. During early design work, design credibility is established by mapping the required system behavior identified in the System Requirements Specification to Behavior-Hiding modules.

Figure 2.4.2-1 illustrates the composition of the Behavior-Hiding Module.

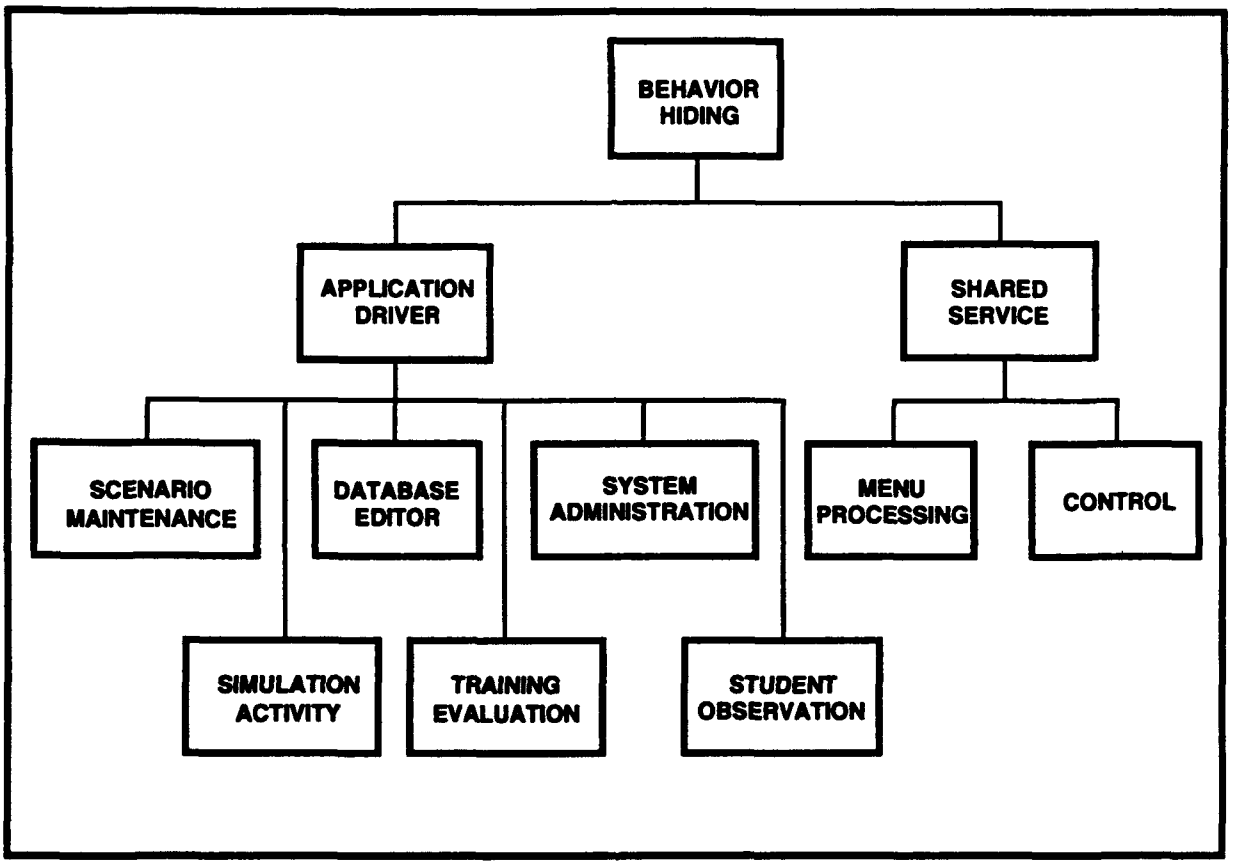

Figure 2.4.2-1. Behavior-Hiding Module and Submodules

2.4.3 Software Decision-Hiding Module

The Software Decision-Hiding module comprises modules that would need to be modified if there are changes to designer-generated decisions. For example, the choice of a specific algorithm not specified in the System Requirements Specification is a designer-generated decision.

The Software Decision-Hiding module is decomposed into three second-level modules: the Scenario Interface module, the Database Utilities module, and the System Generation module.

The Scenario Interface module hides changes to the training scenario validation policies, the translation process from the external scenario language used by the authors to the internal scenario primitives, and the execution of those primitives. All algorithms to parse, validate, translate, and

execute the scenarios are hidden in these modules. These changes were allocated to Software Decision-Hiding modules because the specific language implementation necessary to support required training was designer-determined.

The Database Utilities module consists of software that needs to be modified if changes are made to the database management system or to the internal storage, retrieval, or maintenance policies. To insulate application modules from the underlying database management system, a Virtual Database Interface module is provided. It provides the file management primitives necessary to support indexed sequential access data retrieval. Any changes to the data access policies are limited to this module. Target System, System Administration, Operational, and Scenario Data modules provide the Application Programming Interface (API) services for TRALAB databases. Knowledge of the physical representation of the TRALAB data was hidden from consumer modules through these abstract interfaces.

The System Generation module hides the expected changes related to the software processing environment and the underlying language. It hides the command structures necessary to compile and link the software, values of system generation parameters that select different implementations of a module, and specialized test software.

The Language Implementation module provides an area to discuss features unique to the specific implementation chosen. Originally, the goal was to abstract out the underlying language implementation. Since this was cost prohibitive, it provided an area to discuss the language specific decisions that might affect program portability.

Figure 2.4.3-1 illustrates the composition of the Software Decision-Hiding Module.

