RECENT SEL EXPERIMENTS AND STUDIES

Rose Pajerski
Donald Smith

SOFTWARE ENGINEERING BRANCH
Code 552
NASA/Goddard Space Flight Center
Greenbelt, Maryland 20771

INTRODUCTION

Since 1976, the Software Engineering Laboratory (SEL) has been dedicated to understanding and improving the way in which its organization, the Flight Dynamics Division (FDD) of NASA/Goddard Space Flight Center, develops, maintains, and manages complex flight dynamics software systems. During the past 17 years, the SEL has collected and archived data from over 100 software development projects in the organization. From these data, the SEL has derived models of the development process and product, and conducted studies on the impact of new technologies.

One of the SEL's overall goals is to treat each development effort as a SEL experiment that examines a specific technology or builds a model of interest. The SEL has undertaken many technology studies while developing operational support systems for numerous NASA spacecraft missions. Viewgraph 2 is a rough mapping of spacecraft missions and support software systems (on the top) to SEL studies (on the bottom) over the past 17 years. Measures and experiences from these development projects have been saved and used to understand, characterize, and improve the development environment.

The SEL's basic approach to software process improvement is first to understand and characterize the process and product as they exist to establish a local baseline. Only then can new technologies be introduced and assessed (phase two) with regard to both process changes and product impacts. The third phase synthesizes the results of the first two phases into various packages such as process and product models, training materials, and tools and guidebooks. These products are then fed back into the development environment for subsequent projects to use and benefit from. Viewgraph 4 illustrates the SEL three-phase process improvement model.

The SEL organization consists of three functional areas: software project personnel, database support personnel, and software engineering analysts. The largest part of the SEL is the 250-plus software personnel who are responsible for the development and maintenance of over 4 million source lines of code (SLOC) that provide orbit and attitude ground support for all GSFC missions. Since the SEL was founded, software project personnel have provided software measurement data on over 100 projects. These data have been collected by the database support personnel to be stored in the SEL database for use by both the software project personnel and the software engineering analysts. The database support staff enter measurement data into the SEL database, quality assure the data, and maintain the database and its reports. The software engineering analysts define the experiments and studies, analyze the data, and produce reports which now number over 300. These reports affect such things as project standards, development procedures, and project management guidelines.

The SEL has been a fairly stable organization; however one significant change has occurred

recently. About a year ago, the FDD reorganized and the SEL became responsible for the maintenance of all operational software in the environment. Along with consolidating development and maintenance into one organization, all acceptance testing was added. Now one organization, the SEL, is responsible for a significant portion of the development life cycle, from requirements analysis through maintenance.

The SEL's new areas of responsibility, maintenance and testing, have become focal points for current studies which correspond to the three phases of the SEL improvement model: (1) understanding maintenance processes and products, (2) assessing the effectiveness of different testing approaches, and (3) analyzing our locally derived cost and schedule estimation models to determine if they need updating. These studies will be discussed in reverse sequence, however, going from the most complete data to the most preliminary findings.

SEL COST AND SCHEDULE ESTIMATION MODELS

Over the past several years, organizational and technology changes have occurred in the FDD software development environment that may have affected many of the SEL's baseline models that characterize some of the products and processes of the FDD. The purpose of the cost and schedule estimation study[1] was to update the SEL cost and schedule estimation models. Thirty-nine FDD development projects measured over 15 years were examined. Specific factors which may have an impact on cost and schedule were analyzed. These factors included code reuse, language, application type, and subjective data such as team experience level and technology usage.

Cost Estimation Factors

The SEL baseline cost model is based on software size, productivity, and a weighting term, the growth factor:

$$\text{Effort} = \text{size} / \text{productivity} \times \text{growth factor}$$

In this environment, managers compare current system requirements with historical experience on earlier systems to estimate initial software size in terms of both new SLOC and reused SLOC. Productivity values are also based on data from previous representative development efforts. While size and productivity are commonly used terms, the definition and usage of the growth factor in the above expression is unique to our environment. Previous experience had shown that the size of a software system will grow by 40% from the time the requirements are baselined at the Software Requirements Review to when the system becomes operational. This growth is due to early uncertainties in operational support scenarios as well as changes in the spacecraft hardware which result in software requirements changes.

