CORE
Metadata, citation and similar papers at core.ac.uk
Provided by NASA Technical Reports Server

N92-19422

IMPACT OF A PROCESS IMPROVEMENT PROGRAM IN A PRODUCTION SOFTWARE ENVIRONMENT: ARE WE ANY BETTER?

Gerard H. Heller
Gerald T. Page

COMPUTER SCIENCES CORPORATION
GreenTec II
10110 Aerospace Road
Lanham-Seabrook, MD 20706
(301) 794-4460

ABSTRACT

For the past 15 years, Computer Sciences Corporation (CSC) has participated in a process improvement program as a member of the Software Engineering Laboratory (SEL), which is sponsored by the National Aeronautics and Space Administration (NASA)/Goddard Space Flight Center (GSFC). This paper analyzes the benefits CSC has derived from involvement in this program. In the environment studied, it shows that improvements were indeed achieved, as evidenced by a decrease in error rates and costs over a period in which both the size and the complexity of the developed systems increased substantially. The paper also discusses the principles and mechanics of the process improvement program, the lessons CSC has learned, and how CSC has capitalized on these lessons.

INTRODUCTION

Computer Sciences Corporation (CSC) had some compelling motivations to join with the National Aeronautics and Space Administration (NASA)/Goddard Space Flight Center (GSFC) and the University of Maryland 15 years ago to form the Software Engineering Laboratory (SEL). In the context of 1976 and our partnership with GSFC, we wanted to study our overall flight dynamics software development process closely enough to be able to refine and improve it. Even then, we knew we had to be able to accurately describe and measure that process before real improvements could be made. Slowly and steadily, we embarked on a conscious process improvement program that would help us produce the larger and more complex flight dynamics ground systems required to support the more sophisticated spacecraft being built.

We wanted to build these complex systems with more reliability and greater economy. Our personnel were already committed to building quality systems; what we needed now was to build quality systems more productively. We also needed to expand the skills of our current personnel and to attract and retain new personnel who would enjoy the twin challenges of doing flight dynamics work and simultaneously trying to improve the methods used to do that work.

As competition to provide flight dynamics services increased both here and abroad, CSC became more ambitious in efforts to improve its processes and products and more committed to allocating the resources needed to make these improvements. We wanted to validate our belief that higher quality at lower costs was not a contradiction. We wanted to show that, in fact, those traits go hand in hand and that high-quality software really does cost less.

Sound business practices showed a need to move forward, not only to improve on our current work but to seek new opportunities as well. One way to enter these new business areas was to objectively demonstrate superior products and performance in our work with GSFC. Another way was to pursue and achieve formal recognition by other members of our industry. Our motivations for the SEL partnership were clear and compelling. From our participation in the SEL, we expected to capture specific gains, to learn some vital lessons, and to demonstrate, over time, that we were truly "getting better" at doing flight dynamics work.

Have we achieved these goals after 15 years of participation in the SEL? The rest of this paper answers this question. It describes the principles and mechanics of the SEL process improvement program, including examples of the program in action; examines what we have learned from our role in the program and how we have capitalized on that learning; and analyzes trends over the past 15 years to determine quantitatively whether or not we have met our objectives.

SEL BACKGROUND

The SEL

The SEL (Reference 1) is a research project sponsored by NASA/GSFC and supported by the Computer Science Department at the University of Maryland and by CSC. The SEL's mission is to understand and improve the overall software development process. To do this, the SEL conducts experiments with production software projects, measures the effect of the techniques applied, and then adopts the most beneficial methodologies for future projects.

The SEL Environment

The production software environment studied by the SEL is an environment of similar flight dynamics applications developed by GSFC for such spacecraft problems as attitude and orbit determination and control, mission planning, and maneuver control. These applications are largely scientific and mathematical, with moderate reliability requirements and severe development time constraints imposed by a fixed spacecraft launch date. Table 1 summarizes the current characteristics of this environment.

The SEL Process Improvement Program

The SEL process improvement program is a conscious, continuous effort to build higher quality systems at lower costs by understanding the environment, measuring and evaluating the results of planned process changes, and capturing and packaging experience to optimize the process and to anticipate uncontrollable changes.

Table 1. **Characteristics of the Development Environment Studied by the SEL**

Characteristics	Current State
Organization size	>250 people
Computing environment	HDS 8063 (IBM 3083) VAX 8820, 11/780
Languages	FORTRAN, Ada
Applications	Primarily attitude; some orbit and mission analysis
Average system size	180 KSLOC
Average project duration	2 years
Average staff level	15 to 20 people
Staff background	Computer Science, Mathematics, Physics

For a process improvement program to succeed, it must

- ***Be a conscious effort.*** Improvements will not happen by themselves; resources must be allocated to make them happen.

- ***Be a continuous effort.*** Even very mature processes need to be refined in the face of changing environments and advances in technology.

- ***Be built on a solid understanding of the environment.*** This includes characterizing the products produced and processes used.

- ***Achieve understanding by measurement and evaluation.*** The parameters of the environment must be quantified to evaluate the effectiveness of changes made to it.

- ***Feed back lessons learned.*** The results of measurement and evaluation must be fed back into the process to optimize it.

- ***Package lessons learned.*** Experiences must be packaged so that managers can apply them to their day-to-day challenges and can anticipate changes outside of their control, thus preserving corporate legacy when experienced people leave.

EVOLVING TO AN OPTIMIZING ENVIRONMENT

Given the principles of the SEL process improvement program, we can now look at that program in action over the SEL's first 15 years. For convenience, SEL activities are grouped into three broad classes: evaluating changes to life-cycle processes, evaluating changes to technology and methodology, and providing support to the development organization.

Changing Life-Cycle Processes

A first goal of the SEL was to establish a measurement program to capture and quantify the characteristics of the environment, including all its processes and products. The SEL spent much of the first 5 years simply learning how to collect, analyze, and interpret data. This early analysis showed that testing was one of the weakest activities in the flight dynamics development process, and it set the stage for several early experiments in changing a life-cycle process.

The goal in changing a life-cycle process is to identify a particular life-cycle phase or activity as a candidate for improvement, vary just that one element of the process, and then measure the impact on the process and product. If the analysis shows that the change favorably affects quality and/or productivity, it is incorporated into the process. In essence, this type of change can be viewed as "fine-tuning" an existing process.