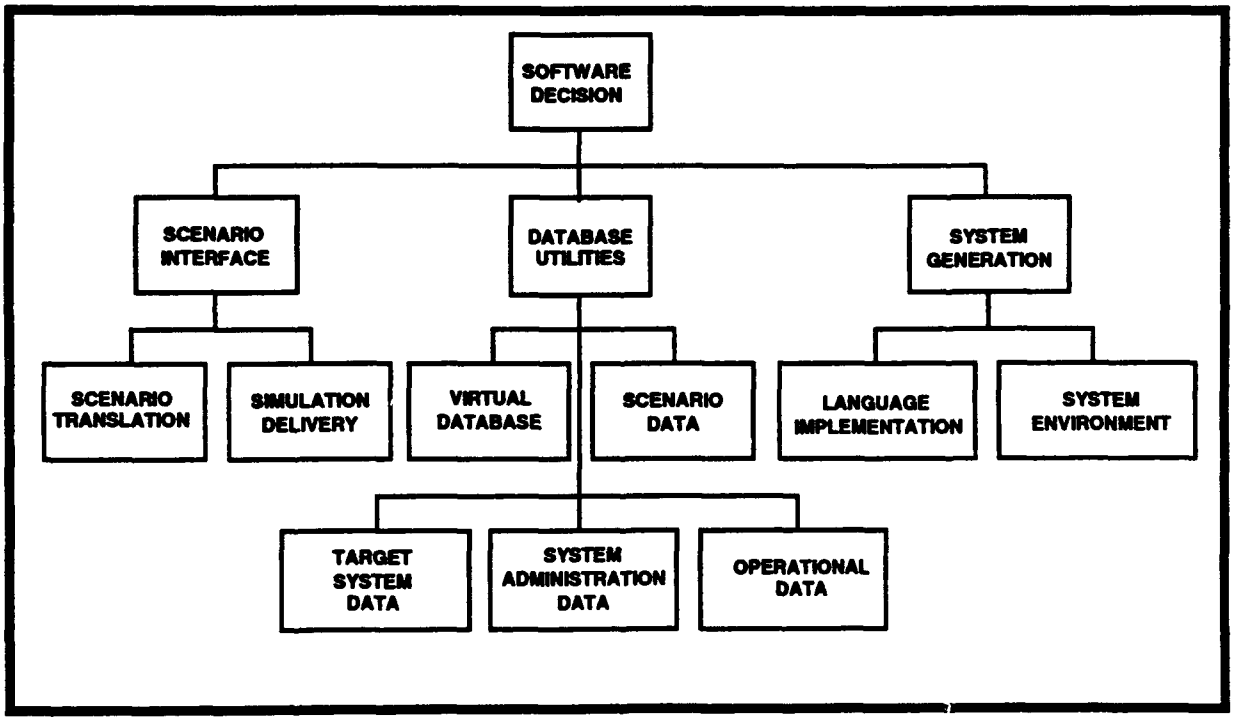

Figure 2.4.3-1 Software Decision-Hiding Module and Submodules

3.0 Data Analysis

3.1 General

TRALAB change data comprises developmental integration and test (I&T) data and post-deployment software support (PDSS) data collected during two time periods. The first period was from August 1988 to April 1989, and encompassed data on software design and code problems found during Software I&T activity and System I&T activity of a major TRALAB upgrade. The upgrade effort, which involved several thousand source lines of code and scenario/database updates, preceded initial TRALAB deployment and essentially extended the initial TRALAB development effort. All problems recorded during this period surfaced during the execution of customer-approved test plans and procedures, targeted at verifying required system behavior as documented in the Software Requirements Specification and the System Technical Specification. As such, all change data collected during this period was problem-oriented, i.e., not related to system enhancements or to discrepancies found in technical documentation. There were a total of 230 identified problems and proposed changes impacting software, scenarios, and databases for this "I&T period". All were recorded in software problem reports (SPRs), reviewed, and resolved under the direction of a contractor Configuration Control Board (CCB). The CCB met as needed based on the recent number and severity of the changes and problems reported. Fourteen (6%) of the problems were attributed to user error (i.e., were not considered system problems) and summarily closed with the resolution that no modifications were required. The remaining 216 problems required rework to designs, databases, code, or scenario scripts. Although documentation defects were not recorded with SPR forms, the effort recorded to close an SPR included the updates to related technical and user documentation. Two hundred and one of the remaining problems (93%), a surprisingly large proportion, were directly related to expected system changes that were listed in the System Technical Specification and refined during initial design.

The second time period was from August 1990 to May 1991, and encompassed data on problems uncovered following initial deployment of the system. There were 39 SPRs generated during this "PDSS period". Five (13%) of these were attributed to user error and summarily closed. Four of the remaining problems (10%) were unresolved and remain open. The other 30 SPRs resulted in updates. Of these, eighteen (60%) were directly related to expected system changes.

Unlike the I&T SPRs, PDSS SPRs included enhancements (i.e., 12) to existing TRALAB functionality. Following an extensive period of operational use, desired TRALAB functionality not originally specified in initial Software Requirement Specification was captured through the configuration management (CM) process.

Figures 3.1-1 is a plot of the accumulation of the 250 SPRs that required, or may yet require, TRALAB modifications. Figure 3.1-2 is a plot of the associated resolution/close-out activity. Although all the SPRs were handled through a typical multi-step CCB process, Figure 3.1-1 indicates that for the majority of the I&T SPRs, there were no significant overall delays between SPR submission and close-out (i.e., the final step during which the CCB verified complete and accurate implementation of the required modifications).

Appendix A contains the TRALAB SPR forms used to document change data. SPRs could be filled out by project engineers or by TRALAB system end users and submitted to the TRALAB Configuration Manager. The Configuration Manager would log the SPR and schedule a CCB