The current study results (viewgraph 7) indicate that the basic effort model is still valid but that accuracy can be improved by including new weighting factors based on language and reuse levels. The language selected, FORTRAN or Ada, has an impact on both the cost of reusing code and the productivity values to be applied. For FORTRAN reuse, a 20% cost multiplier is used while the Ada reuse multiplier is 30%. The productivity values for FORTRAN and Ada are 3.5 and 5.0, respectively. The software growth factor is also affected by the level of reuse. The 40% value still holds for low-reuse systems but decreases by 15% for high-reuse systems.

Schedule Estimation Factors

The study concluded that the only quantifiable factor affecting schedule estimates was the application type. There are two application types in the FDD environment, operational ground systems and simulators. The simulators, which usually begin development with significant requirements undefined, take about 35% longer to develop than the operational ground systems. Neither the cost nor schedule models contain any subjective factors, although the study did look at the effects of subjective data on these models. (Jon Valett's report in Session 3 discusses the results of that part of the study.)

SEL TESTING STUDIES

Over the years, several testing methods have been used in the FDD environment. The goal of the ongoing SEL testing study is to determine the relative effectiveness of each method by examining key product and process measures such as effort and error rates. The testing approaches include:

- SEL Standard Test Process
- Independent Verification and Validation (IV&V)
- SEL Cleanroom
- Independent Test Team

The organizational boundaries and key process elements of each approach are described below and are tabulated in viewgraph 10.

- The SEL standard test process involves two separate organizations, the development group and an acceptance test (AT) group. The developers implement and system test the software by integrating and verifying the end-to-end system flow. Then the final build of the software is turned over to a separate AT group that performs functional acceptance testing based on the requirements.
- The IV&V approach added an independent test group to the standard process. That group worked in parallel with the development and AT groups to completely test the software as well as to verify the requirements and operational scenarios for the system.
- The SEL Cleanroom methodology has separate development and test teams that develop and test, respectively, in builds. The test team generates the test cases using a statistical method based on the frequency of activities of various operational scenarios. The final build of the software is then passed to a different organization for acceptance testing.
- The most recent approach to be applied uses an independent test team and is a direct consequence of the organizational changes within the FDD. This approach has separate development and test teams under the same organization. The test team handles all the build testing of the software using functional test case selection.

The IV&V approach will be discussed briefly because of its relevance to NASA programs today. The other three approaches will be contrasted and compared with one another in terms of some key process and product measures. The process measure that will be used is *effort distribution by activity* and the product measure is the *error rate* through development and testing.

IV&V Test Study

A recent National Research Council (NRC) report[2] recommends that IV&V be part of the standard NASA testing process. This recommendation may be appropriate for the Space Shuttle software that was studied in the NRC report; however it is certainly not appropriate for all NASA software. Results of SEL IV&V experiments[3] conducted in the early 1980s in the FDD environment were not positive. Although one of the expectations for IV&V within the environment had been increased software reliability, the study found that the error rates were not favorably impacted and that total development cost of the software increased significantly (between 30% and 60%). Consequently, it was determined that IV&V was not appropriate for adoption in the FDD.

Process Measures Comparison

Using effort distributions to compare test approaches is an effective way to identify process differences. The point of interest is to see if there is any apparent impact or observable change when compared to the SEL standard process. Viewgraph 12 shows the distribution of effort involved in design and code versus test for the three test methods. There is little difference between the SEL Standard and Cleanroom effort distributions; however, the independent test team approach does display a very different effort distribution. Simply using process measures will not determine which of these approaches is "best" but it does highlight process differences.

Product Measures Comparison

Examining the average error rates as recorded from the design phase through the acceptance test phase is a good way to determine the effectiveness of testing methods. The example in viewgraph 13 shows two ways of viewing error rate data on the same set of project data.

The chart on the left is the average error detection rates grouped by testing approach. Two points of interest are noted:

1) Cleanroom stands out as having a higher error rate than the other two methods. However, this is somewhat misleading because the Cleanroom process includes different types of errors in the error statistics. Previous studies[4] have shown that the Cleanroom process actually produces error rates that are lower than the FDD baseline for certain classes of projects.
2) The independent test team approach shows lower error rates, indicating that this approach shows some promise and deserves further study.

The chart on the right groups the data by low- and high-reuse projects. There is indication that factors such as the level of reuse may affect error rates.

The testing effectiveness study is not yet complete because other process and product measures need to be assessed, including an evaluation of data collected during the maintenance phase.

