Profile of Software Engineering Within the National Aeronautics and Space Administration (NASA)

19th Annual Software Engineering Workshop

Craig C. Sinclair, Science Applications International Corporation (SAIC)
Kellyann F. Jeletic, NASA/Goddard Space Flight Center (GSFC)

ABSTRACT

This paper presents findings of baselining activities being performed to characterize software practices within the National Aeronautics and Space Administration. It describes how such baseline findings might be used to focus software process improvement activities. Finally, based on the findings to date, it presents specific recommendations in focusing future NASA software process improvement efforts. NOTE: The findings presented in this paper are based on data gathered and analyzed to date. As such, the quantitative data presented in this paper are preliminary in nature.

BACKGROUND

The NASA Software Engineering Program was established by the Office of Safety and Mission Assurance (Code Q) at NASA Headquarters in 1991 to focus on the increasingly large and important role of software within NASA. The primary goal of this program is to establish and implement a mechanism through which long-term, evolutionary software process improvement is instilled throughout the Agency.

NASA's Software Engineering Program embraces a three-phase approach to continuous software process improvement. The first and most crucial phase is *Understanding*. In this phase, an organization baselines its current software practices by characterizing the software product (e.g., size, cost, error rates) and the software processes (e.g., standards used, lifecycle followed, methodologies employed). During the Understanding phase, models are developed that characterize the organization's software development or maintenance process. Models are mathematical relationships that can be used to predict cost, schedule, defects, etc. Examples are the relationships between effort, code size, and productivity or the relationship between schedule duration and staff months. This in-depth understanding of software practices is gained within the context of a specific software domain and must precede any proposed change. In the second phase, *Assessing*, a software improvement goal is identified. Based on the specific local organizational goal, a process change is introduced and its impact to the software process and product is measured and analyzed. The results of the Assessing phase are then compared back to the baseline developed during the Understanding phase. In the third phase, *Packaging*, experiences gained and lessons learned are packaged and infused back into the organization for use on ongoing and subsequent projects. Forms of packaging typically include standards, tools, training, etc. This three-phase software process improvement approach (Figure 1) is iterative and continuous.

The importance of the Understanding phase cannot be emphasized enough. Before an organization can introduce a change, it must first establish a baseline with which to compare the measured results of the change. This baseline must be domain-specific and the software goals of the organization must be clearly understood. Continual baselining is necessary not only because people, technology, and activities change, but also because identifying, designing, implementing, and measuring any change requires an in-depth understanding and monitoring of the particular process on which the change is focused. This implies that understanding and change are closely coupled, necessarily iterative, and never-ending. Continual, ongoing understanding and incremental change underlie any process improvement activity.

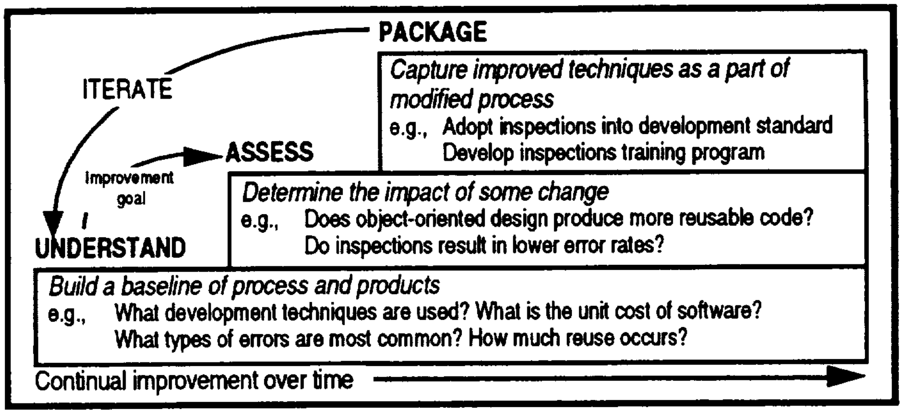

PACKAGE
ITERATE
Capture improved techniques as a part of modified process
e.g., Adopt inspections into development standard
Develop inspections training program
ASSESS
Improvement goal
Determine the impact of some change
e.g., Does object-oriented design produce more reusable code?
Do inspections result in lower error rates?
UNDERSTAND
Build a baseline of process and products
e.g., What development techniques are used? What is the unit cost of software?
What types of errors are most common? How much reuse occurs?
Continual improvement over time

Figure 1. Three-Phase Approach to Software Process Improvement

This paper addresses the Understanding phase, that is, the baselining of NASA software. Since the baselining activities focus on a global organizational level, that is, NASA as a whole, the difference between applying the process improvement approach at the global level rather than at a local organizational level must first be addressed.

SOFTWARE PROCESS IMPROVEMENT AT A GLOBAL LEVEL

The steps in the software process improvement approach are applied differently at the global and local organizational levels. Figure 2 illustrates the differences between the local and global approaches. The *Understanding* phase is predominantly the same at both levels; basic characteristics of software process and product are captured. At a local level, models are also developed, e.g., cost and reliability models, to help engineer the process on ongoing and future projects. At a global organizational level, models can only be very general relationships, such as the percentage of application software in each of the identified software domains of an organization.

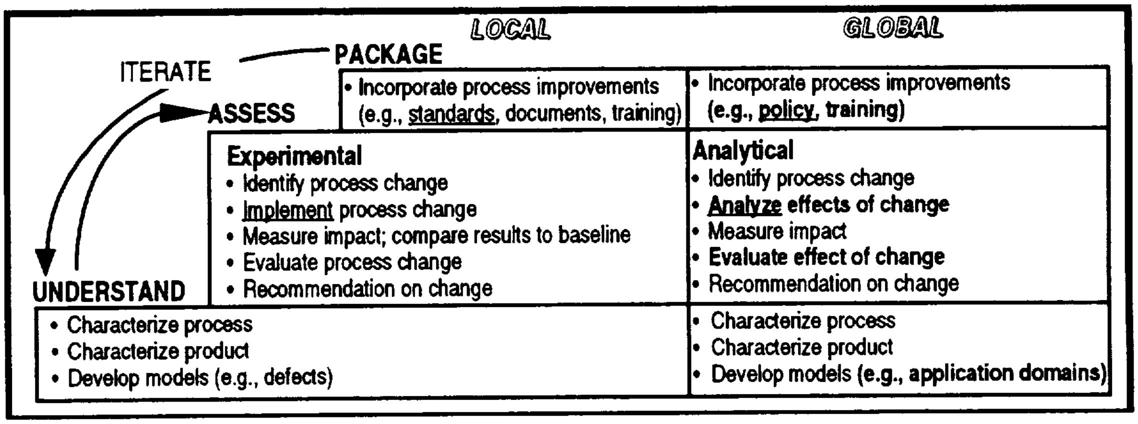

LOCAL
GLOBAL
ITERATE
PACKAGE
• Incorporate process improvements (e.g., standards, documents, training)
• Incorporate process improvements (e.g., policy, training)
ASSESS
Experimental
• Identify process change
• Implement process change
• Measure impact; compare results to baseline
• Evaluate process change
• Recommendation on change
Analytical
• Identify process change
• Analyze effects of change
• Measure impact
• Evaluate effect of change
• Recommendation on change
UNDERSTAND
• Characterize process
• Characterize product
• Develop models (e.g., defects)
• Characterize process
• Characterize product
• Develop models (e.g., application domains)

Figure 2. Local versus Global Software Process Improvement Approach

It is in the *Assessing* phase where most differences occur. At the local level, the Assessing phase is experimental in nature. Specific technologies are introduced to try to attain some local goal (e.g., inspections might be introduced to reduce error rates). The results of these experiments are then compared to the baseline from the Understanding phase to determine what impact the change has had. At a global level, the Assessing phase is analytical rather than experimental. Process changes are identified and the effects of the change(s) are analyzed and evaluated. Recommendations are then made at an organizational level. For example, a potential process change is identified such as code reuse. Analysis and evaluation of the effects of increased reuse in an organization is accomplished by determining which software domains would benefit from reuse, measuring via survey the amount of reuse that currently takes place in those domains, and projecting the potential development time and cost savings that could be achieved by instituting a focused reuse program. Finally, specific recommendations are developed for the organization that stimulate the local implementation of code reuse.

The third phase, *Packaging*, is also similar at both levels. Changes that result in identified improvements are packaged and infused back into the organization's process. There are differences in the types of packages produced at both levels. At the local level packages might include experience-driven standards, guidebooks, training, and tools. Packages at the global level might include a high level training plan or a policy requiring software process improvement activities for various software domains and organizational levels. The global approach is intended to stimulate local implementations so each individual organization can attain its local goals and improve its products and processes. NASA will benefit, as a whole, as local benefits are attained in software organizations throughout the Agency.

BASELINING NASA'S SOFTWARE

As the critical first step toward continual software process improvement, NASA has recently begun the Understanding phase and has baselined its software products and processes. The Mission Operations and Data Systems Directorate (Code 500) at the Goddard Space Flight Center (GSFC) was first characterized to prototype and refine the steps necessary to construct such a baseline [Reference 1]. With the experiences gained during the Code 500 efforts, a single NASA Field Center, GSFC, was then baselined [Reference 2]. Lessons learned were again factored into the process and, finally, NASA as a whole was baselined to determine current Agency software practices. Since the NASA-wide data collection and analysis are not yet complete, this paper presents findings to date. The final NASA baseline, the *Profile of Software at the National Aeronautics and Space Administration,* is nearly complete and is targeted for completion in early 1995 [Reference 3].