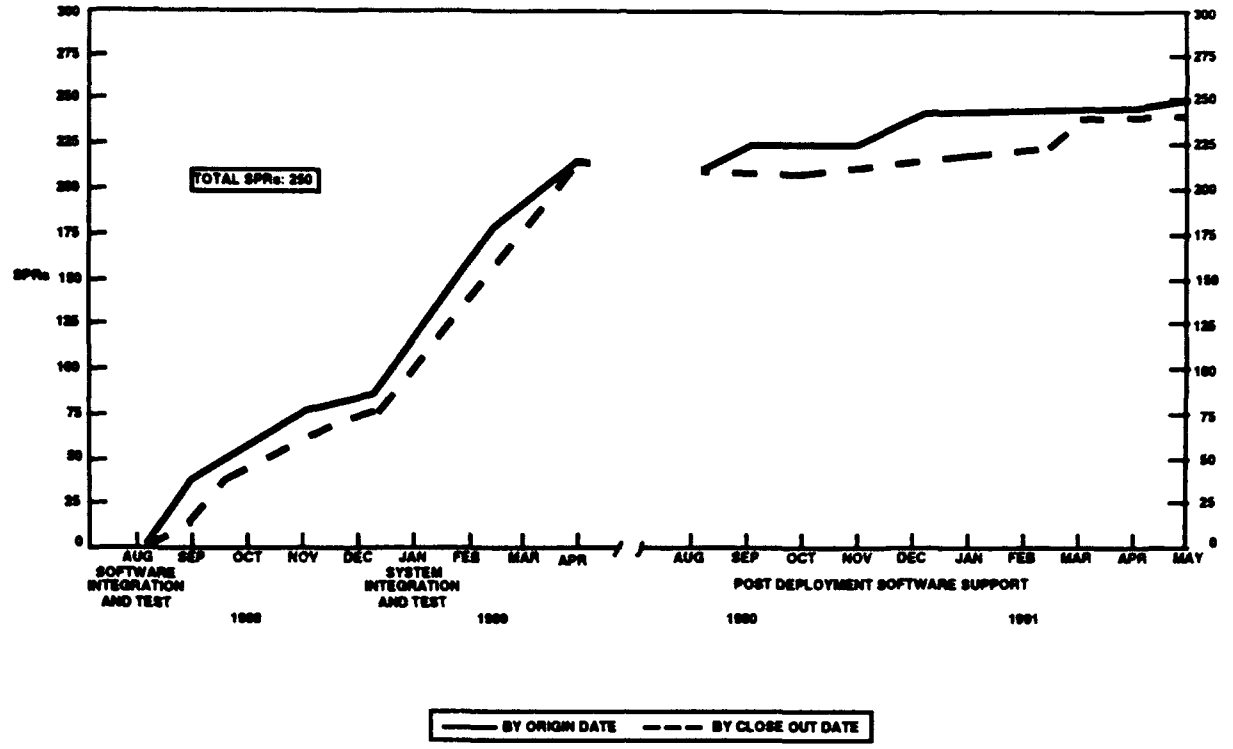

Figure 3.1-1 SPR Accumulation

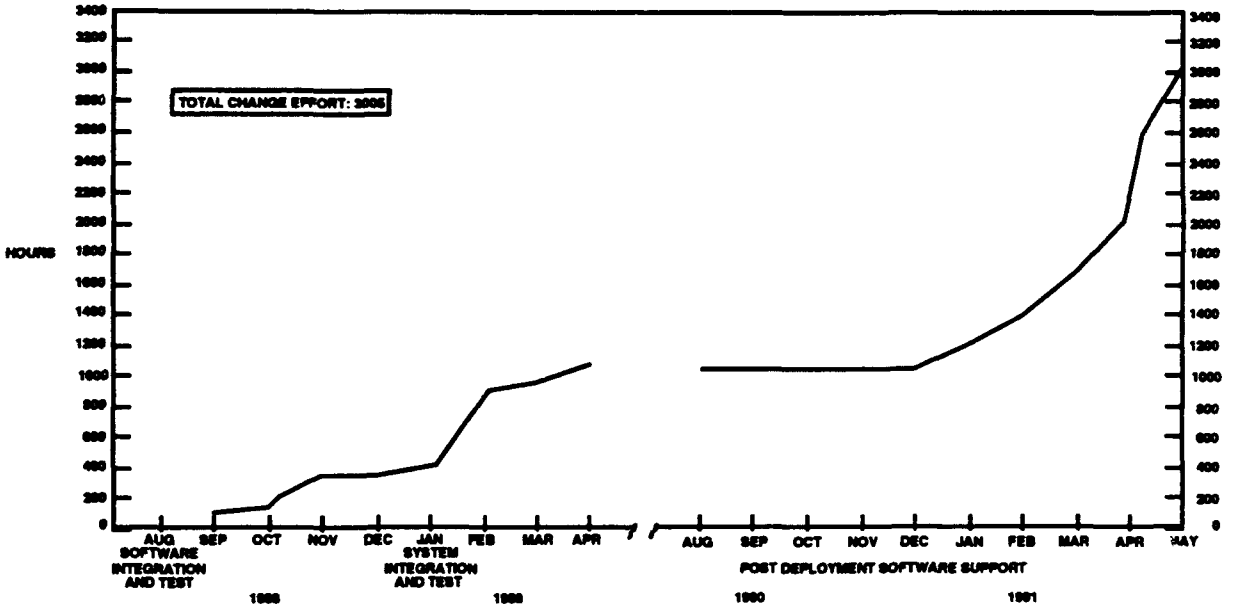

Figure 3.1-2 Cumulative SPR Resolution Effort

meeting when a sufficient number of SPRs were collected or when the severity of a specific SPR required immediate attention.

The CCB would validate the information on the SPR and determine a course of action. In most cases, a software engineer would be assigned the responsibility for identifying and making the necessary modifications to the software, databases, scenarios, and documentation, and performing regression testing. Following successful regression testing, the Software Modification Transmittal (SMT) form would be completed, logged, and submitted to the CCB for review and SPR close-out.

The SMT form was tailored to collect the information necessary in determining the utility of the SCR-based design decomposition approach and resulting module structure. For example:

- The TYPE OF CHANGE portion of the form contains entries that would indicate where modifications were necessary (i.e., in SOFTWARE, SCENARIO, DATABASE, DOCUMENTATION, and OTHER). TRALAB software designers had tried to encapsulate several of the most likely expected changes as database or scenario entries. Database editing and authoring tools were generated to facilitate database and scenario updates. There was interest if this extra design effort paid off.

- Number of software modules impacted by a change, and the lines of code required. Since modifications to scenarios or database files were made with utilities and did not require recompilation of code, such changes were designated as zero-module impacts.

- The EFFORT field included the engineering staff hours necessary to complete updates to the software, scenario and/or database files and to modify related technical and user documentation.

- The LABOR GRADE field allowed costs associated with SPR resolution to be calculated.

Not all the elements of the SMT proved useful. It proved difficult for project personnel to complete information on FAILURE CAUSE portion of the form. Accordingly, the following analyses omit the subject of failure causes.

3.2 Analyses

The following analyses are based on the 246 SPRs that resulted in updates to code, database entries, or scenario scripts. The I&T tables and graphs are based on 216 I&T SPRs; the PDSS tables, 30 PDSS SPRs.