SEL MAINTENANCE STUDIES

The first area of focus by the SEL in the maintenance arena is to build a baseline understanding of maintenance products and processes such as software characteristics, effort distributions, and change and error profiles. Understanding these elements will enable cost and schedule estimation models to be built, which is one of the future goals of studying maintenance. The number of systems and the size of the software being maintained varies: 105 systems ranging in size from 10,000 SLOC to 250,000 SLOC totaling 3.5 M lines of code. A high percentage of these systems are FORTRAN mainframe systems so they are the first ones to be analyzed in the study. Information learned so far is based on data from a handful of these systems, so the maintenance baseline presented (in viewgraphs 14-16) is considered a preliminary characterization of the maintenance process and products.

FDD is currently maintaining two types of systems: multimission systems, which support many spacecraft and have a software lifetime of 10-30 years, and single-mission support systems, which run as long as the spacecraft is operational (typically 2-7 years).

Error and Change Characteristics

Preliminary studies reveal that although the software sizes are similar, these two types of systems show very different error characteristics after 5 years of operations (viewgraph 15). For multimission systems, the error rate is an order of magnitude higher than the rate for single-mission systems. There are many possible explanations for this. The multimission systems are used more and they are also updated with more frequent enhancements. If the difference in error rates shown here is confirmed by further analysis, the reasons for it will need to be examined and evaluated.

Also shown in the viewgraph are two pie charts depicting two types of change distributions:

- Change type (right chart) as determined by the requestor on the change request forms: about 25% of the changes are enhancements and 75% are error corrections with less than 1% being adaptations (changes due to operating system or compiler upgrades).
- Effort distribution (left chart) as determined by the maintainers in satisfying the change requests: about 66% of the maintainers' time is spent on implementing enhancements and the remainder is spent correcting errors and adapting software.

Effort Characteristics

Cost (effort) is another important element for understanding software maintenance. Viewgraph 16 displays several cost characteristics, again divided into multimission and single-mission systems.

The cost to initially develop these systems is about the same for both types of system; however, the cost of maintenance varies. Maintenance costs on the multimission systems are running about 3 staff years per year, or about 10% of the development cost per year. (The 10% figure has been used as a rule of thumb in the FDD for many years in calculating maintenance costs, so it is interesting to see it confirmed with some recent data). The yearly maintenance cost for single-mission systems is currently running at about 2% of development cost, which is probably due to these systems being enhanced less often.

Another way of understanding an unfamiliar process such as maintenance is to compare it with an established process that has been baselined. The pie charts on viewgraph 15 compare the maintenance and development processes in terms of effort distribution. The biggest difference is the relative effort spent in testing activities: 30% for development and 5% for maintenance. This difference certainly must be probed further in defining a baseline understanding of maintenance processes and products in the FDD environment.

SUMMARY

The studies discussed in this paper are all examples of activities that are performed as part of the SEL's process improvement model. Using this model, the SEL starts by understanding the product and process, then assesses the impact of new technologies, and finally packages what was learned. The preliminary examination of maintenance effort, error, and change profiles to establish a maintenance baseline exemplifies understanding-phase activities. The ongoing testing study that is examining the effects of various testing approaches on process and product measures is an example of typical assessing efforts. Finally, the derivation of cost and schedule estimation models from locally driven factors such as reuse level, application type, and language is an example of experience packaging.

In the SEL, no study is ever really completed. Studies will be repeated and iterated upon in the future as part of the ongoing software improvement process.

REFERENCES

1. Condon, S., et al., *Cost and Schedule Estimation Study Report*, SEL-93-002, November 1993

2. *An Assessment of Space Shuttle Flight Software Development Process*, National Research Council, 1993

3. Page J., "Methodology Evaluation: Effects of Independent Verification and Integration On One Class of Application," *Proceedings of the Sixth Annual Software Engineering Workshop*, SEL-81-013, December 1981

4. Green, S. E., and R. Pajerski, Cleanroom Process Evolution in the SEL," *Proceedings of the Sixteenth Annual Software Engineering Workshop*, SEL-91-006, December 1991

