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.
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.

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 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.

1Data 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.
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.
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.

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](image)

**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


ACRONYMS AND ABBREVIATIONS

CASE
Code 500
Code Q
COTS
FY93
GSFC
IRM
MSLOC
NASA
SAIC
SLOC

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

<table>
<thead>
<tr>
<th>LOCAL</th>
<th>GLOBAL</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>PACKAGE</strong></td>
<td></td>
</tr>
<tr>
<td>Incorporate process improvements</td>
<td>Incorporate process improvements (e.g., policy, training)</td>
</tr>
<tr>
<td>(e.g., standards, documents, training)</td>
<td></td>
</tr>
<tr>
<td><strong>ASSESS</strong></td>
<td></td>
</tr>
<tr>
<td>Experimental</td>
<td>Analytical</td>
</tr>
<tr>
<td>Identify process change</td>
<td>Identify process change</td>
</tr>
<tr>
<td>Implement process change</td>
<td></td>
</tr>
<tr>
<td>Measure impact; compare results to baseline</td>
<td>Measure impact</td>
</tr>
<tr>
<td>Evaluate process change</td>
<td>Evaluate effect of change</td>
</tr>
<tr>
<td>Recommendation on change</td>
<td>Recommendation on change</td>
</tr>
<tr>
<td><strong>UNDERSTAND</strong></td>
<td></td>
</tr>
<tr>
<td>Characterize process</td>
<td>Characterize process</td>
</tr>
<tr>
<td>Characterize product</td>
<td>Characterize product</td>
</tr>
<tr>
<td>Develop models (e.g., defects)</td>
<td>Develop models (e.g., application domain)</td>
</tr>
</tbody>
</table>

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

Profile of Software Engineering Within NASA

## 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 understand, 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

- Other (Simulation, research, etc.) 15%
- Flight/Embedded 6%
- Administrative/IRM 26%
- Science Analysis 10%
- General Support 18%
- Mission Ground Support 35%

Largest software domains indicate where software improvement efforts could be focused.

Profile of Software Engineering Within NASA

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:
  - 71,400 Non-software personnel
  - 8,300 Software

- $1 Billion Software costs
- $13 Billion Non-software costs

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

NASA has a significant investment of manpower and budget in software.

Profile of Software Engineering Within NASA
Language Preferences and Trends

![Bar chart showing language preferences and trends](chart)