3.2.1 Areas of Change

Table 3.2.1-1 is a summary of the total number of SPRs that required code modifications in each of the major TRALAB functional areas. Any SPR that impacted more than one functional area has been counted once for each area. SPRs that only required modifications to database records or scenario script files are considered not to impact software and are not counted.

Table 3.2.1-1 SPRs Impacting Major Software Functional Areas	
FUNCTIONAL AREA	NUMBER OF SPRs
Hardware-Hiding	10
Behavior-Hiding	105
Software Decision-Hiding	87

Twenty nine (12%) of the SPRs required modifications that spanned two or more functional areas. Seventeen of these are I&T SPRs; twelve, PDSS SPRs. For both the I&T and PDSS SPRs, most are changes that rippled across both the Behavior-Hiding submodule Simulation Activity (SMA) and the Software-Decision submodule Simulation Delivery (SMD). The SMA module hides the behavior of the collection and processing systems that are to be trained, while the SMD module hides details of the simulation delivery mechanism. Unfortunately, changes to the behavior of target systems generally required modifications to the simulation delivery mechanism (SMD) as well as to the simulations of target system behavior (SMA). Fortunately, modifications to the SMD module frequently were minor and related more to the control mechanisms necessary to invoke the proper simulation activity function. A few of the 29 SPRs required modifications to the APIs of data-hiding modules in addition to modifications to data structures used in module implementations. Changes to a module's APIs typically require modifications to programs in other modules that use that API.

Table 3.2.1-2 is a summary of the total number of SPRs that required software modifications in each TRALAB third-level software module. Any SPR that impacted more than one module has been counted once for each impacted module.

There are no SPRs that required modifications spanning all three functional areas.

3.2.2 Change Confinement

The TRALAB module structure was built around the following list of areas of expected change, which were identified first in the System Technical Specification produced early in initial TRALAB development.

- terminal interface (e.g., blinking approach, bolding approach, window addressing scheme, keyboard drivers)
- underlying operating system
- networking environment (i.e., Ethernet, protocols to establish a circuit and to send and receive messages)
- simulation messages and displays (e.g., target-system menus, queries, and prompts)
- simulation timing
- simulation commands
- student evaluation criteria and reports
- student monitoring formats
- authoring exchange necessary to create/modify scenarios

Table 3.2.1-2 SPRs Impacting Third-Level Modules	
FUNCTIONAL AREA	NUMBER OF SPRs
HARDWARE-HIDING	
VIRTUAL OPERATING SYSTEM	3
VIRTUAL NETWORK	1
VIRTUAL TERMINAL	6
BEHAVIOR-HIDING	
SCENARIO MAINTENANCE	12
DATA BASE EDITOR	8
SYSTEM ADMINISTRATION	6
SIMULATION ACTIVITY	45
TRAINING EVALUATION	8
STUDENT OBSERVATION	7
MENU PROCESSING	8
CONTROL	11
SOFTWARE DECISION-HIDING	
SCENARIO TRANSLATION	12
SIMULATION DELIVERY	45
VIRTUAL DATABASE	3
SYSTEM DATA	6
TARGET DATA	9
SYSTEM ADMINISTRATION DATA	1
OPERATIONAL DATA	13

- specifications for key data structures (e.g., scenario data formats, target system display formats, audit trail formats)
- access policies for key data structures
- run-time environment (e.g., number of terminals, number of student errors before instructors are alerted, standard duration for maintaining student audit trails)
- language implementation
- additional authors, students, and instructors
- additional classroom management tools (e.g., student roster generation support, student performance trend reports)

The original TRALAB designers tried to minimize the potential ripple effect of carrying out such changes. Table 3.2.2-1 lists the proportion of SPRs that impacted 0, 1, 2, and 3 or more TRALAB "lowest-level" modules (i.e., third or fourth-level modules with no submodules). A large percentage (82%) of the TRALAB SPRs impacted zero or one lowest-level modules (an SPR resulting only in modifications to database records or scenario script files was considered to impact zero modules). Only 3% impacted three or more modules.

Tables 3.2.2-2 and 3.2.2-3 break down the data for Table 3.2.2-1 in terms of I&T SPRs and PDSS SPRs. No PDSS SPRs impacted zero modules. This is not surprising because by that time the target systems to be trained were complete and stable, which was not the case during initial TRALAB development. There is a much larger percentage of PDSS SPRs that impacted two modules, but none resulted in modifications that rippled across three or more modules.

Table 3.2.2-1 SPRs and Modules Requiring Modification

PERCENT OF SPRs	NUMBER OF MODULES IMPACTED
29	0
53	1
15	2
3	3+

Table 3.2.2-2 I&T SPRs and Modules Requiring Modification

PERCENT OF SPRs	NUMBER OF MODULES IMPACTED
33	0
53	1
11	2
3	3+

Table 3.2.2-3 PDSS SPRs and Modules Requiring Modification	
PERCENT OF SPRs	NUMBER OF MODULES IMPACTED
0 59 41 0	0 1 2 3+

Figure 3.2.2-1 presents the same data used for table 3.2.2-1 over time. After November 1988 during the I&T period, the proportions of SPRs impacting 0, 1, and two or more lowest-level modules remain relatively stable. After February 1991 during PDSS, there is modest overall growth in the percentage of SPRs for which modifications ripple across two or more modules.

