UNCLASSIFIED

Copy 17 of 82 copies

IDA PAPER P-2996

# SURVEY OF SOFTWARE METRICS IN THE DEPARTMENT OF DEFENSE AND INDUSTRY


Beth Springsteen, *Task Leader*

Dennis W. Fife
John F. Kramer
Reginald N. Meeson
Judy Popelas
David A. Wheeler




April 1994



*Prepared for*
Office of the Director, Defense Research and Engineering

Approved for public release, unlimited distribution: September 23, 1994.


IDA

INSTITUTE FOR DEFENSE ANALYSES
1801 N. Beauregard Street, Alexandria, Virginia 22311-1772


UNCLASSIFIED


IDA Log No. HQ 94-045746

## DEFINITIONS

IDA publishes the following documents to report the results of its work.

### Reports

Reports are the most authoritative and most carefully considered products IDA publishes. They normally embody results of major projects which (a) have a direct bearing on decisions affecting major programs, (b) address issues of significant concern to the Executive Branch, the Congress and/or the public, or (c) address issues that have significant economic implications. IDA Reports are reviewed by outside panels of experts to ensure their high quality and relevance to the problems studied, and they are released by the President of IDA.

### Group Reports

Group Reports record the findings and results of IDA established working groups and panels composed of senior individuals addressing major issues which otherwise would be the subject of an IDA Report. IDA Group Reports are reviewed by the senior individuals responsible for the project and others as selected by IDA to ensure their high quality and relevance to the problems studied, and are released by the President of IDA.

### Papers

Papers, also authoritative and carefully considered products of IDA, address studies that are narrower in scope than those covered in Reports. IDA Papers are reviewed to ensure that they meet the high standards expected of refereed papers in professional journals or formal Agency reports.

### Documents

IDA Documents are used for the convenience of the sponsors or the analysts (a) to record substantive work done in quick reaction studies, (b) to record the proceedings of conferences and meetings, (c) to make available preliminary and tentative results of analyses, (d) to record data developed in the course of an investigation, or (e) to forward information that is essentially unanalyzed and unevaluated. The review of IDA Documents is suited to their content and intended use.

The work reported in this document was conducted under contract MDA 903 89 C 0003 for the Department of Defense. The publication of this IDA document does not indicate endorsement by the Department of Defense, nor should the contents be construed as reflecting the official position of that Agency.

UNCLASSIFIED

IDA PAPER P-2996

# SURVEY OF SOFTWARE METRICS IN THE DEPARTMENT OF DEFENSE AND INDUSTRY


Beth Springsteen, *Task Leader*

Dennis W. Fife
John F. Kramer
Reginald N. Meeson
Judy Popelas
David A. Wheeler


| Accesion For | |
| --- | --- |
| NTIS CRA&I | ☒ |
| DTIC TAB | ☐ |
| Unannounced | ☐ |
| Justification | |
| By | |
| Distribution / | |
| Availability Codes | |
| Dist | Avail and / or Special |
| A-1 | |


April 1994






IDA

INSTITUTE FOR DEFENSE ANALYSES


Contract MDA 903 89 C 0003
Task T-A15-742


UNCLASSIFIED

# PREFACE

This document was prepared by the Institute for Defense Analyses under the Task Order, Analysis of Software Initiatives, for the Office of the Director of Defense Research and Engineering. It fulfills an objective of the task, to survey the availability and applicability of commercial capabilities in the area of software metrics, including research and development and state of practice, and to compare commercial use of such capabilities with those within the Department of Defense.

A review of this document was performed by Mr. Bill Brykczynski, Dr. Michael Frame, Ms. Audrey Hook, Dr. Richard Ivanetich, and Ms. Kathleen Jordan.

# EXECUTIVE SUMMARY

This report provides summaries and an initial assessment of the state of development and use of computer software measurement in Department of Defense (DoD) organizations and private companies, including DoD contractors. The work was undertaken to support planning for improving the implementation of software metrics throughout DoD's weapon system acquisition process.

The term "software metrics" in practice refers to a wide variety of data associated with software development projects. Many metrics are composites or aggregates of individual factors, and many are direct observations or determinations of someone such as a programmer, configuration manager, or software tester. Metrics address software product factors such as size in lines of source code, as well as management and work progress factors such as expended funds or effort.

The study results emerged primarily from interviews with about 50 people within DoD organizations and industry, selected because they are leaders in developing and applying software metrics and also are widely knowledgeable of metrics usage within their organization. They appeared to have the best perspective for addressing all of the study's questions, while practical limitations prevented interviews also with direct metrics users such as DoD acquisition program offices or industry product development groups. Metrics policies, guides or handbooks, and technical articles also were reviewed.