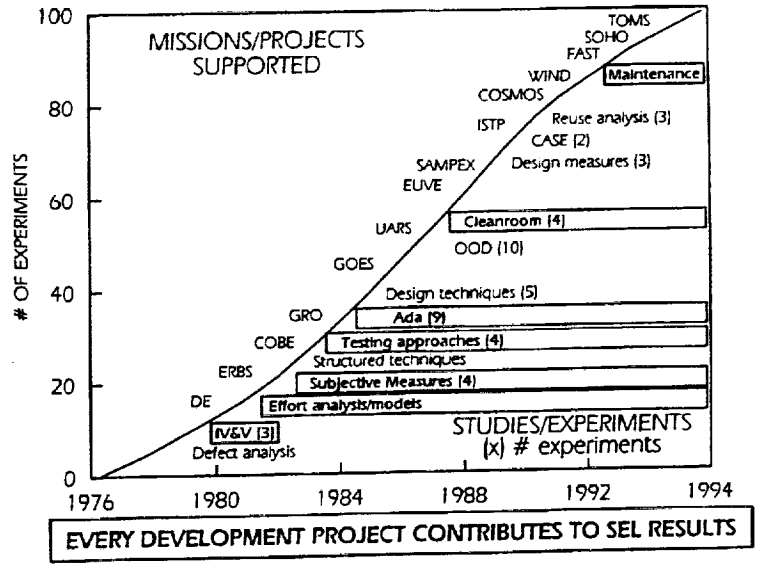
RECENT SEL EXPERIMENTS AND STUDIES

18TH ANNUAL
SOFTWARE ENGINEERING
WORKSHOP

ROSE PAJERSKI
DONALD SMITH
NASA/GSFC

B649.001

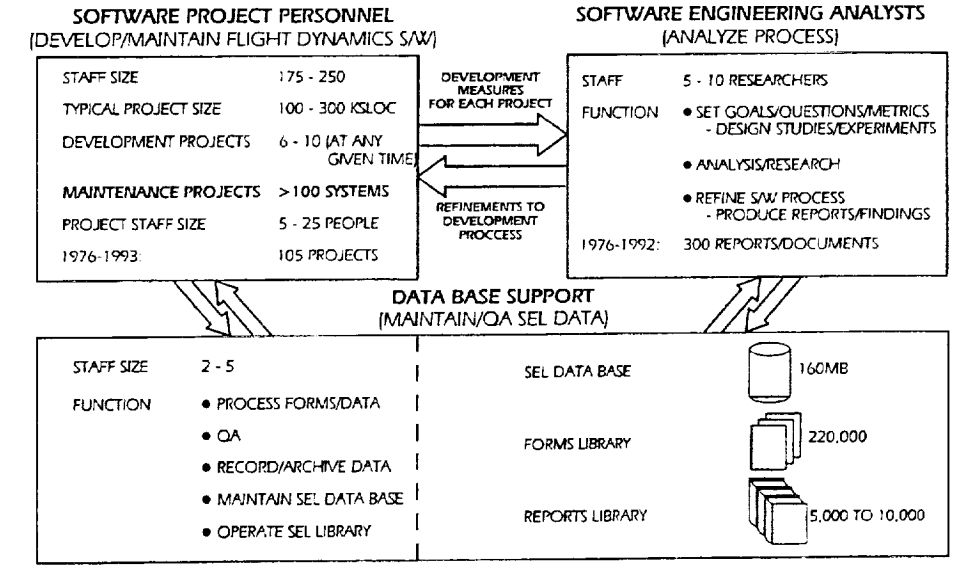
STUDIES ON OVER 100 PRODUCTION PROJECTS



EVERY DEVELOPMENT PROJECT CONTRIBUTES TO SEL RESULTS

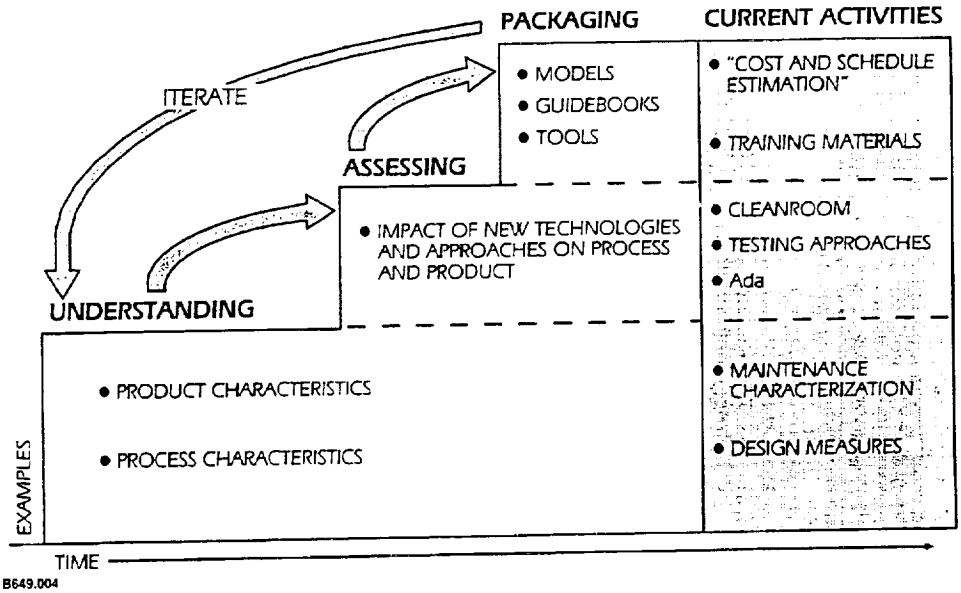
B649.002

SEL ORGANIZATIONAL STRUCTURE



B649.003

SEL IMPROVEMENT MODEL - CURRENT ACTIVITIES



B649.004

SUMMARY OF RECENT SEL STUDIES

PLAN	• COST AND SCHEDULE ESTIMATION STUDY ①
	• SUBJECTIVE FACTORS EVALUATION • NASA DOMAIN PROFILES • TECHNOLOGY TRANSFER
TRAINING	• RECOMMENDED APPROACH TO SOFTWARE DEVELOPMENT • GOVERNMENT/CONTRACTOR TASK MANAGEMENT • FLIGHT DYNAMICS APPLICATIONS
IMPLEMENT	• DESIGN MEASURES • CASE • Ada • OOD
TEST	• COMPARISON OF APPROCHES - INDEPENDENT VERIFICATION AND VALIDATION SEL STANDARD ② CLEANROOM INDEPENDENT TEST TEAM
MAINTENANCE	• PROCESS UNDERSTANDING • PRODUCT CHARACTERIZATION ③

B649.005

① COST AND SCHEDULE STUDY*