During fiscal year 1993 (FY93), NASA software and software engineering practices were examined to gain a basic understanding of the Agency's software products and processes. The objective of the NASA baseline was to understand the Agency's software and software processes. There is no intent to judge right or wrong; it merely presents a snapshot in time of software within NASA. The baseline includes all software developed or maintained by NASA civil servants or under contract to NASA. It does not include commercial-off-the-shelf (COTS) software such as operating systems, network software, or database management systems. It also does not include COTS application packages such as word processing packages, spreadsheet software, graphics packages, or other similar tools hosted on workstations and personal computers.

To produce the baseline, software product and process data were gathered from seven NASA Field Centers[1] and the Jet Propulsion Laboratory. Data and insight gathering were performed using four approaches:

(1) *Surveys* administered in person to a representative set of civil servants and support contractors from across the NASA community

(2) *Roundtable* discussions consisting of a structured group interview process

(3) One-on-one *interviews* with management and technical personnel

(4) *Reviews* of organizational and project data (e.g., budgets, policies, software process development documentation)

Reference 4 provides additional details on the baselining approach.

[1] Data were collected from the following NASA Field Centers: Ames Research Center, GSFC, Johnson Space Center, Kennedy Space Center, Langley Research Center, Lewis Research Center, and Marshall Space Flight Center

The remainder of this paper focuses on the findings of the NASA baseline and how they might be used. The baseline will help NASA management understand the scope and extent of software work within the Agency. It will also assist managers in focusing future resources to improve NASA's software products and processes. The baseline can also be assessed to identify candidate areas for improvement. As the baseline findings are presented, examples are given as to how they might be used. Finally, recommendations are proposed for focusing future process improvement efforts.

NASA'S SOFTWARE PRODUCT BASELINE

This section presents results from the analyses performed on the product data gathered throughout the NASA Centers. This section summarizes a selected set of the software product baseline data that can be found in the draft *Profile of Software at the National Aeronautics and Space Administration* [Reference 3]. Examples of additional software product data detailed in the document include the amount of operational software per NASA Field Center, the size of the software domains at the Centers, allocation of resources to the life-cycle phases, and other measures.

The software product baseline characterizes the attributes of the software itself. This paper addresses several questions pertaining to NASA's software product:

- What *classes* of software exist?
- *How much* software exists?
- How much of NASA's *resources* are invested in software?
- What *languages* are used?

These product characteristics are discussed below.

SOFTWARE CLASSES

Six classes (domains) of software were identified throughout NASA. It was necessary to define separate software domains within NASA, since the development and maintenance practices, the management approach, and the purposes of the software in various domains are distinctly different. Hence the software improvement goals for varying domains are generally different. The definitions of the six NASA software domains are given below.

- *Flight/embedded* -- embedded software for on-board spacecraft or aircraft or ground command and control applications (e.g., robotics)
- *Mission ground support* -- software usually not embedded; operates on the ground in preparation for or in direct support of space and aircraft flight missions (e.g., flight dynamics, control center, command processing software, and software for crew or controller training)
- *General support* -- software that supports the development of flight and ground operational software (e.g., engineering models, simulations, engineering analyses, prototypes, wind tunnel analyses, test aids, and tools)
- *Science analysis* -- software used for science product generation, processing and handling of ancillary data, science data archiving, and general science analysis
- *Research* -- software supporting various studies in software, systems, engineering, management, and/or assurance (e.g., software tools, prototyping, models, environments, and new techniques)
- *Administrative information resources management (IRM)* -- software supporting administrative applications (e.g., personnel, payroll, and benefits software)

Figure 3 shows the distribution of these domains for operational software. Mission ground support and administrative/IRM software were found to be the largest and most prevalent software domains within NASA, accounting for over 60 percent of all NASA software. General support software was the next largest software domain, accounting for almost 20 percent of NASA software. The science analysis, research, and flight/embedded software domains were much smaller in size.

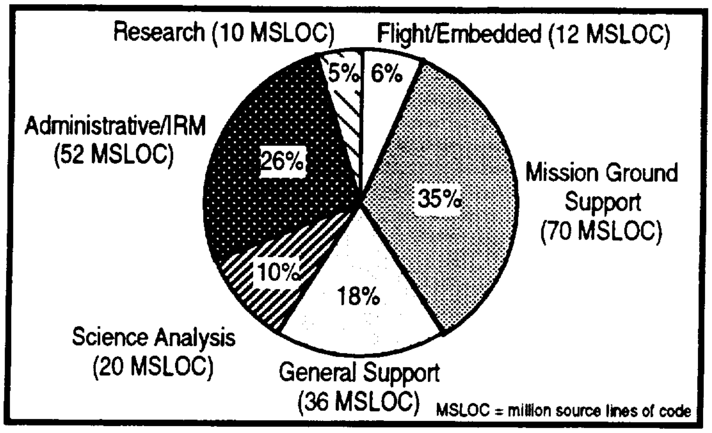


Figure 3 - Operational Software by Domain

How might such baseline information be used? The largest domains could indicate where software improvement efforts might most effectively be applied.

SOFTWARE QUANTITIES

During the baseline period, about 200 million source lines of code (SLOC) were in operational use. During that same period, NASA developed about 6 million SLOC (MSLOC). In terms of lines of operational code, almost 122 million SLOC within NASA is mission ground support (70 MSLOC) software and administrative/IRM (52 MSLOC) software. As mentioned in the previous section, focusing an effective software improvement program in these software domains has the potential of reaping enormous cost benefits. This type of data can be used to assist NASA management in seeing where they should focus their resources to improve software products and processes.

SOFTWARE RESOURCES

Figures 4 and 5 show the amount of resources invested in software in dollars and manpower, respectively. As these figures indicate, NASA has a significant investment in software. More than $1 billion of NASA's total $14 billion budget is spent on the development and maintenance of software (Figure 4). Most of NASA's software budget is spent on contractors, nearly 80 percent of NASA's software work is contracted out to industry. Software staffing accounted for more than 10 percent of NASA's total work force (Figure 5). This includes all civil servants and contractors who spend the majority of their time managing, developing, verifying, maintaining, and/or assuring software. These data can be used to help senior managers at NASA to understand the scope and extent of NASA's investment of manpower and budget in software.

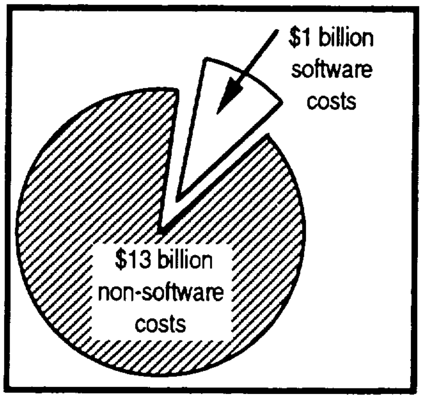


Figure 4 - Software Versus Total Costs

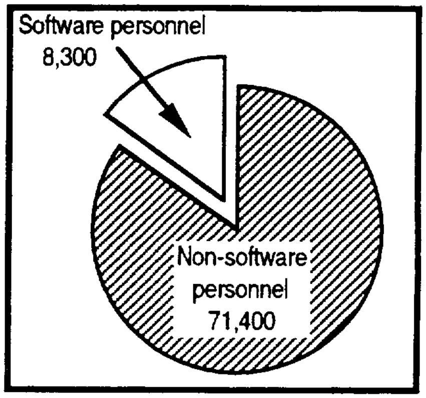


Figure 5 - Software Versus Total Staffing

SOFTWARE LANGUAGES

Figure 6 compares the preferences in software languages being used in current development efforts across NASA with those used in existing software now being maintained. Several trends are apparent. FORTRAN usage has remained relatively constant. Usage of both Cobol and other languages (e.g., Assembler, Jovial, Pascal), has decreased significantly, presumably replaced by the large increase in C/C^{++} usage. The usage of both C/C^{++} and Ada have increased dramatically. This implies that there is a significant trend toward C/C^{++} across NASA. Another trend is the lack of substantial movement toward Ada despite a decade of attention within NASA and advocacy from the Department of Defense. Although Ada use has increased, the magnitude of the increase is small compared to the intensity of past advocacy. It appears that Ada is not "catching on" within NASA culture and that C/C^{++} are becoming the languages of choice.

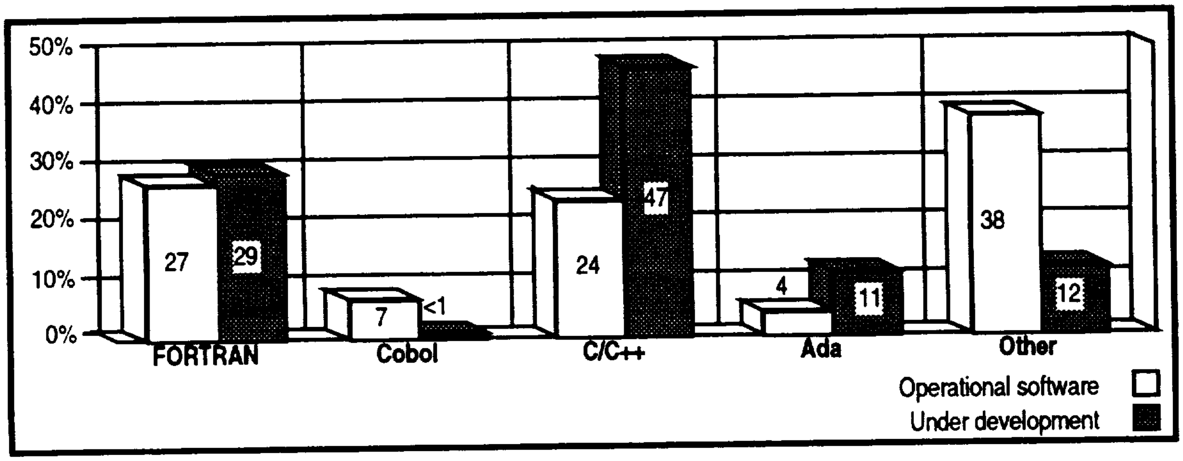


Figure 6. Language Preferences and Trends

Data such as the language preferences and trends might be used to focus training activities, not only toward language training, but also toward methodologies appropriate to specific languages.