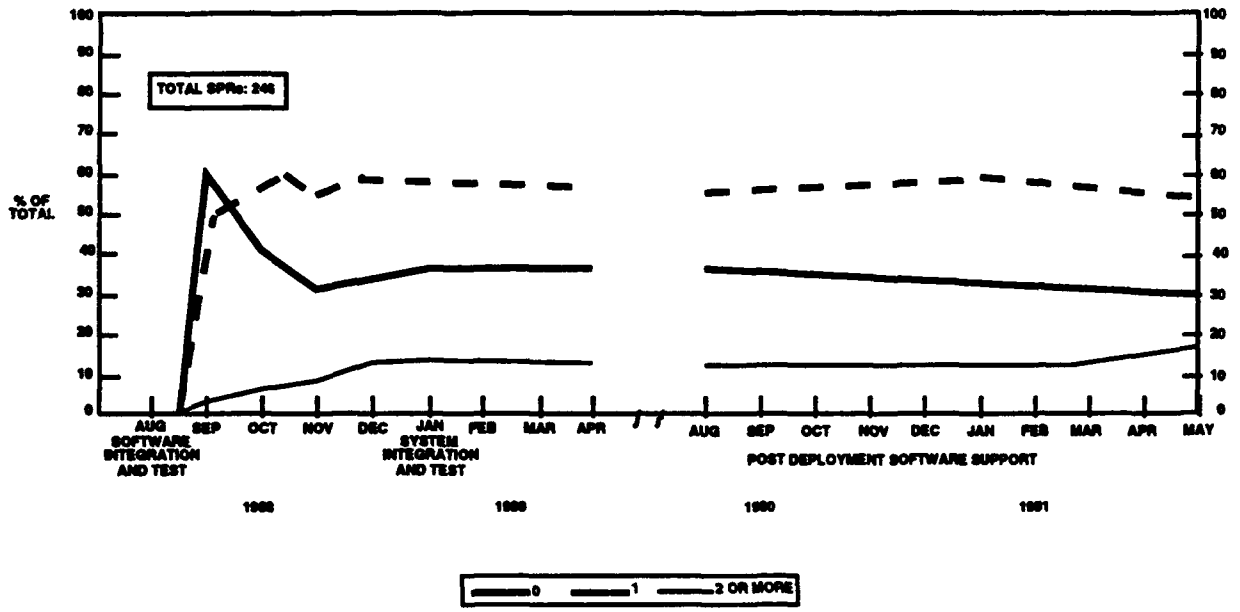

Figure 3.2.2-1 SPRs Categorized By Number of Modules Changed

3.2.3 Ease of Change

Table 3.2.3-1 shows the relationship between SPRs and ranges of required resolution hours. A large percentage (82%) of TRALAB problem reports required a staff day or less (i.e., 8 staff hours or less) to complete. Table 3.2.3-2 shows an even larger percentage (90%) of I&T problem reports were resolved in less than a staff day. Table 3.2.3-3 shows a much different situation for PDSS SPRs. PDSS SPRs seem to require more effort to resolve on the average; only 23% were resolved in a staff day or less. One possible reason for this is that several were utility enhancements, requiring several hundred NCLSLOC to complete. Examples of utility enhancements include utilities to simplify database generation and to verify database integrity following TRALAB upgrades.

Table 3.2.3-1 SPR resolution hours: all SPRs	
RANGE OF HOURS	CUMULATIVE PERCENT OF SPRs
[0, 1]	41
[0, 2]	56
[0, 3]	62
[0, 8]	82

Table 3.2.3-2 SPR resolution hours: I&T SPRs	
RANGE OF HOURS	CUMULATIVE PERCENT OF SPRs
[0, 1]	46
[0, 2]	63
[0, 3]	70
[0, 8]	90

Table 3.2.3-3 SPR resolution hours: PDSS SPRs	
RANGE OF HOURS	CUMULATIVE PERCENT OF SPRs
[0, 1]	0
[0, 2]	0
[0, 3]	6
[0, 8]	23

Tables 3.2.3-4, 3.2.3-5, and 3.2.3-6 present the same data restricted to SPRs that were related to expected types of changes (total is 219). Generally, a greater percentage of SPRs involving changes that were anticipated by initial TRALAB software designers were easier than unanticipated changes.

Table 3.2.3-4 SPR resolution hours: All expected change SPRs	
RANGE OF HOURS	CUMULATIVE PERCENT OF SPRs
[0, 1]	45
[0, 2]	61
[0, 3]	68
[0, 8]	90

Table 3.2.3-5 SPR resolution hours: I&T expected change SPRs	
RANGE OF HOURS	CUMULATIVE PERCENT OF SPRs
[0, 1]	49
[0, 2]	67
[0, 3]	74
[0, 8]	96

Table 3.2.3-6 SPR resolution hours: PDSS expected change SPRs	
RANGE OF HOURS	CUMULATIVE PERCENT OF SPRs
[0, 1]	0
[0, 2]	0
[0, 3]	6
[0, 8]	27

Figures 3.2.3-1 and 3.2.3-2 present SPR resolution effort over time. By December 1989 the summary patterns found in Tables 3.2.3-1 through Table 3.2.3-6 were fairly well established. During the January to May 1992 time frame the effects of the PDSS utility-oriented enhancements increased the percentage of SPRs requiring more than one day to complete.