GOAL	UPDATE SEL BASELINE	EFFORT ESTIMATION MODEL SCHEDULE ESTIMATION MODEL
	DETERMINE IMPACTS OF	REUSE (CODE) LANGUAGE APPLICATION TYPE SUBJECTIVE FACTORS (EXPERIENCE, TECHNOLOGY)

STUDY PARAMETERS
- 39 PROJECTS (1977-1992)
- 2 LANGUAGES (FORTRAN, Ada)
- 2 APPLICATION TYPES
- 20K-300K SOURCE LINES OF CODE

* "SOFTWARE ENGINEERING LABORATORY COST AND SCHEDULE ESTIMATION STUDY REPORT," S. CONDON, M. REGARDIE, S.WALIGORA, SEPTEMBER 1993

B649.006

① WHAT IMPACTS COST?

EFFORT = SIZE/PRODUCTIVITY x GROWTH FACTOR

$$\text{EFFORT (Ada)} = \frac{\text{(NEW SLOC + 30\% REUSED SLOC)}}{\text{PRODUCTIVITY (Ada)}} \times \begin{pmatrix} 1.4 \text{ (REUSE } < 70\%) \\ 1.2 \text{ (REUSE } \geq 70\%) \end{pmatrix}$$

$$\text{EFFORT (FORTRAN)} = \frac{\text{(NEW SLOC + 20\% REUSED SLOC)}}{\text{PRODUCTIVITY (FORTRAN)}} \times \begin{pmatrix} 1.4 \text{ (REUSE } < 70\%) \\ 1.2 \text{ (REUSE } \geq 70\%) \end{pmatrix}$$

COSTS 50% MORE TO REUSE A LINE OF Ada CODE THAN A LINE OF FORTRAN

SOFTWARE SIZE GROWTH IS 15% LOWER FOR HIGH REUSE SYSTEMS

B649.007

① WHAT IMPACTS SCHEDULE?