In 1981, in a step to understand the weaknesses perceived in testing, the SEL evaluated the impact of independent verification and validation (IV&V) in the flight dynamics environment (Reference 2). It applied IV&V techniques on four flight dynamics projects, defined metrics for analyzing the change, and compared these metrics with those of earlier projects that did not use IV&V. The results showed little or no significant improvement in quality and reliability and, at the same time, reflected a substantial increase in development cost. The study concluded that IV&V was not cost effective for use in the SEL flight dynamics environment.

In 1984, continuing its quest to improve testing, the SEL compared three different software verification techniques (Reference 3). It trained a group of professional programmers in structural testing, functional testing, and the peer review technique of code reading, and then gave them programs that had been seeded with errors on which to apply these techniques. After the experimenters calculated such metrics as the number of errors found and the average effort expended to find each error, they concluded that code reading was the most cost-effective technique for uncovering errors in software units. As a result, code reading was incorporated as a formal activity into the flight dynamics software development process.

By participating in these life-cycle process change experiments, CSC has learned several lessons:

- To effectively evaluate and implement life-cycle changes, resources must be allocated; that is, an independent organization like the SEL must be designated, to focus on measuring and evaluating impacts. The job is too big for managers to do in their "spare time." We have carried this lesson beyond the SEL environment by establishing software engineering process groups to perform this type of analysis across the entire Systems, Engineering, and Analysis Support (SEAS) contract (Reference 4) currently being performed for GSFC.

- Peer review techniques are a cost-effective method for isolating errors early in the development life cycle. We have made such techniques a fundamental part of our SEAS System Development Methodology (Reference 5).

Changing Technology/Methodology

After about the first 5 years of studying the flight dynamics environment and its development process and experimenting with life-cycle process changes, the SEL looked back on its experiences and drew some basic conclusions. One was that following a formal methodology, provided that it is not "labor intensive," can produce a 10- to 15-percent improvement in a software development program compared to not following a formal methodology or following an ad hoc approach (Reference 1). Although adding and subtracting new techniques in the form of life-cycle changes can fine-tune the methodology, it does not produce substantial overall improvements to the program. To achieve substantial changes requires a major overhaul of the formal methodology itself or the insertion of new technology.

The SEL approach to methodology and technology changes is different from the relatively simple experimentation performed for life-cycle changes. Rather than performing a single experiment, evaluating the results, and deciding to implement a new technique across the entire program, the SEL knew that introducing an entire methodology or technology would require a more cautious approach because of risks associated with the immaturity of the methodology or technology and the extensive retraining of staff required. The SEL approach is to experiment with the new methodology or technology via a pilot project or

projects, evaluate the metrics collected, hypothesize about the potential benefits, and then repeat the experiment several times to confirm or deny initial hypotheses and to establish trends.

In 1984, the SEL began evaluating a methodology based on the Ada language and object-oriented design. This was a radical change from the top-down structured design techniques and the FORTRAN mindset then in place in the flight dynamics environment. To evaluate the new methodology, the SEL began an experiment in which the same flight dynamics simulator was built in two parallel development efforts: one in FORTRAN and the other in Ada. Known as the GRODY experiment, its results have been documented in a number of papers and reports in the SEL series (Reference 6). Since this first study, five more simulators have been built in Ada, and a separate study was performed to transport one of the simulators from a VAX environment to an IBM mainframe environment (Reference 7). Although the trends on these Ada projects are still being analyzed, a significant increase in reuse, with substantial development cost savings, seems to be the greatest benefit.

Another methodology change with which the SEL has begun to experiment recently is the cleanroom development methodology (Reference 8). This methodology relies on human discipline and peer review techniques to eliminate errors early in the life cycle. It isolates the designers and coders from the testers and prohibits the coders from even compiling their programs. Although the SEL had done some early evaluations of this methodology (the code-reading technique already discussed was adopted from the cleanroom methodology), it did not begin a cleanroom pilot project until 1988. The ACME project used the cleanroom approach to develop one of the subsystems for an attitude ground support system (AGSS). Initial ACME data showed an improvement in error rates (Reference 9). Currently, two other projects are using the cleanroom methodology to confirm and expand upon the initial trends observed on ACME. One effort is trying to reproduce the trends on another project of ACME's scale (approximately 30 KSLOC in size), and the other is trying to scale up and use the methodology on an entire AGSS (more than 150 KSLOC in size) to see if similar trends appear.

By participating in SEL methodology change experiments, CSC has learned other lessons:

- We have been able to minimize the risks of inserting new technology into the flight dynamics environment by measuring and evaluating impacts in a controlled fashion, allowing educated decisions to be made based on quantitative cost/benefit tradeoffs.
- In the case of Ada, we have been able to take advantage of the lessons learned on the pilot projects by communicating them to other organizations within our company through various technology exchange forums.

Supporting the Organization

The third category of activities in the SEL process improvement program is aimed at supporting the needs of the development organization rather than making controlled changes to the process or environment. This involves the concepts of effectively capturing and packaging experience.

Early in its history, the SEL defined and documented the methodology being used to develop flight dynamics projects. It published a series of documents that established standards and guidelines for both developers and managers in such areas as design, implementation, and testing techniques; life-cycle reviews and documentation;

planning, monitoring, and controlling projects; cost estimation; and product assurance (References 10–15). These documents helped capture experience in the flight dynamics environment and were instrumental in quickly training new staff. As technology changed and the SEL's domain grew, it became evident that these documents had to evolve as well. Thus, the SEL is currently updating this series with the dual objectives of (1) augmenting the methodology to broaden its scope and include new technology and (2) generalizing it where possible to provide greater flexibility for making future changes.

In a related activity, the SEL developed process models for the environment. A process model defines the expected behavior of a particular measure, such as staff resources expended, over the life cycle of a project. Process models capture the experience learned on past projects and package it in a form that can be used on current projects. Models give greater visibility into managing development projects. They allow managers to make at-completion predictions of such measures as resource utilization, error rates, and project schedules. They can also be used to determine when a project is deviating from the typical behavior of past projects and help to determine the causes of such deviations.