NASA'S SOFTWARE PROCESS BASELINE

This section presents results from the analyses performed on the process data gathered throughout the NASA Centers. It summarizes a selected set of the software process baseline data that can be found in Reference 3. Examples of additional software process data detailed in the document include management experience, documentation standards, development tools, training, and other processes.

The software process baseline characterizes the attributes of the software practices. This paper addresses several questions pertaining to NASA's software process:

- What software *standards* are used and are helpful?
- How are *requirements managed*?
- How much and what type of *reuse* occurs?
- What are the Agency's practices with respect to software *metrics (measures)*?
- What *development methodologies* are used?

These process characteristics are discussed below.

SOFTWARE STANDARDS

A software standard refers to any mutually agreed upon specification for a software product or a software process within a software development or maintenance project. Examples of software standards related to software products are coding standards, language standards, and error rate specifications. Examples of software standards related to software processes are specifications of software development standards, software configuration standards, and software methodologies. Almost all the written, baselined software standards within NASA are in the form of software development standards. This is a type of process standard that consists of one or more of the following: software life-cycle phases and their activities, software review requirements, and document format and content requirements. Though software standards exist at various levels within NASA organizations, there is relative little usage of software standards by NASA personnel. On the contrary, standards usage is widespread among NASA's support contractors, which is significant considering that they are responsible for nearly 80 percent of NASA's software work.

One resounding sentiment throughout the Agency was that the most used and useful software standards are typically defined at the project level. Software standards defined and imposed from higher organizational levels were widely ignored. Another observation supported by the process data was that the awareness of software standards baselined at higher organizational levels was relatively low. In fact, there was a clear trend that indicated that the higher up in the organizational chain the standard is baselined, the less likely the project software staff know of its existence.

When software standards do exist, they do not enjoy a high level of use and do not appear to be used by the majority within an organization. This observation appeared to be true at all organizational levels. However, when software standards are used, they are generally perceived as helpful. So even though software standards do not have an overall high level of use, those that do use them generally perceive them to be helpful. Finally, even when software standards exist and are used, they are not enforced by the organizational level at which they are baselined.

This information can be used to provide specific focus in developing and facilitating the effective use of software standards within the Agency.

REQUIREMENTS MANAGEMENT

Software requirements represent an agreement between NASA and its contractors as to what will be produced and how it will perform. These "agreements" form the basis for the software size, schedule, budget, and staffing levels. If the software requirements are not clearly defined before the onset of design, schedule slips, code growth, and cost overruns are often the result. Management of software requirements is especially important for NASA civil servants since over 80 percent of the software projects at NASA are developed or maintained by contractors.

A widespread finding throughout NASA was that unstable requirements were perceived as the major cause of software schedule slips, cost overruns, and code size growth problems. Unstable requirements were interpreted to mean not only changing requirements, but also missing and/or misinterpreted requirements. A related finding was that most of the NASA engineers and managers surveyed claimed that software requirements were generally not stable by the onset of preliminary design.

SOFTWARE REUSE

Software reuse is the establishment, population, and use of a repository of well-defined, thoroughly tested software artifacts. Software artifacts that can be reused include not only code, but software requirements specifications, designs, test plans, documentation standards, etc.

Throughout NASA, most focus on reuse is at the code level. On average, about 15 percent of code is reused from one project to another, however, there is considerable variance in reuse levels between Centers. The level of reuse was also observed to widely vary between projects within a given Field Center. In NASA overall, there was little in the way of defined approaches for handling software reuse.

SOFTWARE MEASUREMENT

Software measures are quantitative data that specify a set of characteristics about the software product or its processes. Software measures can be used to aid in the management of software projects, help in the estimation of new projects, define and model an organization's software characteristics, and guide improvements of software and its processes.

The collection and utilization of software measures varied from non-existent to a few robust programs. Overall, relatively few NASA organizations collected software measures. Of those organizations surveyed that did collect software measures, less than half used the data to analyze and provide information back to the project. Overall, there was little evidence of the collection and use of measures throughout NASA.

DEVELOPMENT METHODOLOGIES

Figure 7 shows the relative awareness, training, and usage of several software development methodologies. Since structured analysis and Computer-Aided Software Engineering (CASE) tools have been around for a long time, it is not surprising that they are well known and widely used. There is a lot of awareness about object-oriented technologies, but usage is moderate. Some newer technologies, e.g., Cleanroom, are much less known and used. With the exception of CASE, one can also see a rather close link between the level of training and the level of usage. CASE is not a surprising exception since, as with other tools, people tend to jump in and use them rather than take courses or delve through documentation. One might surmise by the link with training and usage that NASA may be investing in "just in time" training.

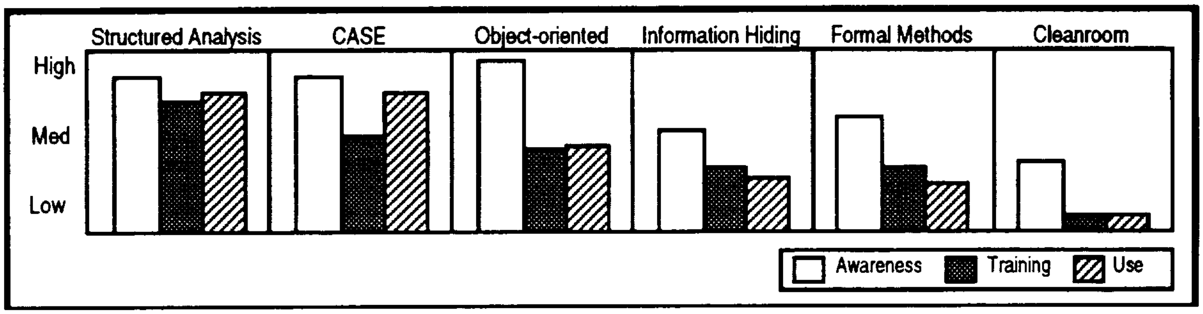


Figure 7. Development Methodologies

APPLYING THE FINDINGS

As previously indicated, the NASA baseline can be used to identify candidate areas for improvement and to develop specific recommendations for implementation of software improvement within NASA. These software improvement recommendations must not consist of rigid NASA-wide requirements imposed from above onto NASA projects. Rather, the software improvement recommendations at the higher levels of organization within NASA need to be top level policy and funding assistance, designed to stimulate and facilitate the development of local implementations of software improvement methods. If the goal is to bring software improvement into the projects, the projects must be given proper incentives and allowed to tailor software improvement implementation to their specific goals and domains. The following are two examples of how the NASA software baseline findings could be assessed and utilized.

Software Reuse

Recall that, on average, about 15 percent of the code is reused from one project to another. Throughout NASA, there is little or no emphasis on reusing anything but code. Overall, there are few defined approaches to reuse and only a few NASA organizations utilize software reuse as part of their software development process.

There are some NASA organizations who focus on more than just the reuse of code (e.g., reuse of code and architecture). These organizations have seen 75 to 80 percent reductions in both the time and cost to develop software. NASA might be able to leverage these few robust programs to assist the adoption of software reuse by other NASA organizations.

Applying proven NASA-developed solutions to the same software domains of other NASA organizations will give a much higher probability for success within the NASA culture.

Software Measurement
Recall that there is little evidence of collection and use of software measures throughout NASA. Collection and use ranged from non-existent to a few robust programs.

Software measurement is critical for project management and for the success of any software process improvement effort. Without measurement, change and improvement cannot be demonstrated. Here also, NASA might be able to leverage the few robust measurement programs to assist in the adoption of measurement by other NASA organizations. As in the case of reuse, applying domain-consistent, NASA-developed solutions to projects has the best chance for acceptance in the NASA culture.

In both examples, NASA and Center level policies could be put in place to encourage the reuse and software measurement programs by the projects. The existing positive examples of projects using reuse and software measurement could be packaged in a way that could be useful for other projects. In some cases, appropriate NASA and Center funding assistance could be applied to get the programs started. The projects themselves should then be responsible for setting their own project-specific goals, tailoring the packaged software improvement processes, and implementing them in a way that contributes positively to their projects.

Other baseline findings can be examined to extract similar observations and to make recommendations for improvement. In analyzing the baseline, software domains and organizational levels must be considered. First, consider software domains. Examining reuse in domains that perform repeated tasks, e.g., mission ground support software, would probably be more beneficial than examining reuse in the area of research software where most software is one-of-a-kind. Similarly, research software might not require much in the area of software measurement. When analyzing the baseline, identifying areas for improvement, applying the findings, and implementing changes, software domains must be considered.

Organizational levels also play a key role in analyzing and applying the findings. Higher organizational levels (e.g., NASA and Center level) should focus on encouraging local implementations via policy and funding assistance. Local projects should determine their own goals and devise an implementation of the software improvement area that fits their experience and domains.