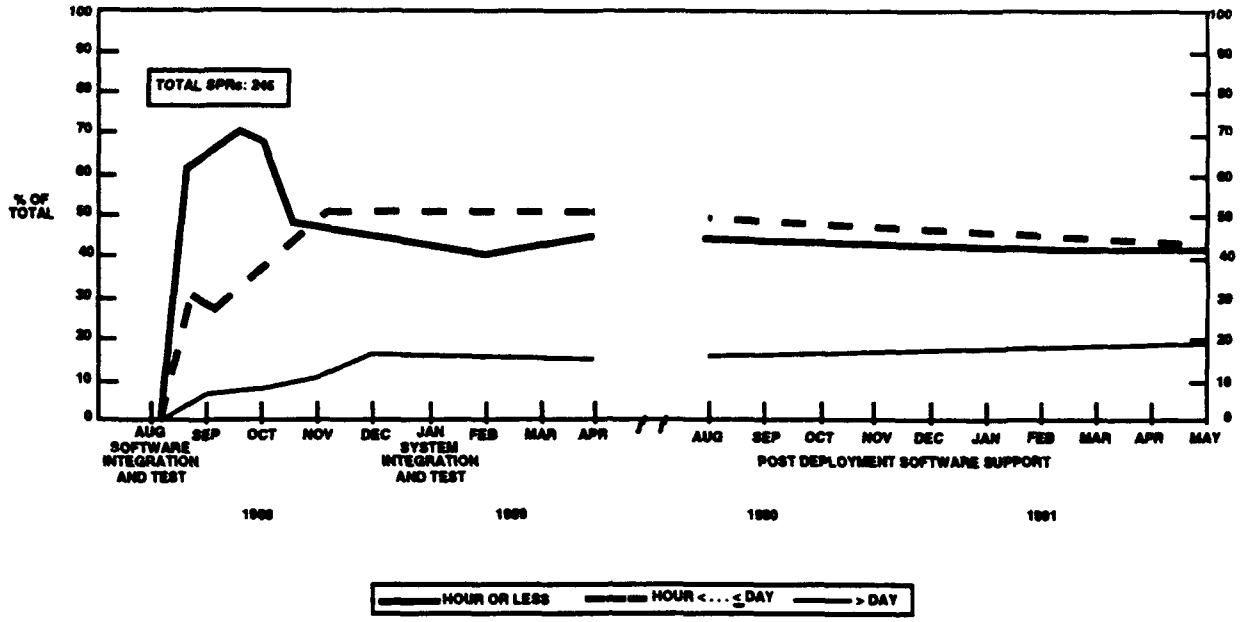

Figure 3.2.3-1 SPR Resolution Effort Categories

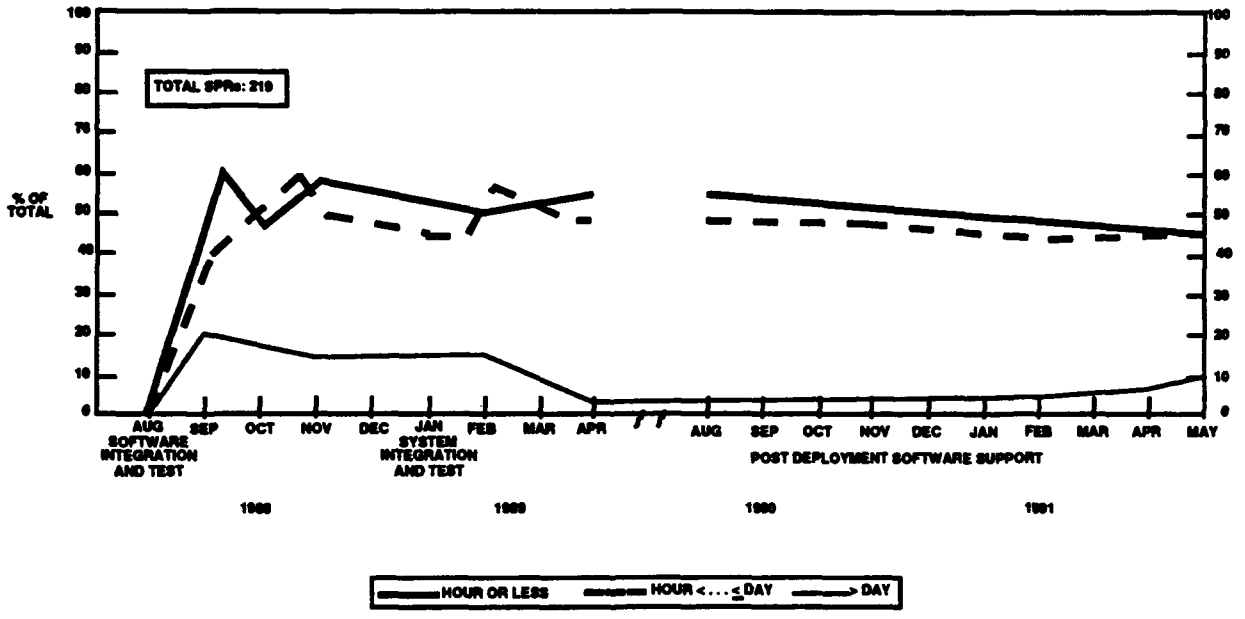

Figure 3.2.3-2 SPR Resolution Effort Categories: Expected Changes Only

3.2.4 Module Size and SPR Occurrences

During initial TRALAB development, contractor management and a good number of the development engineers were apprehensive over what appeared to be some rather large modules (see Table 2.3-1). The concern was that such large entities would be defect prone. Table 3.2.4-1 shows the number of SPRs per 1000 NCLSLOC for third-level modules in different size categories (while not all SPRs involve defects, the vast majority do). The data compares favorably

to typical industry data points [JONES 81] of 10-50 defects per 1000 NCLSLOC (average for American-produced code). And, as of the middle of 1991, there is no clear relationship between module size and error proneness. Card [85] and Conte [86] have observed similar results.

Table 3.2.4-1 Third-level modules and SPR rate: All SPRs

MODULE NCLSLOC COUNT	SPRs per 1000 NCLSLOC
[0, 999]	9.3
[1000, 1999]	3.8
[2000, 2999]	2.8
[3000, 3999]	8.3
[4000, 4999]	1.7
[5000 +]	3.7

We do not have an answer as to why in general larger modules do not seem more error prone. One reason may be that they were coded with more care and experienced more scrutiny during the design reviews because of their size. A more pessimistic explanation may be that, because test coverage tools were not used to support module-level testing, there may be numerous undetected defects within the larger modules due to the scope of the testing required to fully exercise all module paths. TRALAB module testing, however, was done with some care. For example, black-box testing software was written and used to stimulate the external interfaces to TRALAB third and fourth-level modules and compare actual with specified results.

We have a likely explanation for the small SPR/NCLSLOC value for TRALAB modules in the NCLSLOC size category [4000, 4999]. Actually, there is just one such module, the TRALAB Menu Processing (TMP) module. Services provided by TMP are used extensively in the implementations of other Behavior-Hiding modules; that is, TMP services are basic TRALAB services. To avoid troublesome delays during I&T typically caused by flawed basic-service software, TMP services were tested extensively at the module level using black-box techniques before the module was baselined for actual use.

Table 3.2.4-2 shows the average NCLSLOC count change resulting from SPR modifications for third-level modules in different size categories. As of the middle of 1991, there is no clear relationship between module size and the magnitude of SPR impact on NCLSLOC counts.

Table 3.2.4-2 Third-level modules and SPR NCLSLOC impacts: All SPRs	
MODULE NCLSLOC COUNT	MAGNITUDE OF NCLSLOC COUNT DELTAS / SPR
[0, 999]	11.3
[1000, 1999]	9.4
[2000, 2999]	6.0
[3000, 3999]	17.1
[4000, 4999]	7.0
[5000 +]	23.2

3.2.5 Cumulative Resolution Effort

Figure 3.2.5-1 shows the cumulative average hours to resolve SPRs. There are separate plots for all SPRs, all SPRs related to the expected changes, and all SPRs related to expected changes that could be handled as database or scenario updates (i.e., zero-module updates). The graphs tend to support that TRALAB engineers benefit by engineering for change, especially upon entry into PDSS. In particular, they seem to benefit measurably from encapsulating specific changes as database entries.