$$\text{SCHEDULE} = \text{COEFF} \times (\text{EFFORT})^{0.3}$$

$$\text{SCHEDULE (GROUND SYSTEMS)} = 5.0 \times (\text{EFFORT})^{0.3}$$

$$\text{SCHEDULE (SIMULATORS)} = 6.7 \times (\text{EFFORT})^{0.3}$$

SCHEDULE IMPACTED BY APPLICATION TYPE, NOT BY LANGUAGE OR REUSE LEVEL

B649.008

② EXPERIMENTS IN TESTING

GOAL

- ASSESS THE IMPACT OF ORGANIZATIONAL CHANGES ON SEL PROCESSES

- COMPARE TEST APPROACHES
  - INDEPENDENT VERIFICATION AND VALIDATION — 1982 STUDY
  - SEL STANDARD TEST PROCESS — 1978 - CURRENT
  - SEL CLEANROOM — 1986 - CURRENT
  - INDEPENDENT TEST TEAM — 1992 - CURRENT

- ASSESS EFFORT AND ERROR DISTRIBUTIONS TO DETERMINE TESTING EFFECTIVENESS

B649.009

② OVERVIEW OF FOUR TEST APPROACHES

SEL STANDARD PROCESS (24 PROJECTS)

REQUIREMENTS	DEVELOPMENT	SYSTEM TEST	ACCEPTANCE TEST
	CODE READING	END-TO-END FLOW	FUNCTIONAL, REQUIREMENTS BASED

IV & V (3 PROJECTS)

REQUIREMENTS	DEVELOPMENT	SYSTEM TEST	ACCEPTANCE TEST
	IV & V TEST TEAM		

SEL CLEANROOM (4 PROJECTS)

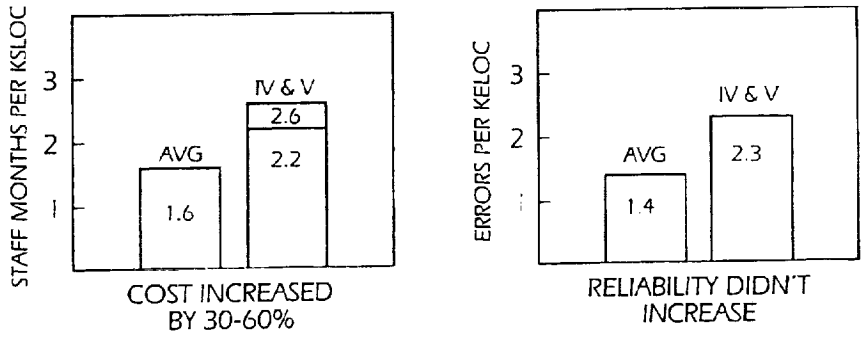
REQUIREMENTS	DEVELOPMENT/TEST	ACCEPTANCE TEST
	STATISTICAL CASES, CODE READING	FUNCTIONAL, REQUIREMENTS BASED

INDEPENDENT TEST TEAM (3 PROJECTS)