RECOMMENDATIONS

Based on the findings to date, some recommendations can be made. First, since a significant portion of NASA's resources (both manpower and budget) is spent on software, each NASA Center and significant software organization should establish a software baseline.

Second, since project level standards are the most used and useful, NASA should focus on project and domain level standards, NOT on NASA-level standards.

Finally, NASA should assess the existing baseline to identify areas for software improvement. Based on the assessment, recommendations should be developed. At the very least, these recommendations should focus on software reuse and software measurement.

SUMMARY

This initial baseline of NASA software provides the answers to basic questions about NASA's software practices. It can provide insight for NASA to focus on potential areas of improvement. It also provides a snapshot in time to be used for future comparisons as changes are introduced and NASA's software process evolves.

This baseline is a first step toward continual software process improvement. It also must be the first of many baselines. As the Agency's process evolves, this baseline must be reexamined and updated to accurately characterize NASA's software practices at that point in time. Maintaining a baseline is critical to retain an ongoing understanding of NASA's software process and products. Without such understanding, improvements cannot be identified and continual software process improvement cannot be attained.

REFERENCES

1. *Profile of Software Within Code 500 at the Goddard Space Flight Center*, Hall, D. and McGarry, F., NASA-RPT-001-94, April 1994.

2. *Profile of Software at the Goddard Space Flight Center*, Hall, D., Sinclair, C., and McGarry, F., NASA-RPT-002-94, June 1994.

3. *Profile of Software Within the National Aeronautics and Space Administration*, Hall, D., Sinclair, C., Siegel, B., and Pajerski, R., NASA-RPT-xxx-95, Draft, January 1995.

4. *Profile of NASA Software Engineering: Lessons Learned from Building the Baseline*, Hall, D. and McGarry, F., Eighteenth Annual Software Engineering Workshop, SEL-93-003, December 1993.

ACRONYMS AND ABBREVIATIONS

CASE	Computer-Aided Software Engineering
Code 500	Mission Operations and Data Systems Directorate (at GSFC)
Code Q	Office of Safety and Mission Assurance (at NASA/Headquarters)
COTS	commercial-off-the-shelf
FY93	fiscal year 1993
GSFC	Goddard Space Flight Center
IRM	Information Resources Management
MSLOC	million source lines of code
NASA	National Aeronautics and Space Administration
SAIC	Science Applications International Corporation
SLOC	source lines of code

Profile of Software Engineering Within NASA

Craig C. Sinclair, SAIC
Kellyann Jeletic, NASA/GSFC

19th Annual Software Engineering Workshop
12/1/94

GOALS

Overall Goal:

- *Apply Software Engineering Laboratory (SEL) software process improvement approach to NASA as a whole*
- *Instill continual software process improvement throughout NASA*
- *Build specific recommendations for software process improvement within NASA*

Study Goal:

- *Establish the baseline of software and software engineering practices throughout NASA*

PURPOSE OF THE BASELINE

- To help NASA management understand the scope and extent of the software work within NASA

- To assist NASA management to see where they should focus future $$$ to improve software products and processes

- To assess the baseline for identification of candidate areas for software improvement

APPROACH

SEL Software Process Improvement Approach:

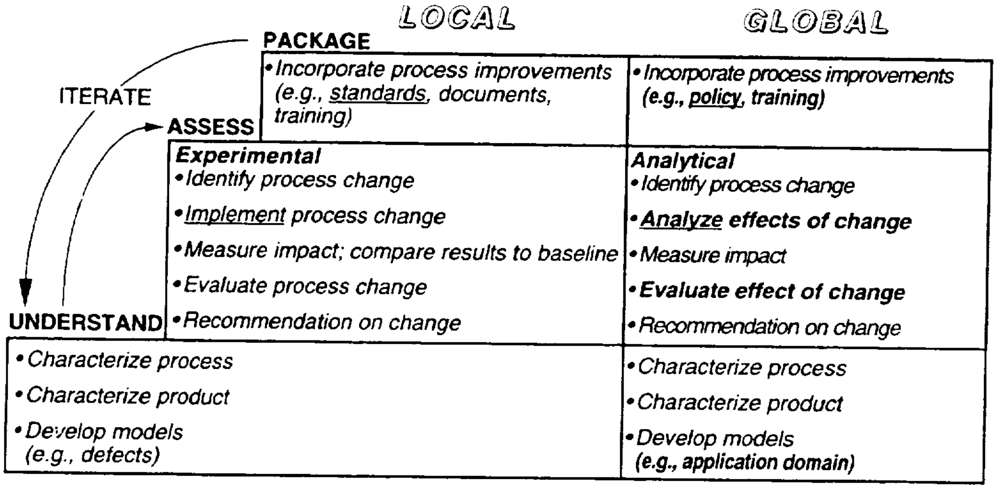
1) Understand (Baseline)

2) Assess

3) Package

There are some differences when applying the SEL approach to a ***global*** organizational level compared to a ***local*** organizational level

APPROACH - LOCAL VS. GLOBAL



	LOCAL	GLOBAL
PACKAGE	• *Incorporate process improvements (e.g., <u>standards</u>, documents, training)*	• *Incorporate process improvements* ***(e.g., <u>policy</u>, training)***
ASSESS	***Experimental*** • *Identify process change* • *<u>Implement</u> process change* • *Measure impact; compare results to baseline* • *Evaluate process change* • *Recommendation on change*	***Analytical*** • *Identify process change* • ***<u>Analyze</u> effects of change*** • *Measure impact* • ***Evaluate effect of change*** • *Recommendation on change*
UNDERSTAND	• *Characterize process* • *Characterize product* • *Develop models (e.g., defects)*	• *Characterize process* • *Characterize product* • *Develop models* ***(e.g., application domain)***

Local ---> Experimental
Global ---> Analytical

ESTABLISH THE BASELINE

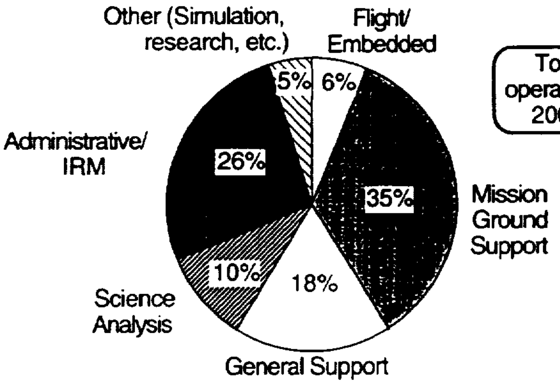
- Captured snapshot of FY93 attributes of:
  - NASA software (the product)
  - NASA's software engineering practices (the process)
- Data gathering methodology
  - Surveys, administered in person
  - Roundtable discussions
  - One on one interviews
  - Review of project documentation

Basic objective is to <u>understand</u>, not to judge right or wrong

- Next few charts describe the NASA Baseline
- Then we show how the baseline might be used

NASA SOFTWARE
PRODUCT CHARACTERISTICS

Amount of Software and Software Domains



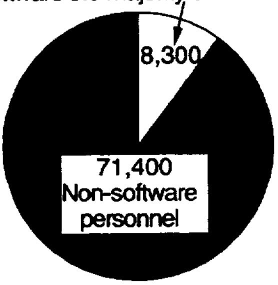
Total NASA operational code: 200 MSLOC

Largest software domains indicate where software improvement efforts could be focused

NASA SOFTWARE
PRODUCT CHARACTERISTICS

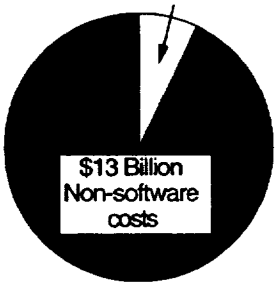
Software Staffing and Cost

More than 10% of NASA's 80,000 civil servants and support contractors were involved with software the majority of the time



About 80% of NASA's software work was contracted to industry

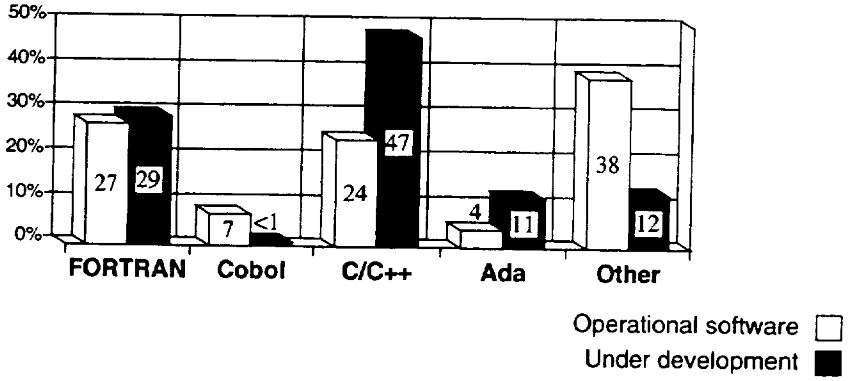
$1 Billion Software costs



NASA has a significant investment of manpower and budget in software

NASA SOFTWARE
PRODUCT CHARACTERISTICS

Language Preferences and Trends



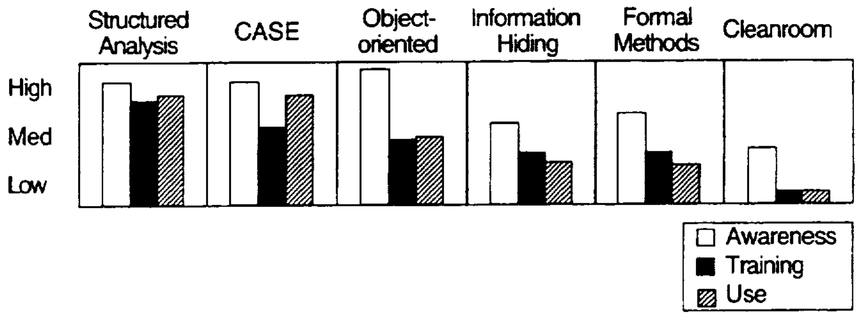
Findings may be used to focus training activities