The SEL developed a tool that its managers use to take advantage of the SEL process models. This tool, the Software Management Environment (SME) (Reference 16), allows managers to use process models that are based on a pool of projects similar to the ones they are currently managing. It helps them analyze progress on their projects, predict outcomes, and plan alternatives, all with the advantage of using the experience base built up by the SEL in the flight dynamics environment.

The SEL process improvement program has also helped recognize and respond to the changing needs of the staff members in the environment. Over the 15 years since the SEL started, the primary background of the developers in the environment has shifted from mathematics and physics to computer science. In response to this, the SEL initiated a training program to give new developers a basic foundation in flight dynamics applications and quickly familiarize them with the SEL methodology.

By participating in SEL activities to support the organization, CSC has learned even more:

- We need to have a documented methodology used consistently across the environment. Drawing on the SEL's experience in documenting the methodology used in the flight dynamics environment and on our own, more general corporate methodology, we have documented a system development methodology for use across the entire SEAS contract (Reference 5), and we have supplemented this methodology with a set of standards and procedures (Reference 17) to help staff members apply it.

- We know that quantitative management works. Measuring process and product allows us to develop quantitative models that enable projects to be better planned, more accurately estimated, and more effectively controlled. We can also detect deviations from our plans more easily, and hence we can correct problems earlier. Recognizing the importance of quantitative management, we have packaged our experiences in this area in a data collection, analysis, and reporting handbook to be used by our managers on the SEAS contract (Reference 18).

- We need to train our organization in the methodology and in process improvement concepts. We have developed a required training program (Reference 19) for all engineers, developers, testers, integrators, and managers to ensure consistent

understanding and application of the SEAS System Development Methodology and of quantitative management techniques across the entire contract.

- We can write better proposals when estimates are backed up with solid data. From a business point of view, being able to point to a quantitative experience base lends credibility to proposals and brings in more work.

ASSESSMENT OF PROGRESS

We have seen some mechanics of the SEL process improvement program, some specific examples of the types of activities in which the SEL engages, and how CSC has benefited from participating in these activities. Using the SEL's own data, we now address the question "Are we any better?" by examining some growth, reliability, and productivity trends over the past 15 years.

Three areas measure changes in the nature of flight dynamics systems: complexity, general requirements, and system size. CSC performed a study in 1988 to examine trends in these areas, as well as in software reuse (Reference 20). At that time, it was generally felt that systems were becoming more complex, primarily as a reflection of the increased complexity of the spacecraft they were developed to support. Table 2, adapted from the study, shows a comparison of typical spacecraft configurations in the mid-1970s with those of the late 1980s. This table shows that the required attitude accuracy is 50 times greater than it was, data rates are over 14 times faster, and there are 3 times as many telemetry data types and 10 times as many sensors. In the above-mentioned study, these and other characteristics were combined into a synthetic measure of spacecraft complexity. A plot showing the overall trend in this complexity measure (Figure 1) shows that it has more than doubled over the past 15 years.

Table 2. Comparison of Spacecraft Characteristics

Characteristics	Mid-1970s	Late 1980s
Control	Spin stabilized	3-axis stabilized
Sensors	1	8 to 11
Torquers	1	2 to 3
Onboard computer	Analog, simple control	Digital, autonomous
Telemetry types	5	12 to 15
Data rates	2.2 kb/s	32 kb/s
Accuracy	1 degree	0.02 degree

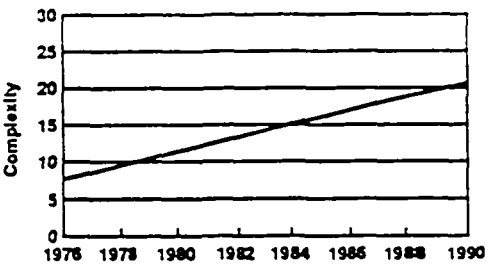


Figure 1. Trends in Spacecraft Complexity

The same study also derived a measure of functional specification complexity to reflect the growth in general requirements. This measure also more than doubled over the past 15 years, yet requirements growth was not directly proportional to spacecraft complexity. For example, going from a spacecraft with one sensor to a spacecraft with five sensors means that software must be developed to process data from all five sensors. Beyond that, however, it may also mean an additional requirement to create a

utility that determines the best time to use one sensor instead of another or to create a program that predicts periods when the motor that runs one sensor might interfere with the operation of another sensor. In addition, requirements have been added to build programs that perform such functions as predicting Earth occultation of a given set of stars, predicting Moon interference with sensor operation, or predicting antenna contact times. Thus, both increased spacecraft complexity and general requirements growth can be seen as separate drivers in the growth of system size.

In terms of system size, Figure 2 shows that total number of delivered lines of code (including blank lines and comments) has not quite tripled. At the same time, development error rates have been reduced by 65 percent (Figure 3). Figure 4 shows the trend in the cost per developed line of code. It has remained relatively constant, although the narrowing of the maximum and minimum range lines indicates that it is becoming more predictable.

Looking at all of these trends together now helps us answer our original question. Although a rigorous study of the relationship between spacecraft complexity, requirements growth, and system size has not been performed, one could expect that a doubling in both complexity and general requirements might result in a quadrupling of system size. Since system size did not quite triple, we conclude that developers are now packaging more functionality per line of code than they were 15 years ago. Thus, the SEL process improvement program has enabled us to build systems that provide more functionality per line of code, with significantly fewer errors per line of code, at a lower cost per line of code than systems of 15 years ago. It is clearly possible to improve productivity and lower error rates at the same time.

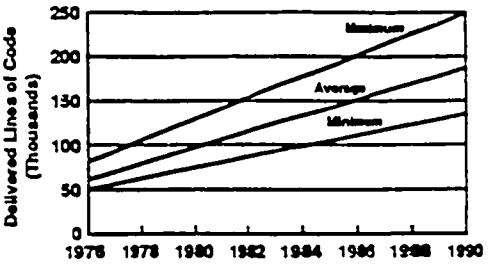


Figure 2. Trends in Size Growth of Flight Dynamics Applications

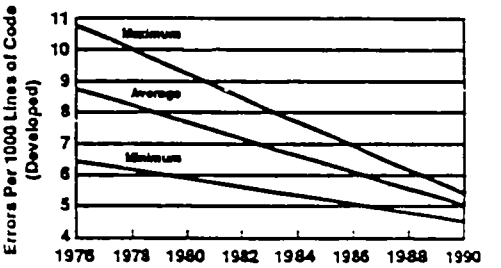


Figure 3. Trends in Development Error Rates

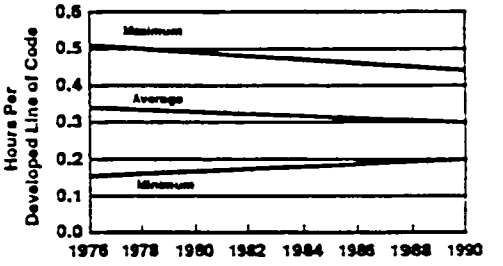


Figure 4. Trends in Software Cost

In addition, process models derived from SEL-collected data have helped us predict error rates and system costs more accurately. Thus, the answer to the question "Are we any better?" has to be an unqualified "Yes."