The analysis addresses the following areas:

- Goals for software metrics
- Implementation of metrics, including policy and standards
- Benefits and successes in applying metrics
- Metrics used and their reporting across organizational levels
- Tools and repositories supporting metrics collection and use
- Lessons learned from metrics experience and best practices

- Future plans and research recommendations

Summary reports are given for 11 DoD organizations, the National Aeronautics and Space Administration Goddard Space Center's Software Engineering Laboratory, and the Software Engineering Institute of Carnegie Mellon University, as well as 10 industry organizations. The private companies are not identified by name, as agreed in order to encourage their information disclosure.

Two key questions were among those formulated as the scope of the study:

- What factors distinguish industrial and DoD practice in software metrics?
- Does industrial practice reveal metrics capabilities, concepts, or tools that could benefit DoD's weapon system acquisition process?

Significant conclusions on these questions include the following:

- The main focus of software measurement considerations within DoD organizations is to help acquisition program managers. However, such support is limited and immature. DoD organizations recommend metrics without giving sufficient guidance for incorporating them into the practical decisions and exercise of program management. Measurement needs of DoD activities, other than acquisition program management, are insufficiently recognized and supported. Two types of DoD in-house software development activity for weapon systems may be cited as examples of other software metrics needs: development of support software, e.g., simulators, trainers, test environments, and post-deployment software support of weapon systems.
- Major impediments to metrics improvement within DoD include lack of goals that especially benefit software acquisition, lack of documented successes, lack of resources for metrics technology transfer and training, and apparently strong program office opposition to requirements imposing additional unfunded cost or current metrics reporting to outside organizations.
- In general, industry practice is more mature and further evolved for both DoD contractors and strictly commercial organizations. It is motivated by a clear and widely supportable purpose, i.e., improving a company's competitiveness for the sake of gaining new business. Companies are moving to gain marketplace benefits from measurement, not merely to solve short-range problems within projects. In their approach, companies are establishing metrics policies and

measurement support groups, and they are investing in building or acquiring better metrics tools.

- Industry trends appear at odds with recommendations heard relative to current DoD acquisition practice. Company practices aim to collect the data needed for well-defined purposes, to integrate measurement with development process innovations, and to exploit metrics in an overall business perspective. DoD staff are recommending limited data collection and measurement practices devised to meet constraints and choices suited to individual development projects.
- To pursue a measurement approach comparable to that emerging in industry, DoD must have acquisition process objectives that transcend individual acquisition programs. Also, it would have to provide incentives for performance similar to what the marketplace offers to commercial producers.

## Table of Contents

## List of Figures

## List of Tables

# 1. INTRODUCTION

## 1.1 PURPOSE

This report provides summaries and an initial assessment of the state of development and use of computer software measurement in Department of Defense (DoD) organizations and private companies. The survey is the first phase product of a two-phase project. The second phase will prepare policy and research recommendations for consideration by DoD's Director, Defense Research and Engineering (DDR&E), for the purpose of improving the implementation of software metrics throughout DoD's weapon system acquisition process.

## 1.2 SCOPE

The scope of the assessment is shown by the following basic questions that were formulated at the outset:

- What factors distinguish commercial and DoD practice in software metrics?
- What problems are managers trying to address through metrics collected during the software development process?
- To what extent has the application of software metrics proven successful?
- Does commercial practice reveal metrics capabilities, concepts, or tools that could benefit DoD's software acquisition process?
- What lessons have been learned in applying software metrics?
- Which metrics have been most beneficial?
- Has the use of metrics influenced the software development process or planning of future efforts?
- What human factors issues have been taken into consideration in applying metrics?

The scope also included initially a comparative assessment of DoD science and technology programs with commercial industry needs and ongoing independent research efforts. Results in this direction are limited by the sources available for the study and their disclosure of ongoing research subjects.

## 1.3 APPROACH

Phase one results have been developed through two major sources (see Figure 1-1). The primary information source was telephone and personal interviews conducted with about 50 people in DoD organizations and private companies. The interviewees were selected because they were leaders in developing and applying software metrics or were widely knowledgeable of metrics use within their organization and closely related organizations. As the figure suggests, interviewees for DoD usually were members of a software engineering research or support group within a subordinate command or engineering laboratory of a military service. By contrast, interviewees from industry usually were members of a software engineering process improvement group attached to a major company division devoted to one application or business domain.

For practical reasons, interviews were not done with organizations that are direct, first-line metrics users, i.e., neither DoD acquisition program offices nor industry projects, whether DoD contractor projects or strictly commercial projects. This is noteworthy because use of metrics in DoD acquisition is a major study consideration. It is believed that the persons contacted were able to describe the acquisition viewpoint and usage status in basic terms, as well as respond well to other study questions. But this survey does not provide detailed information about individual metrics and their implementations in today's major DoD acquisition or industry development programs.