NASA SOFTWARE
PROCESS CHARACTERISTICS

- ***Software Standards***
  - Project level were found to be most used and useful
  - Relative little usage by NASA personnel; widespread among contractors
- ***Requirements Management***
  - Unstable requirements are the biggest cause of schedule, budget, and code size growth problems
  - In general, requirements are not stable by preliminary design
- ***Software Reuse***
  - On average, about 15% of code is reused from one NASA project to another
  - Most focus is on code reuse; considerable variance in levels between Centers
- ***Software Metrics***
  - Little evidence of collection and use throughout NASA as a whole
  - Collection and use varied from non-existent to a few robust programs

NASA SOFTWARE *PROCESS CHARACTERISTICS*

Development Methodologies



NASA may be investing in "just in time" training

HOW BASELINE CAN BE USED

- To assess the baseline for **identification of candidate areas for software improvement**
- To **develop specific recommendations for implementation** of software improvement within NASA
- To **stimulate local implementation** of software improvement recommendations (bottom-up)

ASSESSING THE BASELINE *EXAMPLE 1*

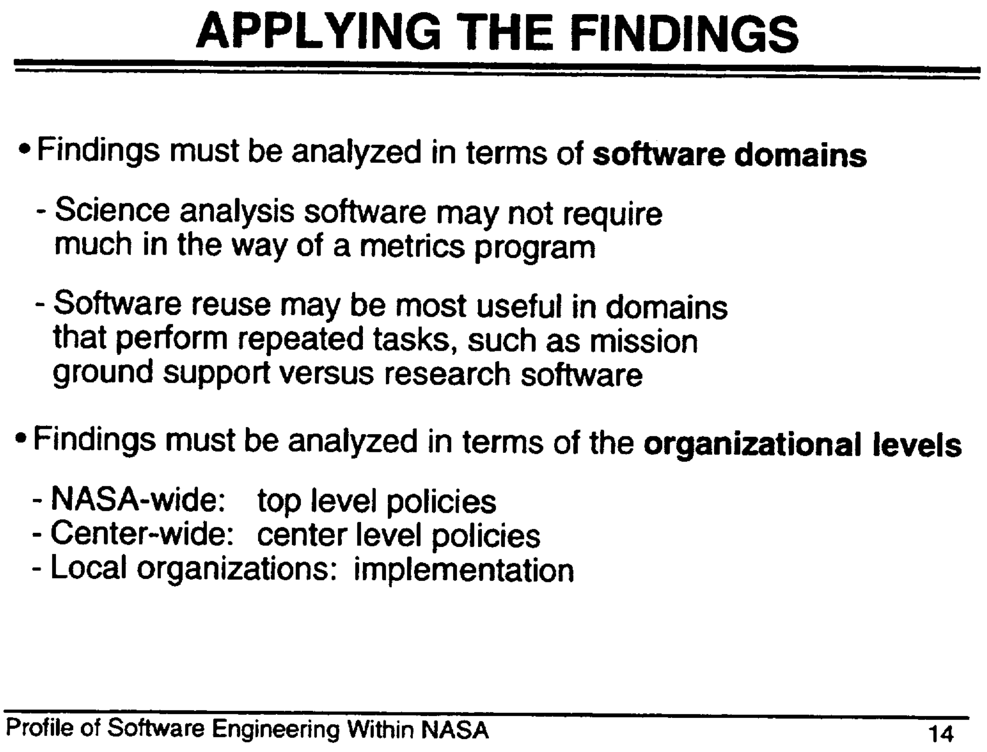
Measurement

- **Findings**
  - Collection and use varied from non-existent to a few robust programs
  - Little evidence of collection and use throughout NASA as a whole
- **Observations**
  - Software metrics need to be used for project management and to determine success of software improvement efforts
  - NASA could leverage the few robust metrics programs to assist the adoption of metrics by other NASA organizations

ASSESSING THE BASELINE *EXAMPLE 2*

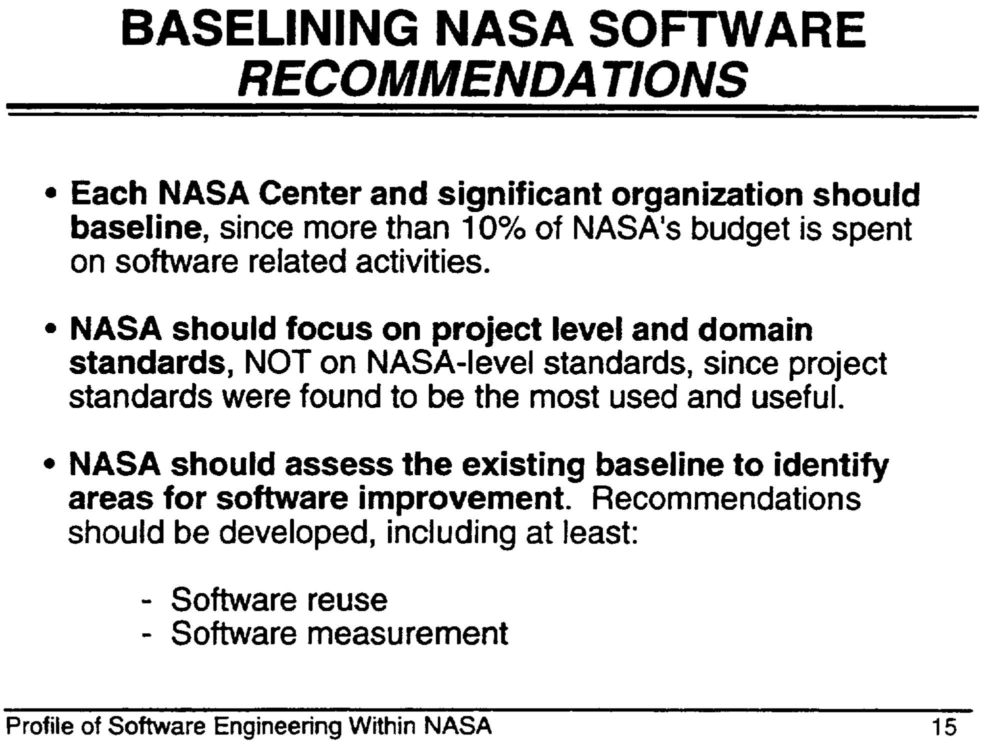
Reuse

- **Findings**
  - On average, about 15% of code is reused from one project to another
  - A few NASA organizations utilize software reuse as a normal part of their software development process
  - Overall, there were few defined approaches to reuse
- **Observations**
  - Organizations with software reuse (architecture and code) have made 75 - 80% reductions in cycle time and development cost
  - NASA could leverage the few robust programs to assist the adoption of software reuse by other NASA organizations



APPLYING THE FINDINGS

- Findings must be analyzed in terms of **software domains**
  - Science analysis software may not require much in the way of a metrics program
  - Software reuse may be most useful in domains that perform repeated tasks, such as mission ground support versus research software
- Findings must be analyzed in terms of the **organizational levels**
  - NASA-wide: top level policies
  - Center-wide: center level policies
  - Local organizations: implementation



BASELINING NASA SOFTWARE *RECOMMENDATIONS*

- **Each NASA Center and significant organization should baseline**, since more than 10% of NASA's budget is spent on software related activities.
- **NASA should focus on project level and domain standards**, NOT on NASA-level standards, since project standards were found to be the most used and useful.
- **NASA should assess the existing baseline to identify areas for software improvement.** Recommendations should be developed, including at least:
  - Software reuse
  - Software measurement

Appendix A: Workshop Attendees

[illegible]

Aalto, Juha-Markus, Nokia Telecommunications
Aaronson, Kenneth, Computer Sciences Corporation
Abrahamson, Shane, HQ AFC4A
Amad, Shahla, Hughes STX Corp.
Angevine, Jim, ALTA Systems, Inc.
Armentrout, Terry, Computer Sciences Corporation
Averill, Edward L., Edward Averill & Associates
Ayers, Everett, ARINC Research Corp.

Bailey, John, Software Metrics, Inc.
Barski, Renata M., AlliedSignal Technical Services Corp.
Basili, Victor R., University of Maryland
Bassman, Mitchell J., Computer Sciences Corporation
Beall, Shelley, Social Security Administration
Bean, Paul S., DSD Labs
Beierschmitt, Michael J., Loral AeroSys
Beifeld, David, Unisys Corp.
Bellamy, David, MITRE Corp.
Belvins, Brian E., Naval Surface Warfare Center
Bender, Jeff, University of Maryland
Beswick, Charlie A., Jet Propulsion Laboratory
Blagmon, Lowell E., Naval Center For Cost Analysis
Blankenship, Donald D., U.S. Air Force
Blatt, Carol, U.S. Census Bureau
Bobowiec, Paul W., COMPTEK Federal Systems
Bohn, Ilsa, SECON
Boland, Dillard, Computer Sciences Corporation
Boloix, Germinal, Ecole Polytechnique de Montreal
Bond, Jack, DoD
Borger, Mark W., Software Engineering Institute
Borkowski, Karen, SECON
Bowers, Allan, Loral AeroSys
Boyd, Andrew, U.S. Air Force
Bozoki, George J., Lockheed Missiles & Space Co., Inc.
Branson, Jim, American Systems Corp.
Bredeson, Mimi, Space Telescope Science Institute
Britt, Joan J., MITRE Corp.
Brown, Cindy, Computer Sciences Corporation
Brown, Richard A., Boeing Information Services
Bunch, Aleda, Social Security Administration
Burke, Karen, Loral Federal Systems
Burkhart, Michael R., IIT Research Institute
Burr, Carol, Computer Sciences Corporation
Busby, Mary, Loral Federal Systems Group
Buseman, Bill, EER Systems Corp.