ACKNOWLEDGMENT

We would like to thank Michele Bissonette for her help in preparing this paper.

REFERENCES

1. NASA/GSFC Software Engineering Laboratory, SEL-81-104, *The Software Engineering Laboratory*, D. Card, F. McGarry, et al., February 1982

2. —, SEL-81-101, *Evaluation of an Independent Verification and Validation (IV&V) Methodology for Flight Dynamics*, G. Page, F. McGarry, and D. Card, June 1985

3. —, SEL-85-001, *A Comparison of Software Verification Techniques*, D. Card, R. Selby, et al., April 1985

4. NASA/GSFC, Request for Proposal (RFP) 5-74300/184, Systems, Engineering, and Analysis Support (SEAS), September 1986

5. Computer Sciences Corporation, *SEAS System Development Methodology (SSDM)*, July 1989

6. Software Engineering Laboratory, SEL-90-004, *Gamma Ray Observatory Dynamics Simulator in Ada (GRODY) Experiment Summary*, T. McDermott et al., September 1990

7. —, SEL-90-003, *A Study of the Portability of an Ada System in the Software Engineering Laboratory (SEL)*, L. Jun et al., June 1990

8. IBM Federal Systems Division, *Cleanroom Software Development Method*, M. Dyer, October 1982

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

10. —, SEL-81-205, *Recommended Approach to Software Development*, F. McGarry, G. Page, et al., April 1983

11. —, SEL-83-001, *An Approach to Software Cost Estimation*, F. McGarry, G. Page, et al., February 1984

12. —, SEL-84-101, *Manager's Handbook for Software Development*, L. Landis, F. McGarry, et al., November 1990

13. —, SEL-85-005, *Software Verification and Testing*, D. Card, C. Antle, and E. Edwards, December 1985

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

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

16. —, SEL-89-003, *Software Management Environment (SME) Concepts and Architecture*, W. Decker and J. Valett, August 1989

17. Computer Sciences Corporation, *SEAS System Development Methodology (SSDM) Standards and Procedures*, June 1990

18. —, *SEAS Software Measurement System (SSMS) Handbook*, to be published in early 1991

19. —, *Catalog of SEAS Training Courses*, September 1990

20. —, CSC/TM-89/6031, *A Study on Size and Reuse Trends in Attitude Ground Support Systems (AGSSs) Developed for the Flight Dynamics Division (FDD) (1976–1988)*, D. Boland et al., February 1989

VIEWGRAPH MATERIALS

FOR THE

G. PAGE PRESENTATION

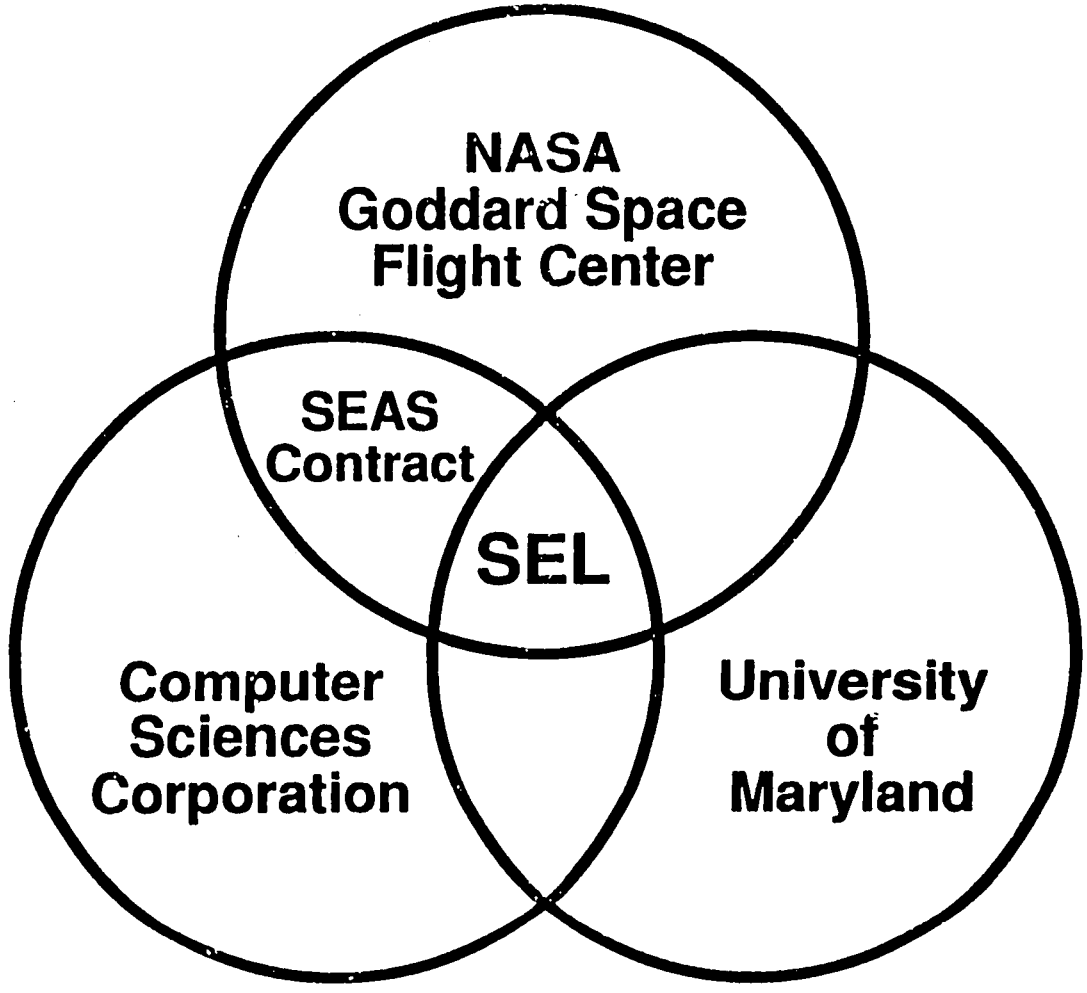
6269-0

Impact of a Process Improvement Program in a Production Software Environment: Are we any better?