An important question is whether the differences seen in Figure 3.2.5-1 are significant statistically. The population yielded greatly unequal sample sizes for the groups in question. There are a very small number of unexpected changes (i.e., 27) in comparison to expected (219). The number of expected changes that are handled as database updates (70) is somewhat small in comparison to the remaining expected changes (149), and these two data sets exhibit some different characteristics (e.g., spread). Unfortunately, no statistical arguments can be made at this time.

3.2.6 Conclusions

The goals of the TRALAB program were to specify, design, and implement a system that would promote the ability to control the:

- ripple effect when implementing expected system changes, and

- effort required to implement expected system changes.

To achieve this, initial TRALAB design engineers identified expected changes early in the requirements specification process and factored them explicitly into the identification of TRALAB modules and the specification of abstract interfaces to these modules. Analysis of the TRALAB change data to date continues to indicate informally that such an engineering approach controls the ripple of change modifications and the effort required to implement changes. Additionally, applying information hiding concepts in the definition of the module structure without regard to ultimate module NCLSLOC sizes does not seem to negatively impact defect densities.

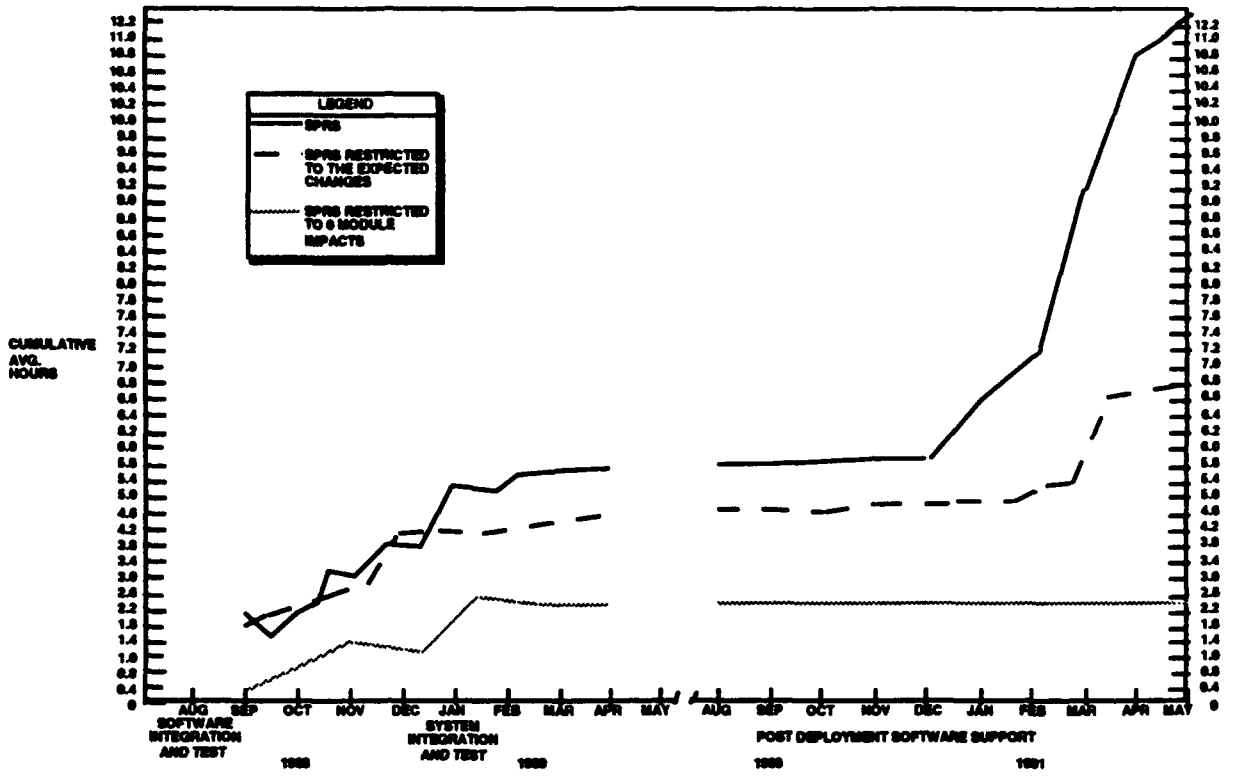

Figure 3.2.5-1 Cumulative Average Resolution Effort: By Close-Out Date

4.0 Acknowledgments

We are deeply indebted to Mr. Terry Fisher, who had the vision and determination to apply SCR technology to an actual development and who made this report possible. We would like to thank Mr. Keith Kramer for his analyses of the TRALAB data and for preventing us from reading to much into the data. This work was funded under Contract Number: N00014-90-C-2168.

5.0 References

[BAUMERT 92] Baumert, J. H., and McWhinney, M.S. "Software Measures and the Capability Maturity Model." Software Engineering Institute Technical Report, CMU/SEI-92-TR-25.

[BOEHM 73] Boehm, B.W. "Software and Its Impact: A Quantitative Assessment." *Datamation*, May 1973, pp. 48-59.

[CARD 85] Card, D.N., Page, G.T., and McGarry, F.E., "Criteria For Software Modularization." *Proceedings of the Eighth International Conference on Software Engineering*, 1985, pp 372-377.

[CHMURA 90] Chmura, L. J., Norcio, A. F., and Wicinski, T. J. "Evaluating Software Design Processes by Analyzing Change Data Over Time." *IEEE Transactions on Software Engineering*, Vol.16, No. 7 (July 1990), pp. 729-740.

[CLEMENTS 83] Clements, P.C., Parnas, D.L., and Weiss, D.M. "Enhancing Reusability with Information Hiding." *Proceedings, Workshop on Reusability in Programming* pp. 251-257, September 1983.

[CLEMENTS 86] Clements, P.C., and Parnas, D.L. "A Rational Design Process: How And Why To Fake It." *IEEE Transactions on Software Engineering*, Vol. SE-12, No. 2 (February 1986), pp. 251-257.