Caldiera, Gianluigi, University of Maryland
Callahan, Jack, NASA/WVU IV&V Facility
Canfield, Roger A., DoD Joint Spectrum Center
Carlin, Catherine M., Veterans Benefits Administration
Carlisle, Candace, NASA/GSFC
Carlson, Randall, NSWCDD
Carmichael, Kevin R., NASA/LeRC
Carmody, Cora L., PRC, Inc.
Celentano, Al, Social Security Administration
Centafont, Noreen, DoD
Chapman, Robert, EG&G WASC, Inc.
Chen, Lily Y., AlliedSignal Technical Services Corp.
Chiverella, Ron, Pennsylvania Blue Shield
Chu, Richard, Loral AeroSys
Clamons, Jim, Harris Corp.
Clark, Carole A., Veterans Benefits Administration
Clark, James D., Naval Surface Warfare Center
Cochrane, Gail, TRW
Cohen, Judy, Jet Propulsion Laboratory
Condon, Steven E., Computer Sciences Corporation
Cooke, Terrence, Eagle Systems, Inc.
Coon, Richard W., Computer Sciences Corporation
Corbin, Regina, Social Security Administration
Cover, Donna, Computer Sciences Corporation
Cowan, Marcia, Loral AeroSys
Coyne, Edward J., SETA Corp.
Crawford, Art, Mantech Services Corp.
Crowder, Grace, University of Maryland-Baltimore Co.
Cuesta, Ernesto, Computer Sciences Corporation

Daku, Walter, Unisys Corp.
Daniele, Carl J., NASA/LeRC
Deutsch, Michael S., Hughes Applied Info. Systems, Inc.
Devasirvatham, Josiah, CARDS/D.N. American
DiNunno, Donn, Computer Sciences Corporation
Diaczun, Paul, Tidewater Consultants, Inc.
Dickson, Charles H., USDA
Diskin, David H., Defense Information Systems Agency
Dixson, Paul E., Loral AeroSys
Doland, Jerry T., Computer Sciences Corporation
Dolphin, Leonard, ALTA Systems, Inc.
Donnelly, Laurie M., AlliedSignal Technical Services Corp.
Dotseth, Margaret, Computer Sciences Corporation
Dowen, Andrew Z., Jet Propulsion Laboratory

Drake, Anthony M., Computer Sciences Corporation
DuBard, James E., Computer Sciences Corporation
Dudash, Ed, Naval Surface Warfare Center
Duncan, Scott P., BELLCORE
Dyer, Michael, DYCON Systems

Edelson, Robert, Jet Propulsion Laboratory
Eichmann, David, University of Houston-Clear Lake
Elliott, Geoffrey, Raytheon Engineers & Contractors
Ellis, Walter J., Software Process & Metrics
Esker, Linda J., Computer Sciences Corporation
Estep, James, WVHTC Foundation
Ettore de Cesare, Luciano, University of Maryland
Evans, Calvin Wayne, DoD

Farrell-Presley, Mary Ann, Applied Systems Technology, Inc.
Fedor, Gregory A., ADF, Inc.
Ferguson, Frances, Stanford Telecommunications, Inc.
Fernandes, Vernon, Computer Sciences Corporation
Finley, Doug, Unisys Corp.
Flora, Jackie, Social Security Administration
Flynn, Nick, Mantech Services Corp.
Forsythe, Ron, NASA/Wallops Flight Facility
Frahm, Mary J., Computer Sciences Corporation
Froehlich, Donna, IIT Research Institute
Futcher, Joseph M., Naval Surface Warfare Center

Gaddis, John B., CARDS/DSD Laboratories
Gallagher, Barbara F., DoD
Gallo, Al, Unisys Corp.
Gantzer, Don
Garlick, Teri, Pennsylvania Blue Shield
Getzen, Phil, DIA
Gill, David C., TECHSOFT, Inc.
Gluck, Raymond M., FCDSSA NSWC PHD ECO
Godfrey, Sally, NASA/GSFC
Golden, John R., Information Technologies
Goon, Linda, QA Consultant
Gosnell, Arthur B., U.S. Army Missile Command
Gover, Gary, Veterans Benefits Administration
Graffman, Ira, Hughes STX Corp.
Grano, Vince, Hughes STX Corp.
Grant, Jr., Ralph D., New Technology, Inc.
Grech, Neill, Computer Sciences Corporation
Green, David S., Computer Sciences Corporation
Green, Leonard, DoD
Green, Scott, NASA/GSFC
Greene, James S., U.S. Air Force
Gregory, Judith A., NASA/MSFC
Gregory, Shawna C., MITRE Corp.
Griffin, Brenda, McDonnell Douglas Space Systems Co.
Gwennet, Lance, Systems Research & Application Corp.
Gwynn, Thomas R., Computer Sciences Corporation

Haislip, William, U.S. Army
Hall, Angela, Computer Sciences Corporation
Hall, Charley, SECON
Hall, Dana L., SAIC
Hall, Ken R., Computer Sciences Corporation
Han, Cecilia, Jet Propulsion Laboratory
Hannon, Nanci, DISA/JIEQ/TBEC
Hanson, Pauline, U.S. Census Bureau
Harris, Barbara A., USDA
Hartzler, Ellen
Heintzelman, Clinton L., U.S. Air Force
Heller, Gerard H., Computer Sciences Corporation
Hendrzak, Gary, Booz, Allen & Hamilton, Inc.
Henry, Joel, East Tennessee State University
Herbsleb, Jim, Software Engineering Institute
Heuser, Wolfgang, Daimlerbenz, Research
Higgins, Herman A., DoD
Hoffmann, Kenneth, Ryan Computer Systems, Inc.
Holloway, Mel, Joint Warfare Analysis Center
Holmes, Barbara, CRM
Holmes, Joseph A., IRS
Hopkins, Clifford R., DoD
Hoppe, William C., U.S. Army/ISSC
Howard, William H., Unisys Corp.
Hultgren, Ed, DoD
Hung, Chaw-Kwei, Jet Propulsion Laboratory

Ingram, Darryl R., Maryland Tectrix, Inc.

Jefferson, Pat, IRS
Jeletic, Jim, NASA/GSFC
Jeletic, Kellyann, NASA/GSFC
Johnson, Pat, NASA/GSFC
Johnson, Temp, Hughes-STX
Jones, Christopher C., IIT Research Institute
Jones, Linda J., TRW
Jones, Lori M., Tidewater Consultants, Inc.
Jordan, Gary, Unisys Corp.
Jordano, Tony J., SAIC

Kalin, Jay, Loral AeroSys
Karlsson, Even-Andre, Q-Labs
Keeler, Kristi L., Computer Sciences Corporation
Kelly, John C., Jet Propulsion Laboratory
Kemp, Kathryn M., Office of Safety & Mission Assurance

Kester, Rush W., DC SIGAda
Kierk, Isabella K., NASA/JPL
Kim, Robert D., Computer Sciences Corporation
Kim, Yong-Mi, University of Maryland
Kistler, David M., Computer Sciences Corporation
Klein, Jr., Gerald A., QSS Group, Inc.
Kleptin, Laurie, Unisys Corp.
Knight, Colette A., NSWC
Knight, John C., University of Virginia
Kontio, Jyrki, University of Maryland
Kotov, Alexei, Oregon Graduate Institute
Kraft, Steve, NASA/GSFC
Kronisch, Mark, U.S. Census Bureau
Kuhle, Sherry, Booz, Allen & Hamilton
Kuhne, Fran, Social Security Administration
Kushner, Todd R., CTA, Inc.
Kyser, Frances, Nichols Research Corp.