Jerry Page

November 28, 1990

Computer Sciences Corporation
System Sciences Division


NASA
Goddard Space
Flight Center
SEAS
Contract
SEL
Computer
Sciences
Corporation
University
of
Maryland

Software Engineering Laboratory Environment

Characteristic	Current State
Organization Staff Level	> 250
Computing Environment	HSD 8063 (IBM 3083), VAX 8820, VAX-11/780
Languages	FORTRAN, Ada
Application	Attitude, orbit, mission analysis
Average System Size	180 KSLOC
Average Project Duration	2 Years
Average Staff Level	15 to 20
Staff Background	Computer Science, Mathematics, and Physical Sciences

What Is a Process Improvement Program?

A *conscious, continuous* effort to build higher quality systems by

- ***Understanding* the environment**
- ***Measuring* and *evaluating* the results from planned changes**
- ***Capturing* and *packaging experience* to *optimize* the process and to *anticipate* uncontrollables**

Life Cycle Process Changes

- Perform experimental studies on production projects
- Vary one element of process and measure impacts on process and product
- Fine tune process to take advantage of benefits

Testing Studies	Observations and Actions
Code Reading, Functional and Structural Testing	• Code reading most effective technique • Add to process
Independent Verification and Validation	• Small effects for relatively large cost • IV&V inappropriate for SEL projects

Technology/Methodology Changes

- Test new technology in production environment with pilot project
- Measure impacts on project profiles and products produced
- Package lessons learned, adjust training, and repeat for effect

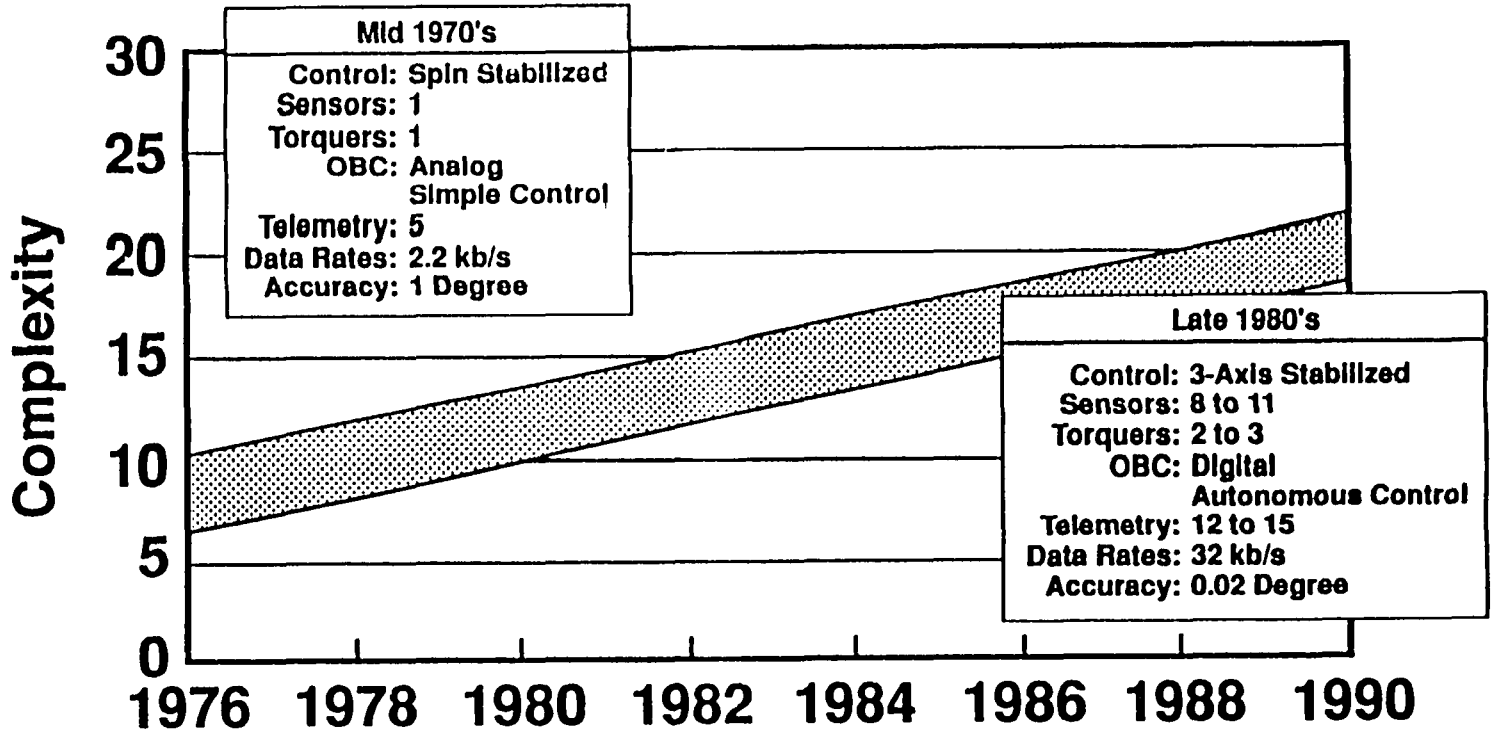
Technology	Observations and Actions
Ada	• Very promising trends on software reuse • Conduct further and more detailed studies
Cleanroom	• Initially, error levels very low • Scale up experiment and verify findings

Organizational Changes

- Look for deviations from process models
- Determine impacts
- Strengthen definitions of overall approach

Change	Action Taken
Staff Turnover or Staff Growth	• Created and augmented standards and guidelines • Developed Software Management Environment (SME)
Staff Background	• Established required training program for developers and managers • Developed Software Development Environment (SDE)
Domain Growth	• Augmented methodology to broaden scope • Generalized methodology to make it more flexible