[CONTE 86] Conte, S.D., Dunsmore, H.E., and Shen, V.Y. *Software Engineering Metrics and Models.* Benjamin-Cummings, 1986.

[HAGER 88] Hager, J.A., and Chmura, L.J. Software Engineering Principles Study Report, Naval Research Laboratory Technical Report, April 1988.

[HAGER 89] Hager, J.A. "Software Cost Reduction Methods in Practice." *IEEE Transactions on Software Engineering*, Vol.15, No. 12 (December 1989), pp. 1638-1644.

[HAGER 91] Hager, J.A. "Software Cost Reduction Methods in Practice: A Post-Mortem Analysis." *Journal of Systems and Software*, Vol. 14, No. 2 (February 1991), pp. 67-77.

[HENINGER 80] Heninger, K.L. "Specifying Software Requirements for Complex Systems: New Techniques and their Application." *IEEE Transactions on Software Engineering*, Vol SE-6, No. 1 (January 1980), pp 2-13.

[HESTER 81] Hester, S.D., Parnas, D.L., and Utter, D.F. "Using Documentation as a Software Design Medium." *The Bell System Technical Journal*, Vol. 60, No. 8 (October 1981), pp. 1941-1977.

[JONES 81] Jones, T. C. "Program Quality and Programmer Productivity: A Survey of the State of the Art." *ACM Lectures*, November 1981.

[LAMB 88] Lamb, D.A. *Software Engineering: Planning For Change* Prentice Hall, 1988.

[PARNAS 72] Parnas, D.L. "On the Criteria for Decomposing Systems into Modules." *Communications of the ACM*, Vol. 15, No. 12 (December 1972) pp. 1053-1058.

[PARNAS 84] Parnas, D.L., Clements, P.C., and Weiss, D.M. "The Modular Structure of Complex Systems." *Proceedings of the 7th International Conference on Software Engineering*, pp. 408-417, March 1984.

[WALLACE 87] Wallace, R.H., Stockenberg, J.E., and Charette, R.N. *A Unified Methodology For Developing Systems.* McGraw Hill, 1987.

Appendix A. TRALAB Data Collection Forms

SPR NO.:

SOFTWARE PROBLEM REPORT

I&T NO.:

SHEET ___ OF ___

ORIGINATOR/ORG. _____ EXT.:_____

DATE:_____

PROBLEM AREA: SOFTWARE_____ HARDWARE_____
DOCUMENTATION_____

INTERFACE_____ OTHER_____

PROBLEM WITH: ROUTINE_____ DATABASE_____ DOCUMENT_____

SCENARIO_____ TEST FILE_____ OTHER_____

PROBLEM IDENTIFIED DURING: ANALYSIS_____ DESIGN_____ OTHER_____

SOFTWARE TEST_____ SYSTEM TEST_____

IF TEST: THREAD/CATEGORY_____ CASE/SEGMENT_____

BRIEF DESCRIPTION:

DETAILED DESCRIPTION:

PROBABLE CAUSE:

IMPACT:

RECOMMENDATIONS/REMARKS:

MODULES REQUIRED FOR FIX:

DOCUMENT UPDATES REQUIRED:

SETD APPROVAL: _____ DATE: _____

**CM USE ONLY **

DATE LOGGED:_____ PRIORITY:_____

ACTION ASSIGNEE:_____ CMO:_____

CCB ACTION:_____ DATE:_____

RESOLUTION DATE:_____ SMT NO.____ DOCS.UPDATED_____

Figure A-1. Software Problem Report

SMT NO.:_____
DATE:_____
SHEET ___ OF ___

SOFTWARE MODIFICATION
TRANSMITTAL SPR NO.:_____

I&T NO.:_____

ORIGINATOR/ORG.:_____ EXT.:_____ DATE:_____

MODULE NAME:_____ BASELINE ID:_____

CLASSIFICATION OF FILES:_____
TYPE OF CHANGE SOFTWARE _____ SCENARIO_____ DATABASE_____
DOCUMENTATION_____OTHER_____

SOFTWARE AFFECTED:_____

DIRECTORY/LOCATION:_____

SOURCE FILE NAME/BRIEF DESCRIPTION OF CHANGE | CHANGED/ADDED DELETED | NEW/LAST REVISION

DOCUMENTATION AFFECTED:
DOCUMENT ACCESSION #S | DOCUMENT TITLE/VOLUME/DATE

FAILURE CAUSE TOTAL

CORRECTION TIME LABOR GRADE
09-ENG
10-AE
12-SE
14-PE

___(1) REQUIREMENT CHANGE
___(2) MISINTERPRETATION OF REQUIREMENT
___(3) CHANGE TO TARGET SYSTEM
___(4) MISINTERPRETATION OF DOCUMENTATION
___(5) IMPROPER DESIGN
___(6) IMPROPER IMPLEMENTATION OF DESIGN
CHANGED __
___(7) TIMING/SIZING
CHANGED__
___(8) DATABASE ERROR
___(9) INTERFACE ERROR
__(10) OTHER_____

TOTAL NUMBER OF:
MODULES

LOCATION

Figure A-2. Software Modification Transmittal

SMT NO.:_____
DATE:_____
SHEET ___ OF ___

SOFTWARE MODIFICATION
TRANSMITTAL SPR NO.:_____

I&T NO.:_____

ADDITIONAL COMMENTS:

FOLLOWING ITEMS UPDATED/DELIVERED:

_____ SOURCE	_____ MDF/TDF
_____ STS	_____ DATABASE
_____ MIMPS	_____ TEST PROCEDURE
_____ MINTS	_____ STREAMING TAPE
_____ USER GUIDE	_____ OTHER:_____

** CM/SETD USE ONLY **

CCB APPROVAL: _____ DATE:_____

CCB APPROVAL/RECORDED: _____ DATE:_____

STATUS OF SOFTWARE CHANGES: OPEN:_____
CLOSED_____

STATUS OF DOCUMENTATION CHANGES: OPEN:_____
CLOSED_____

INTEGRATION AND TEST: ACCEPTED: _____YES _____NO

I&T COMMENTS:

TEST BASELINE:_____
SETD APPROVAL:_____
DATE:_____
MASTER BASELINE UPDATED BY:_____
DATE"_____
COMMENTS:

Figure A-2. Software Modification Transmittal (Concluded)