Findings may be used to focus training activities

Profile of Software Engineering Within NASA

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

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

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
<table>
<thead>
<tr>
<th>Name</th>
<th>Organization</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aalto, Juha-Markus</td>
<td>Nokia Telecommunications</td>
</tr>
<tr>
<td>Aaronson, Kenneth</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>Abrahamson, Shane</td>
<td>HQ AFC4A</td>
</tr>
<tr>
<td>Amad, Shahla</td>
<td>Hughes STX Corp.</td>
</tr>
<tr>
<td>Angervine, Jim</td>
<td>ALTA Systems, Inc.</td>
</tr>
<tr>
<td>Armentrout, Terry</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>Averill, Edward L.</td>
<td>Edward &amp; Associates</td>
</tr>
<tr>
<td>Ayers, Everett</td>
<td>ARINC Research Corp.</td>
</tr>
<tr>
<td>Bailey, John</td>
<td>Software Metrics, Inc.</td>
</tr>
<tr>
<td>Barski, Renata M.</td>
<td>AlliedSignal Technical Services Corp.</td>
</tr>
<tr>
<td>Basil, Victor R.</td>
<td>University of Maryland</td>
</tr>
<tr>
<td>Bassman, Mitchell J.</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>Beall, Shelley</td>
<td>Social Security Administration</td>
</tr>
<tr>
<td>Bean, Paul S.</td>
<td>DSD Labs</td>
</tr>
<tr>
<td>Beierschmitt, Michael J.</td>
<td>Loral AeroSys</td>
</tr>
<tr>
<td>Beifeld, David</td>
<td>Unisys Corp.</td>
</tr>
<tr>
<td>Bellamy, David</td>
<td>MITRE Corp.</td>
</tr>
<tr>
<td>Belvins, Brian E.</td>
<td>Naval Surface Warfare Center</td>
</tr>
<tr>
<td>Bender, Jeff</td>
<td>University of Maryland</td>
</tr>
<tr>
<td>Beswick, Charlie A.</td>
<td>Jet Propulsion Laboratory</td>
</tr>
<tr>
<td>Blagmon, Lowell E.</td>
<td>Naval Center For Cost Analysis</td>
</tr>
<tr>
<td>Blankenship, Donald D.</td>
<td>U.S. Air Force</td>
</tr>
<tr>
<td>Blatt, Carol</td>
<td>U.S. Census Bureau</td>
</tr>
<tr>
<td>Bohn, Ilsa</td>
<td>SECON</td>
</tr>
<tr>
<td>Boland, Dillard</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>Boloix, Germainal</td>
<td>Ecole Polytechnique de Montreal</td>
</tr>
<tr>
<td>Bond, Jack</td>
<td>DoD</td>
</tr>
<tr>
<td>Borger, Mark W.</td>
<td>Software Engineering Institute</td>
</tr>
<tr>
<td>Borkowski, Karen</td>
<td>SECON</td>
</tr>
<tr>
<td>Bowers, Allan</td>
<td>Loral AeroSys</td>
</tr>
<tr>
<td>Boyd, Andrew</td>
<td>U.S. Air Force</td>
</tr>
<tr>
<td>Bozoki, George J.</td>
<td>Lockheed Missiles &amp; Space Co., Inc.</td>
</tr>
<tr>
<td>Branson, Jim</td>
<td>American Systems Corp.</td>
</tr>
<tr>
<td>Bredeson, Mimi</td>
<td>Space Telescope Science Institute</td>
</tr>
<tr>
<td>Britt, Joan J.</td>
<td>MITRE Corp.</td>
</tr>
<tr>
<td>Brown, Cindy</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>Brown, Richard A.</td>
<td>Boeing Information Services</td>
</tr>
<tr>
<td>Bunch, Aleida</td>
<td>Social Security Administration</td>
</tr>
<tr>
<td>Burke, Karen</td>
<td>Loral Federal Systems</td>
</tr>
<tr>
<td>Burkhart, Michael R.</td>
<td>IIT Research Institute</td>
</tr>
<tr>
<td>Burr, Carol</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>Busby, Mary</td>
<td>Loral Federal Systems Group</td>
</tr>
<tr>
<td>Buseman, Bill</td>
<td>EER Systems Corp.</td>
</tr>
<tr>
<td>Caldierna, Gianluigi</td>
<td>University of Maryland</td>
</tr>
<tr>
<td>Callahan, Jack</td>
<td>NASA/WVU IV&amp;V Facility</td>
</tr>
<tr>
<td>Canfield, Roger A.</td>
<td>DoD Joint Spectrum Center</td>
</tr>
<tr>
<td>Carlin, Catherine M.</td>
<td>Veterans Benefits Administration</td>
</tr>
<tr>
<td>Carlisle, Candace</td>
<td>NASA/GSFC</td>
</tr>
<tr>
<td>Carlson, Randall</td>
<td>NSWCDD</td>
</tr>
<tr>
<td>Carnichael, Kevin R.</td>
<td>NASA/LeRC</td>
</tr>
<tr>
<td>Carmody, Cora L.</td>
<td>PRC, Inc.</td>
</tr>
<tr>
<td>Celentano, Al</td>
<td>Social Security Administration</td>
</tr>
<tr>
<td>Centafoot, Noreen</td>
<td>DoD</td>
</tr>
<tr>
<td>Chapman, Robert</td>
<td>EG&amp;G WASC, Inc.</td>
</tr>
<tr>
<td>Chen, Lily Y.</td>