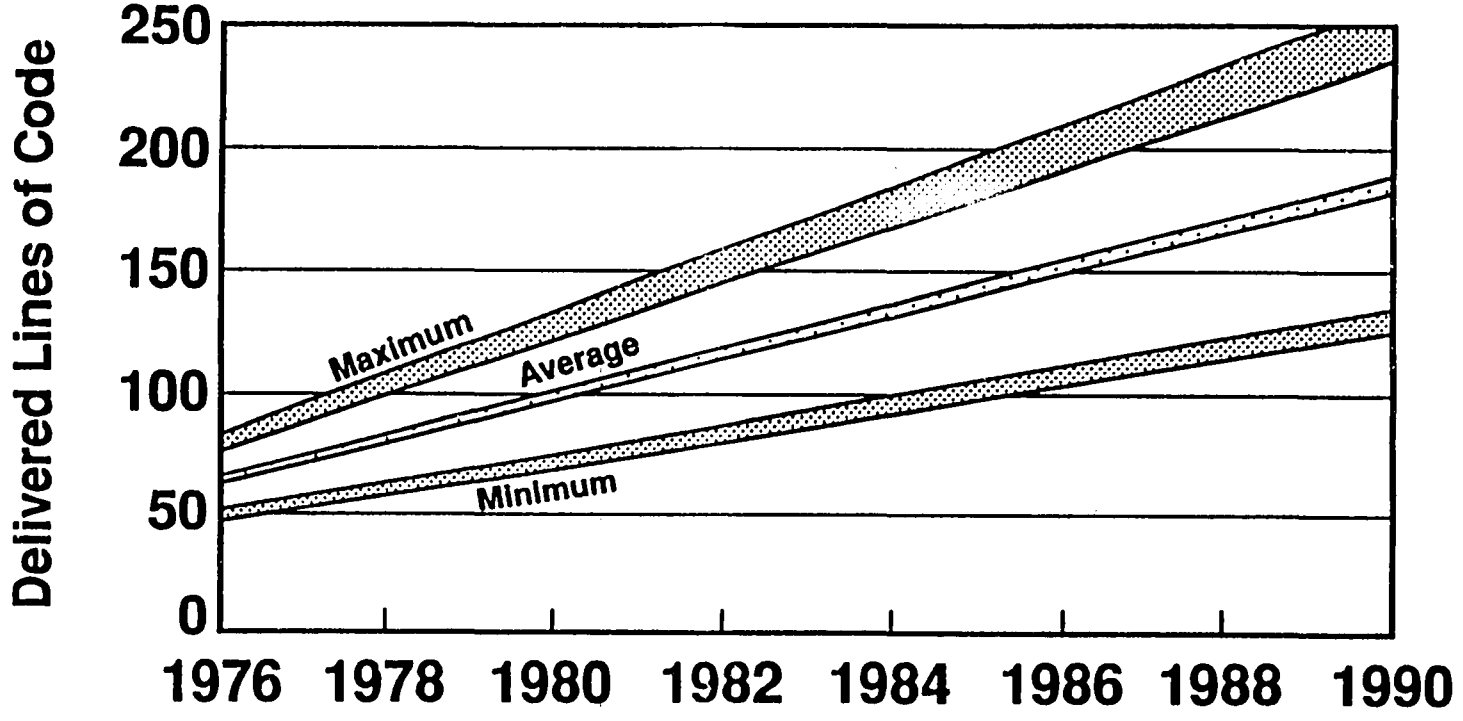
System Complexity*



Complexity has more than doubled.

*D. Boland, *A Study on Size and Reuse Trends In Attitude Ground Support Systems (AGSSs) Developed for the Flight Dynamics Division (FDD) (1976–1988)*, CSC/TM-89/6031, CSC, February 1989