LaPorte, Claude Y., Oerlikon Aerospace
Laitenberger, Oliver
Lamia, Walter, Software Engineering Institute
Landis, Linda C., Computer Sciences Corporation
Lane, Allan C., AlliedSignal Technical Services Corp.
Langston, James H., Computer Sciences Corporation
Lawrence, Raymond L., Loral AeroSys
Lay, Barbara N., Motorola, Inc.
Lehman, Meir M., Imperial College of Science
Levesque, Michael, Jet Propulsion Laboratory
Levinson, Laurie H., NASA/LeRC
Levitt, Dave S., Computer Sciences Corporation
Leydorf, Steven M., IIT Research Institute
Li, Rorry, ORACLE
Libson, Ted, Boeing Information Services, Inc.
Liebermann, Roxanne, U.S. Census Bureau
Liebrecht, Paula L., Computer Sciences Corporation
Lindsay, Scott, Government Systems, Inc.
Lindsey, Brad, IIT Research Institute
Lipsett, Bill, IRS
Liu, Jean C., Computer Sciences Corporation
Livingston, Karen, IIT Research Institute
Loesh, Bob E., Software Engineering Sciences Corporation
Lott, Christopher M., University of Kaiserlautern
Loy, Patrick H., Loy Consulting, Inc.
Lubash, Steven, ITT Avionics
Lucas, Janice P., Financial Management Service
Luczak, Ray W., Computer Sciences Corporation
Lutz, Robyn R., Iowa State University
Luu, Kequan, NASA/GSFC
Lynnes, Chris, NASA/GSFC
Lyons, Howarette P., AFPCA/GADB

Mabry, Bobbie L., DoD
Mahmud, Maruf, IRS
Marciniak, John J., Kaman Sciences Corporation
Marijarvi, Jukka, Nokia Cellular Systems
Martinez, Bill, Loral Federal Systems
Masters, Wen C., Jet Propulsion Laboratory
Maury, Jesse, Omitron, Inc.
Mazzola, Ray, Loral AeroSys
McCafferty, Brian, XonTech, Inc.
McConnell, Andrew, ASSET
McCreary, Faith A., Jet Propulsion Laboratory
McGarry, Frank E., Computer Sciences Corporation
McGrane, Janet K., U.S. Census Bureau
McHenry, Ron, Hughes STX Corp.
McIlwraith, Isabel, IRS
McKay, Judith A., U.S. Census Bureau
McNeill, Justin F., Jet Propulsion Laboratory
McSharry, Maureen, Computer Sciences Corporation
Melo, Walcelio L., University of Maryland
Mendonca, Manoel G., University of Maryland
Methia, Linda, NASA/HQ
Miller, Dave, COMSO, Inc.
Mills, Marilyn K., Computer Sciences Corporation
Minninger, John R., DoD
Moleski, Laura, CRM
Moore, Paula, NOAA/SPOx3
Moore, Robin W., Air Force Pentagon Communication Agency
Morgan, Pam, IIT Research Institute
Morusiewicz, Linda M., Computer Sciences Corporation
Moseley, Patricia, DoD
Murphy, Thomas, Siemans Corporate Research, Inc.
Myers, Philip I., Computer Sciences Corporation

Narrow, Bernie, AlliedSignal Technical Services Corp.
Neilan, Hester, Jet Propulsion Laboratory
Nestlerode, Howard, Unisys Corp.
Neuman, Harriet J., FAA
New, Ronald, NOAA
Nichols, Dan, CARDS/EWA
Nickerson, Brenda, Loral Federal Systems Group
Niemela, Mary, SECON
Nokovich, Sandra L., U.S. Census Bureau
Norcio, Tony F., University of Maryland-Baltimore Co.
Norton, William F., FAA
Nusenoff, Ron, Loral Software Productivity Lab

O'Brien, Robert L., Unisys Corp.
O'Donnell, Charlie, ECA, Inc.
O'Neill, Don, Software Engineering Consultant
Obenski, Dave, DoD
Offer, Regina W., AFPCA/6ADB
Ohlmacher, Jane A., Social Security Administration

Page, Gerald T., Computer Sciences Corporation
Pailen, William, Pailen-Johnson Associates, Inc.
Pajerski, Rose, NASA/GSFC
Paletar, Teresa L., Naval Air Warfare Center
Panlilio-Yap, Nikki M., Loral Federal Systems Group
Parker-Gates, Linda, Software Productivity Consortium
Passaretti, Gennaro, Consorzio SOFTIN
Patton, K. Kay, Computer Sciences Corporation
Pavnica, Paul, Treasury/Fincen
Peeples, Ron L., Intermetrics
Perry, Howard, Computer Sciences Corporation
Peterman, David, Texas Instruments
Petro, Jim, EWA, Inc.
Pettengill, Nathan, Martin Marietta
Phan, Quyen, BTG, Inc.
Polk, Bryant, Systems Research & Applications Corp.
Porter, Adam A., University of Maryland
Pottinger, David L., SAIC
Powers, Larry T., Unisys Corp.
Pressley, Coretta T., DoD
Provenza, Clint, Booz, Allen & Hamilton, Inc.

Quann, Eileen S., Fastrak Training, Inc.

Radley, Charles, Raytheon - EBASCO
Ramsey, Jack, Pennsylvania Blue Shield
Raney, Dale L., TRW
Redding, John L., Defense Information Systems Agency
Reeb, Jim, U.S. Army MICOM
Reed, James J., Karman Sciences Corporation
Regardie, Myrna L., Computer Sciences Corporation
Reitzel, Morris, DoD
Rifkin, Stan, Master Systems, Inc.
Riihinen, Jaakko, Nokia Cellular Systems
Risser, Gary E., Veterans Benefits Administration
Rizer, Stephani, NAWC-AD
Rodenas, Albe, ALTA Systems, Inc.
Rohr, John A., Jet Propulsion Laboratory
Rohrer, Amos M., SYSCON Corp.
Rosenberg, Linda H., Unisys Corp.
Roy, Dan M., STP&P
Russell, Wayne M., GTE
Rymer, John, Loral Federal Systems Group

Sabolish, George J., NASA IV&V Facility
Saisi, Robert O., DSD Laboratories, Inc.
Samadi, Shahin, NASA/GSFC
Samuels, George, Social Security Administration
Santiago, Richard, Jet Propulsion Laboratory
Satyen, Uma D., MITRE Corp.
Sawanobori, Tina, Computer Sciences Corporation
Schellhase, Ronald J., Computer Sciences Corporation
Schilling, Mark, Informatics, Inc.
Schmidt, Evan, EWA
Schrom, Roberta, Hughes Training, Inc.
Schuler, Pat M., NASA/LaRC
Schwartz, Benjamin L., Consultant
Schwarz, Henry, NASA/KSC
Scott, Donna, PRC, Inc.
Scott, Rhonda M., Loral Federal Systems Group
Seaman, Carolyn B., University of Maryland
Seiber, Dwayne, OAO Corp.
Seidewitz, Ed, NASA/GSFC
Sharma, Jagdish, NOAA
Sharma, Khem, Computer Sciences Corporation
Sheckler, John D., AlliedSignal Technical Services Corp.
Sheppard, Sylvia B., NASA/GSFC
Shi-Hung Hsueh, Bryan, University of Maryland
Short, Cathy, IRS
Singer, Carl A., BELLCORE
Six, Richard E., DoD
Siy, Harvey, University of Maryland
Slaton, Gordon, U.S. Air Force
Slonim, Jacob, Centre for Advanced Studies - IBM
Smith, Donald, NASA/GSFC
Smith, George F., Space & Naval Warfare Systems Command
Smith, Sharon, Loral Federal Systems Group
Smith, Vivian A., FAA
Sohmer, Robert, RAM Engineering Associates
Solomon, Carl A., Hughes-STX
Sova, Don, NASA/HQ
Squire, Jon S., Westinghouse
Squires, Burton E., Orion Scientific, Inc.
Ssemwogerere, Joe, Hughes STX Corp.
Stamboulis, Peter, SECON
Stang, David, ADF, Inc.
Stark, Michael, NASA/GSFC
Staub, Jane B., Tidewater Consultants, Inc.
Steger, Warren L., Computer Sciences Corporation
Steinberg, Sandee, Computer Sciences Corporation
Stoddard, Robert W., Texas Instruments

Summa, Robert L., Computer Sciences Corporation
Suryani Jamin Tung, Angela, AlliedSignal Technical Services Corp.
Szulewski, Paul A., C.S. Draper Labs, Inc.

Tasaki, Keiji, NASA/GSFC
Tesoriero, Roseanne, University of Maryland
Thomas, William, MITRE Corp.
Thomason, Clarke, Pailen-Johnson Associates, Inc.
Thompson, Sid, Unisys Corp.
Thornton, Thomas H., Jet Propulsion Laboratory
Tichenor, Charley, IRS
Trammell, Carmen J., University of Tennessee
Trible, Sue, GDE Systems, Inc.
Turcheck, Susan, Loral Federal Systems
Twombly, Mark A., Computer Based Systems, Inc.

Ulery, Bradford T., MITRE Corp.

Valdivia, Aaron, IIT Research Institute
Valett, Jon, NASA/GSFC
Van Verth, Patricia B., Canisius College
VanMeter, Charlene L., DoD
Vaughan, Frank R., RAM Engineering Associates
Vause, David G., Loral Federal Systems Group
Vernacchio, Al, NASA/GSFC
Viehman, Ralph, NASA/GSFC
Voigt, David, AlliedSignal Technical Services Corp.