<td>AlliedSignal Technical Services Corp.</td>
</tr>
<tr>
<td>Chiverella, Ron</td>
<td>Pennsylvania Blue Shield</td>
</tr>
<tr>
<td>Chu, Richard</td>
<td>Loral AeroSys</td>
</tr>
<tr>
<td>Clamons, Jim</td>
<td>Harris Corp.</td>
</tr>
<tr>
<td>Clark, Carole A.</td>
<td>Veterans Benefits Administration</td>
</tr>
<tr>
<td>Clark, James D.</td>
<td>Naval Surface Warfare Center</td>
</tr>
<tr>
<td>Cochrane, Gail</td>
<td>TRW</td>
</tr>
<tr>
<td>Cohen, Judy</td>
<td>Jet Propulsion Laboratory</td>
</tr>
<tr>
<td>Condon, Steven E.</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>Cooke, Terrence</td>
<td>Eagle Systems, Inc.</td>
</tr>
<tr>
<td>Coon, Richard W.</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>Corbin, Regina</td>
<td>Social Security Administration</td>
</tr>
<tr>
<td>Crowder, Grace</td>
<td>University of Maryland-Baltimore Co.</td>
</tr>
<tr>
<td>Cuesta, Ernesto</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>Cowan, Marcia</td>
<td>Loral AeroSys</td>
</tr>
<tr>
<td>Coyne, Edward J.</td>
<td>SETA Corp.</td>
</tr>
<tr>
<td>Daniele, Carl J.</td>
<td>NASA/LeRC</td>
</tr>
<tr>
<td>DiNunno, Donn</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>Diaczun, Paul</td>
<td>Tidewater Consultants, Inc.</td>
</tr>
<tr>
<td>Dickson, Charles H.</td>
<td>USDAA</td>
</tr>
<tr>
<td>Diskin, David H.</td>
<td>Defense Information Systems Agency</td>
</tr>
<tr>
<td>Dixon, Paul E.</td>
<td>Loral AeroSys</td>
</tr>
<tr>
<td>Doland, Jerry T.</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>Dolphin, Leonard</td>
<td>ALTA Systems, Inc.</td>
</tr>
<tr>
<td>Donnelly, Laurie M.</td>
<td>AlliedSignal Technical Services Corp.</td>
</tr>
<tr>
<td>Dotseh, Margaret</td>
<td>Computer Sciences Corporation</td>
</tr>
<tr>
<td>Downing, Andrew Z.</td>
<td>Jet Propulsion Laboratory</td>
</tr>
</tbody>
</table>

**SEW Proceedings**

335

**SEL-94-006**
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
Elliot, 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, ITT 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, Neil, 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, Shawn 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
Hendrak, 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
Karlssson, Even-Andre, Q-Labs
Keeler, Kristi L., Computer Sciences Corporation
Kelly, John C., Jet Propulsion Laboratory
Kemp, Kathryn M., Office of Safety & Mission Assurance

SEW Proceedings

336

SEL-94-006
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-78-007, Applicability of the Rayleigh Curve to the SEL Environment, T. E. Mapp, December 1978
SEL-79-005, Proceedings From the Fourth Summer Software Engineering Workshop, November 1979
SEL-80-005, A Study of the Musa Reliability Model, A. M. Miller, November 1980
SEL-80-006, Proceedings From the Fifth Annual Software Engineering Workshop, November 1980


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


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


SEL-82-004, *Collected Software Engineering Papers: Volume 1*, July 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-89-005, *Lessons Learned in the Transition to Ada From FORTRAN at NASA/Goddard*, C. Brophy, November 1989


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


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


**SEL-RELATED LITERATURE**


NOTES:

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

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


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

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

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


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

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


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.
The Software Engineering Laboratory (SEL) is an organization sponsored by NASA/GSFC and created to investigate the 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.