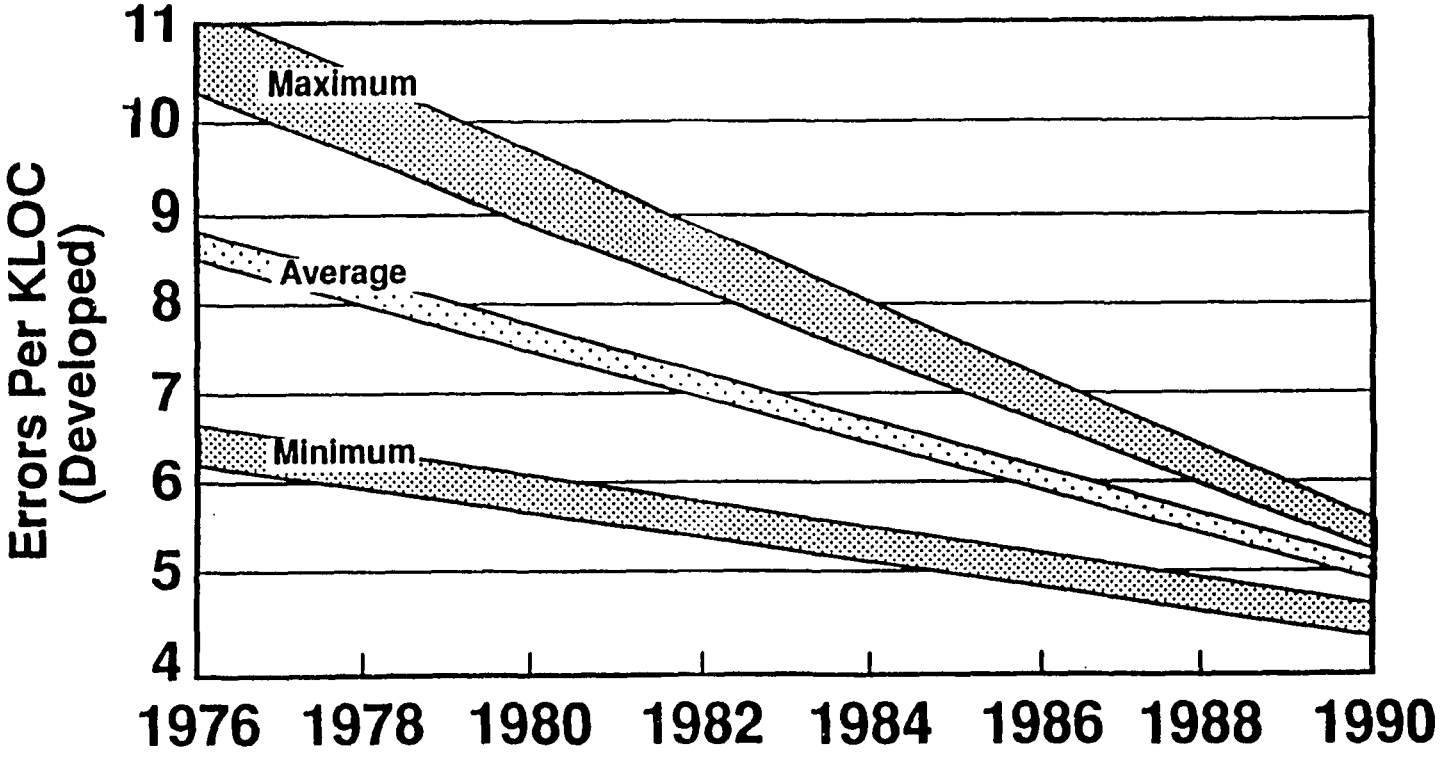
System Size



System size has more than doubled.

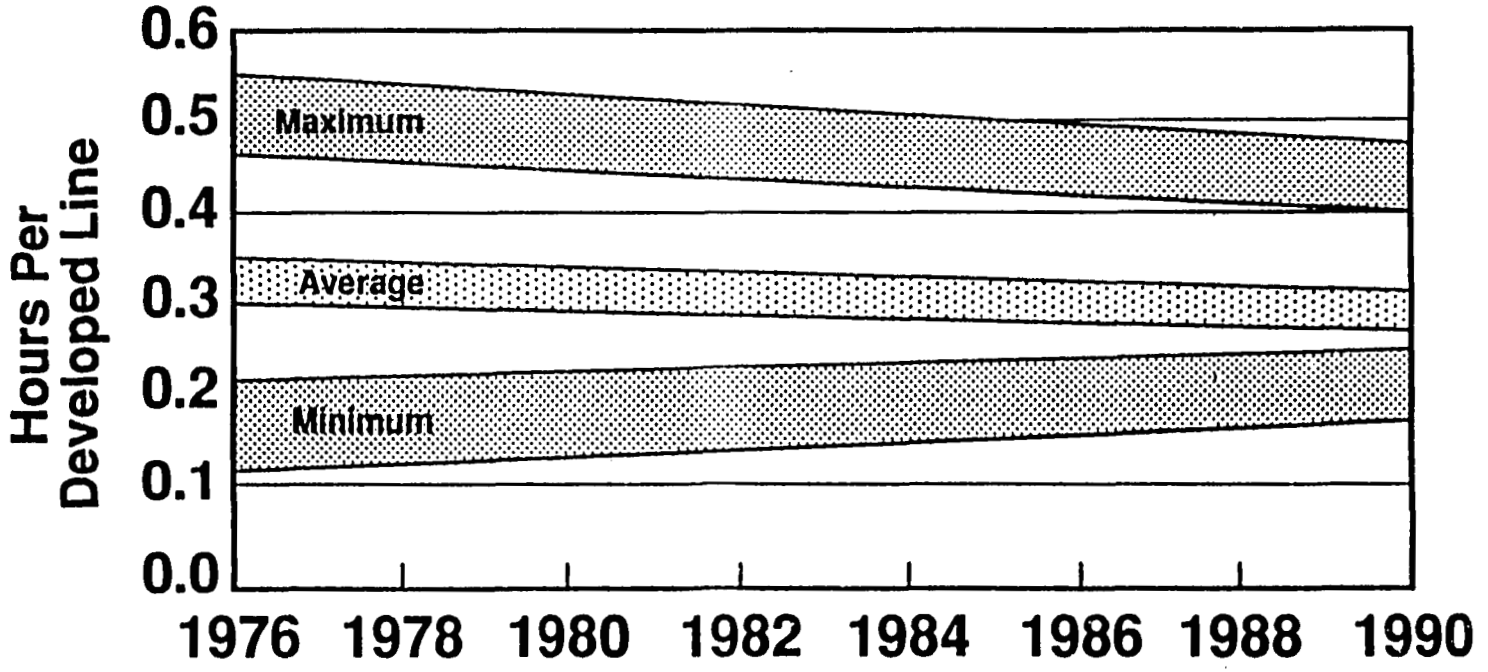
Computer Sciences Corporation
System Sciences Division