REQUIREMENTS	DEVELOPMENT/TEST	ACCEPTANCE TEST
	FUNCTIONAL CASES BY BUILD REQUIREMENTS BASED	

B649.010

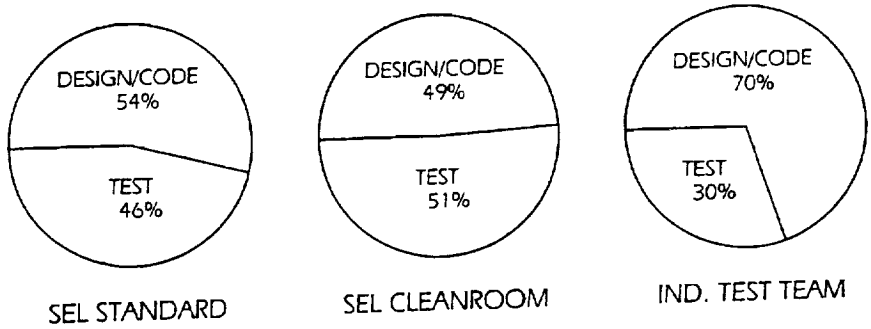
② IV & V TEST APPROACH



DOMAIN AND PRODUCT GOALS ARE KEY DRIVERS FOR PROCESS CHANGES

B649.011

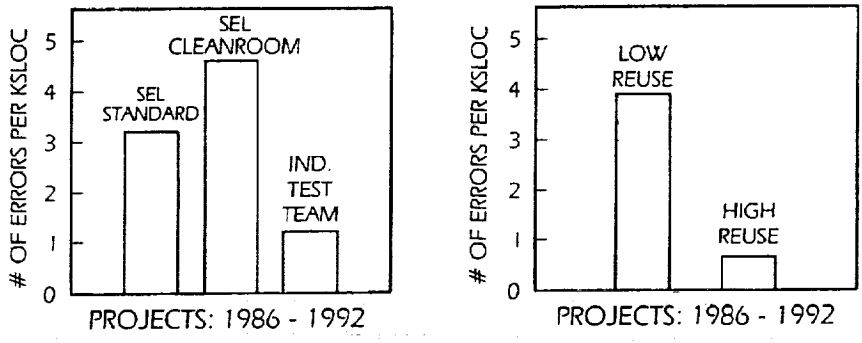
② TEST EFFORT DISTRIBUTION (BY ACTIVITY)



INDEPENDENT TEST TEAM APPROACH IMPACTING TEST PROCESS

B649.012

② ERROR DETECTION RATES - TWO VIEWS
(DESIGN THROUGH ACCEPTANCE TEST)



TEST APPROACH AND REUSE LEVEL
IMPACT ERROR RATES

B649.013

③ MAINTENANCE STUDY

CURRENT FOCUS	BUILD BASELINE UNDERSTANDING	SOFTWARE CHARACTERISTICS EFFORT DISTRIBUTIONS ERROR/CHANGE PROFILES ESTIMATION MODELS

OPERATIONAL SYSTEMS UNDER MAINTENANCE*

105 SYSTEMS	RANGE FROM 10 KSLOC TO 250 KSLOC TOTALS 3.5 MILLION SLOC		
LANGUAGES	85% FORTRAN	10% Ada	5% OTHER
PROCESSORS	80% MAINFRAME	5% OTHER	10% PC/WKSTN

MAINTENANCE ACTIVITY PRIMARILY SUPPORTS
FORTRAN MAINFRAME OPERATIONAL ENVIRONMENT

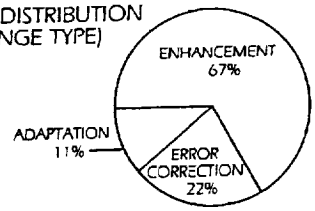
* MAINTENANCE INCLUDES ALL ACTIVITY AFTER OPERATIONAL START

B649.014

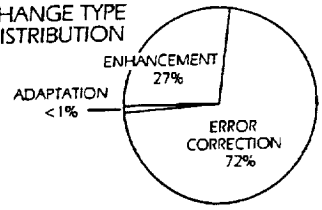
③ MAINTENANCE CHANGE/ERROR PROFILES

SYSTEM TYPE (SIZE IN KSLOC)	ERRORS DETECTED (5 YEARS OF OPERATIONS)	# DAILY USES
MULTI-MISSION (~200 KSLOC)	1.5 ERRORS/KSLOC	20-40
SINGLE MISSION (~150 KSLOC)	0.1 ERRORS/KSLOC	1-5

EFFORT DISTRIBUTION (BY CHANGE TYPE)



CHANGE TYPE DISTRIBUTION



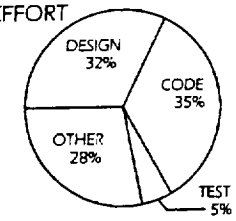
CHANGE EFFORT DISTRIBUTION NOT PROPORTIONAL TO NUMBER OF CHANGES

B649.016

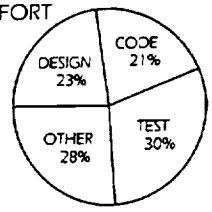
③ MAINTENANCE EFFORT

SYSTEM TYPE (SIZE IN KSLOC)	COST TO DEVELOP (STAFF YEARS(SY))	COST TO MAINTAIN (SY/YEAR)
MULTI-MISSION (~200 KSLOC)	30-40	3.0
SINGLE MISSION (~150 KSLOC)	25-35	0.5

MAINTENANCE EFFORT DISTRIBUTION



DEVELOPMENT EFFORT DISTRIBUTION



MAINTENANCE COST SIGNIFICANTLY DRIVEN BY SYSTEM TYPE

B649.015

SEL IMPROVEMENT -- AN ONGOING PROCESS