Wald, Lawrence, NASA/LeRC
Waligora, Sharon R., Computer Sciences Corporation
Wallace, Dolores, NIST
Waterman, Robert E., Unisys Corp.
Webb, Hattie R., Naval Surface Warfare Center
Weiss, Sandy L., IIT Research Institute
Wenneson, Gregory J., Sterling Software, Inc.
West, Tim, DIA
Weston, William N., NASA/GSFC
Weszka, Joan, Loral Federal Systems Group
Wetherholt, Martha S., NASA/LeRC
Whisenand, Tom, Social Security Administration
White, Gilbert, NASA/HQ
Whittlesey, Raquel S., ADF, Inc.
Wiggers, Grace E., Computer Sciences Corporation
Wika, Kevin G., University of Virginia
Wilkins, Lori H., New-Bold Enterprises
Willey, Allan L., Motorola, Inc.
Williams, Bonnie, Computer Sciences Corporation
Wilson, Robert K., Jet Propulsion Laboratory
Wingfield, Charles G., DoD
Wolfish, Hannah K., Social Security Administration
Wood, Dick, Computer Sciences Corporation
Wu, Sabina L., IIT Research Institute

Yakstis, Louis C., Computer Sciences Corporation
Yassini, Siamak, NASA/HQ
York, Lee, Veterans Benefits Administration
Youman, Charles, SETA Corp.
Young, Andy, Young Engineering Services, Inc.
Young, Jeff, Lawrence Livermore National Lab
Yu, Phil, TRW

Zaveler, Saul, U.S. Air Force
Zeitvogel, Barney, SECON
Zelkowitz, Marv, University of Maryland
Zimet, Beth, Computer Sciences Corporation

Appendix B: Standard Bibliography of SEL Literature

STANDARD BIBLIOGRAPHY OF SEL LITERATURE

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

SEL-ORIGINATED DOCUMENTS

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

SEL-81-305SP1, *Ada Developers' Supplement to the Recommended Approach*, R. Kester and L. Landis, November 1993

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

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

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

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

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

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

SEL-82-1306, *Annotated Bibliography of Software Engineering Laboratory Literature*, D. Kistler, J. Bristow, and D. Smith, November 1994

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

SEL-89-301, *Software Engineering Laboratory (SEL) Database Organization and User's Guide (Revision 3)*, L. Morusiewicz, February 1995

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

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

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

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

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

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

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

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

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

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

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

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

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

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

SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992

SEL-92-004, *Proceedings of the Seventeenth Annual Software Engineering Workshop*, December 1992

SEL-93-001, *Collected Software Engineering Papers: Volume XI*, November 1993

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

SEL-93-003, *Proceedings of the Eighteenth Annual Software Engineering Workshop*, December 1993

SEL-94-001, *Software Management Environment (SME) Components and Algorithms*, R. Hendrick, D. Kistler, and J. Valett, February 1994

SEL-94-002, *Software Measurement Guidebook*, M. Bassman, F. McGarry, R. Pajerski, July 1994

SEL-94-003, *C Style Guide*, J. Doland and J. Valett, August 1994

SEL-94-004, *Collected Software Engineering Papers: Volume XII*, November 1994

SEL-94-005, *An Overview of the Software Engineering Laboratory*, F. McGarry, G. Page, V. Basili, et al., December 1994

SEL-94-006, *Proceedings of the Nineteenth Annual Software Engineering Workshop*, December 1994

SEL-RELATED LITERATURE

[10]Abd-El-Hafiz, S. K., V. R. Basili, and G. Caldiera, "Towards Automated Support for Extraction of Reusable Components," *Proceedings of the IEEE Conference on Software Maintenance-1991 (CSM 91)*, October 1991

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

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

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

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

[10]Bailey, J. W., and V. R. Basili, "The Software-Cycle Model for Re-Engineering and Reuse," *Proceedings of the ACM Tri-Ada 91 Conference*, October 1991

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

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

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

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

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

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

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

[9]Basili, V. R., G. Caldiera, and G. Cantone, "A Reference Architecture for the Component Factory,"*ACM Transactions on Software Engineering and Methodology*, January 1992

[10]Basili, V., G. Caldiera, F. McGarry, et al., "The Software Engineering Laboratory—An Operational Software Experience Factory," *Proceedings of the Fourteenth International Conference on Software Engineering (ICSE 92)*, May 1992

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

[12]Basili, V., and S. Green, "Software Process Evolution at the SEL," *IEEE Software*, July 1994

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

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

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

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

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

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

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

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

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

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

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

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

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

[9]Basili, V. R., and H. D. Rombach, "Support for Comprehensive Reuse," *Software Engineering Journal*, September 1991

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Bassman, M. J., F. McGarry, and R. Pajerski, *Software Measurement Guidebook*, NASA-GB-001-94, Software Engineering Program, July 1994

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

[10]Booth, E. W., and M. E. Stark, "Software Engineering Laboratory Ada Performance Study—Results and Implications," *Proceedings of the Fourth Annual NASA Ada User's Symposium*, April 1992

[10]Briand, L. C., and V. R. Basili, "A Classification Procedure for the Effective Management of Changes During the Maintenance Process," *Proceedings of the 1992 IEEE Conference on Software Maintenance (CSM 92)*, November 1992

[10]Briand, L. C., V. R. Basili, and C. J. Hetmanski, "Providing an Empirical Basis for Optimizing the Verification and Testing Phases of Software Development," *Proceedings of the Third IEEE International Symposium on Software Reliability Engineering (ISSRE 92)*, October 1992

[11]Briand, L. C., V. R. Basili, and C. J. Hetmanski, *Developing Interpretable Models with Optimized Set Reduction for Identifying High Risk Software Components*, University of Maryland, Technical Report TR-3048, March 1993

[12]Briand, L. C., V. R. Basili, Y. Kim, and D. R. Squire, "A Change Analysis Process to Characterize Software Maintenance Projects", *Proceedings of the International Conference on Software Maintenance*, September 1994

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

[11]Briand, L. C., S. Morasca, and V. R. Basili, "Measuring and Assessing Maintainability at the End of High Level Design," *Proceedings of the 1993 IEEE Conference on Software Maintenance (CSM 93)*, November 1993

[12]Briand, L., S. Morasca, and V. R. Basili, *Defining and Validationg High-Level Design Metrics*, University of Maryland, Technical Report TR-3301, June 1994

[11]Briand, L. C., W. M. Thomas, and C. J. Hetmanski, "Modeling and Managing Risk Early in Software Development," *Proceedings of the Fifteenth International Conference on Software Engineering (ICSE 93)*, May 1993

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

[11]Li, N. R., and M. V. Zelkowitz, "An Information Model for Use in Software Management Estimation and Prediction," *Proceedings of the Second International Conference on Information Knowledge Management*, November 1993

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

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

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

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

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

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

[12]Porter, A. A., L. G. Votta, Jr., and V. R. Basili, *Comparing Detection Methods for Software Requirements Inspections: A Replicated Experiment*, University of Maryland, Technical Report TR-3327, July 1994

[5]Ramsey, C. L., and V. R. Basili, "An Evaluation of Expert Systems for Software Engineering Management," *IEEE Transactions on Software Engineering*, June 1989

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

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

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

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

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

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

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

[10]Rombach, H. D., B. T. Ulery, and J. D. Valett, "Toward Full Life Cycle Control: Adding Maintenance Measurement to the SEL," *Journal of Systems and Software*, May 1992

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

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

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

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

[10]Seidewitz, E., "Object-Oriented Programming With Mixins in Ada," *Ada Letters*, March/April 1992

[12]Seidewitz, E., "Genericity versus Inheritance Reconsidered: Self-Reference Using Generics," *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications*, October 1994

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

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

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

[11]Stark, M., "Impacts of Object-Oriented Technologies: Seven Years of SEL Studies," *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications, September 1993*

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

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

[10]Straub, P. A., and M. V. Zelkowitz, "On the Nature of Bias and Defects in the Software Specification Process," *Proceedings of the Sixteenth International Computer Software and Applications Conference (COMPSAC 92)*, September 1992

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

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

[10]Tian, J., A. Porter, and M. V. Zelkowitz, "An Improved Classification Tree Analysis of High Cost Modules Based Upon an Axiomatic Definition of Complexity," *Proceedings of the Third IEEE International Symposium on Software Reliability Engineering (ISSRE 92)*, October 1992

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

[10]Valett, J. D., "Automated Support for Experience-Based Software Management," *Proceedings of the Second Irvine Software Symposium (ISS _92)*, March 1992

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

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

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

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

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

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

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

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

NOTES:

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

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

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

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

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

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

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

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

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

[10] This article also appears in SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992.

[11] This article also appears in SEL-93-001, *Collected Software Engineering Papers: Volume XI*, November 1993.

[12] This article also appears in SEL-94-004, *Collected Software Engineering Papers: Volume XII*, November 1994.

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
December 1994

3. REPORT TYPE AND DATES COVERED
Contractor Report

4. TITLE AND SUBTITLE
Proceedings of the Nineteenth Annual Software Engineering Workshop

5. FUNDING NUMBERS
552

6. AUTHOR(S)
Software Engineering Laboratory

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Software Engineering Branch
Code 552
Goddard Space Flight Center
Greenbelt, Maryland

8. PERFORMING ORGANIZATION REPORT NUMBER
SEL-94-006

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
NASA Aeronautics and Space Administration
Washington, D.C. 20546-0001

10. SPONSORING/MONITORING AGENCY REPORT NUMBER
CR-189411

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Unclassified-Unlimited
Subject Category: 61
Report is available from the NASA Center for AeroSpace Information,
800 Elkridge Landing Road, Linthicum Heights, MD 21090; (301) 621-0390.

12b. DISTRIBUTION CODE

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

The Software Engineering Laboratory (SEL) is an organization sponsored by NASA/GSFC and created to investigatethe effectiveness of software engineering technologies when applied to the development of applications software. The goals of the SEL are (1) to understand the software development process in the GSFC environment; (2) to measure the effects of various methodologies, tools, and models on this process; and (3) to identify and then to apply successful development practices. The activities, findings, and recommendations of the SEL are recorded in the Software Engineering Laboratory Series, a continuing series of reports that includes this document.

14. SUBJECT TERMS
Software Engineering, Software Measurement, Data Collection

15. NUMBER OF PAGES
357

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT
Unclassified

18. SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. SECURITY CLASSIFICATION OF ABSTRACT
Unclassified

20. LIMITATION OF ABSTRACT
Unlimited