Development Error Rates



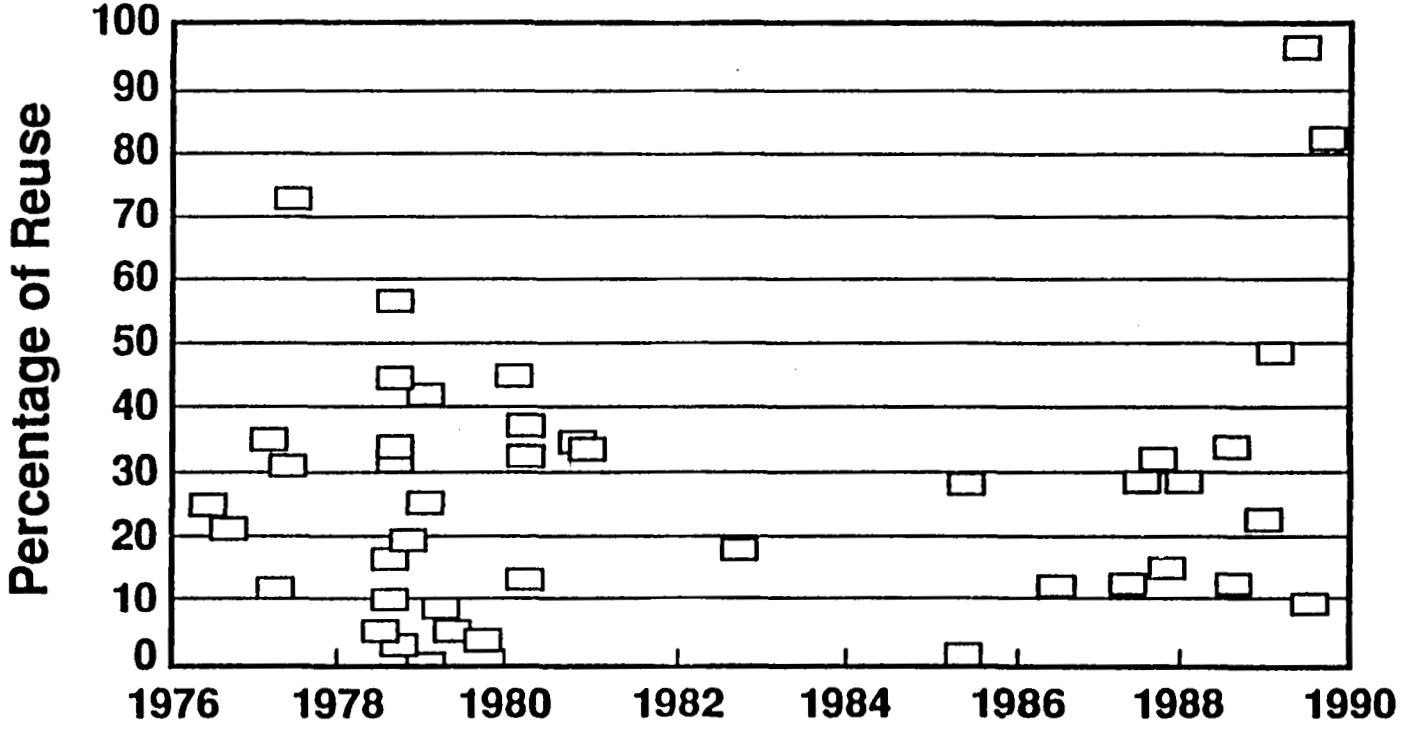
Error rates have been reduced by 65 percent.
Error models are fairly well established.

Cost of Code



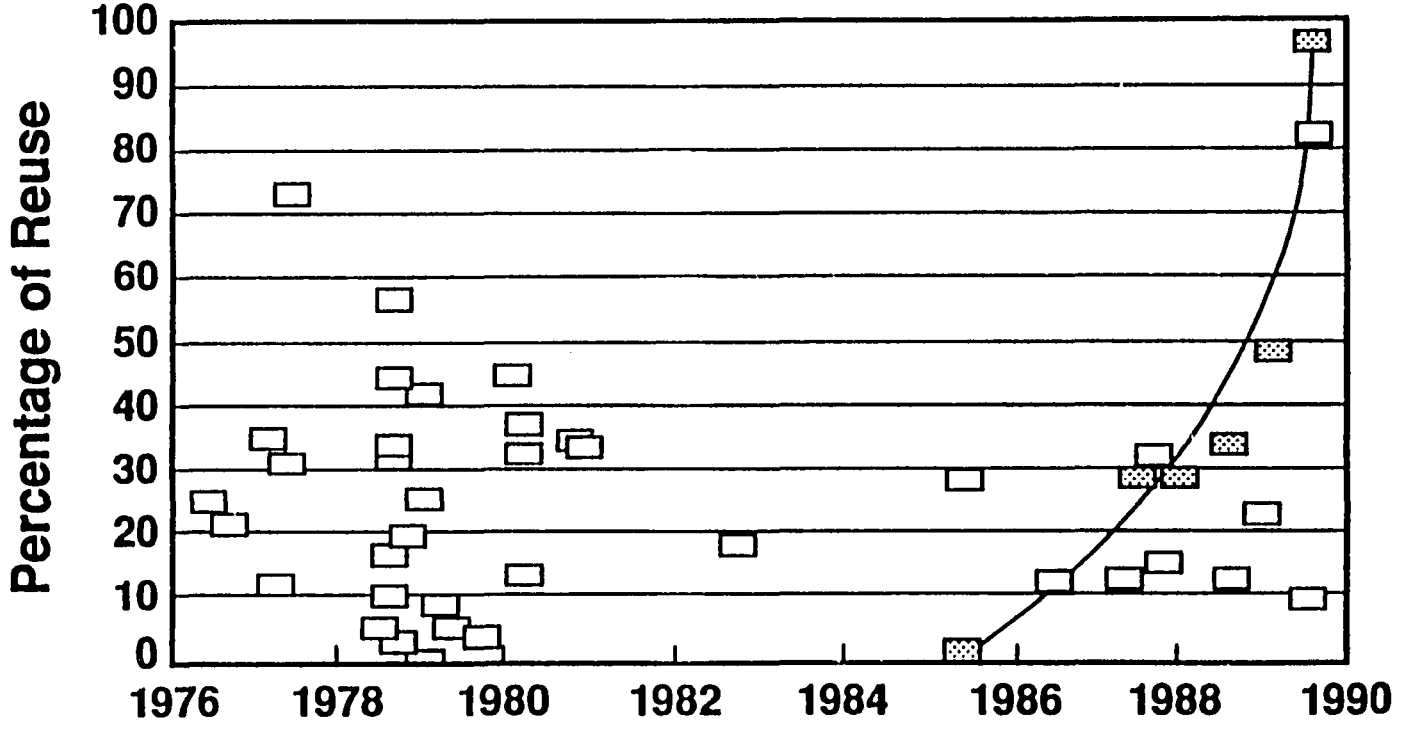
Cost per LOC remained relatively constant. Predictability is improving.

Code Reuse in Systems



Sometimes interesting things in a picture are lost because of shallow depth of field.

Code Reuse in Ada Simulators



However, searching with a reduced field of view can pay off.

What CSC Has Learned

- Quantitative management works
- Peer review works
- You can lower error rates
- You can raise productivity
- You can write more credible proposals when you can back them up with data

What Has CSC Done to Capitalize on Its Learning?

- Developed a System Development Methodology based on its experience
- Packaged its experience with quantitative management in a manager's Data Collection, Analysis, and Reporting Handbook
- Developed a set of standards and guidelines to complement its methodology
- Developed required training programs for engineers, developers, testers, integrators, and managers to maximize the benefits of its methodology
- Established measurement-based Engineering Process Groups to identify improvement areas, recommend changes, and evaluate the impact of those changes

Presentation Contributors

Gerry Heller
Tim McDermott
Sharon Waligora