A second major source was a substantial collection of metrics documents obtained from the interviewees and others. Significant documents and published articles were reviewed to develop additional depth for answering the primary questions.

Also, some professional meetings or metrics workshops provided information, including the Air Force Rome Laboratory's Cooperstown I Workshop, entitled "Establishing a National Vision and Force in Software Through Measurement" [Kaman 1993].

A perspective must be given for the terms "software measurement" or "software metric." In practice, they aren't used with much precision. For example, many metrics are not measurements in the usual sense of mechanically determined physical values from a

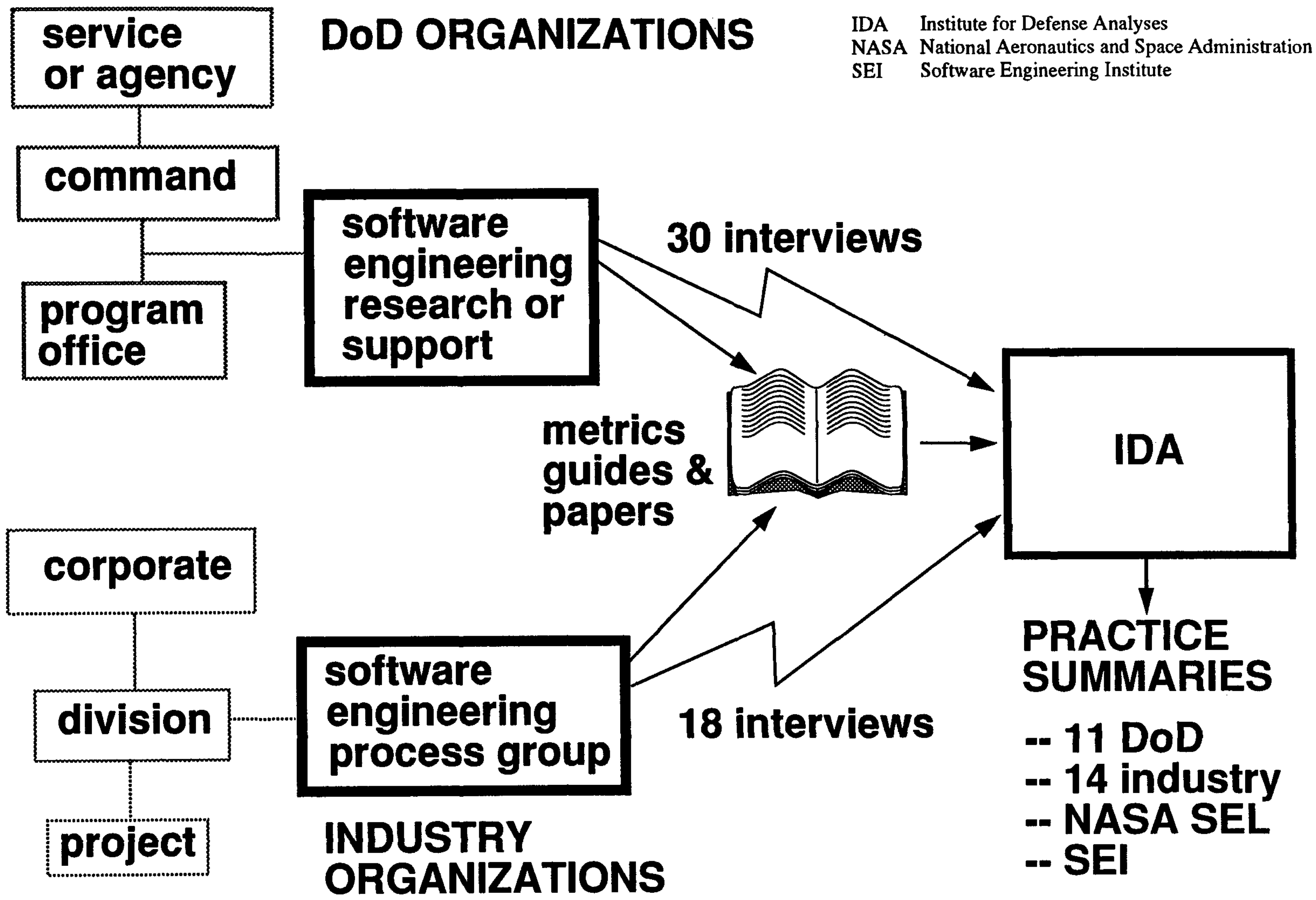



Figure 1-1. Information Sources and Capture

wide range of possibilities. Rather they are direct judgements made by some person, possibly with wide latitude to choose. An example is software progress, involving a decision that a software component's design is completed. This would be a somewhat arbitrary decision since redesign often is involved to fix bugs found later in testing. Further, some so-called metrics in fact are derived or aggregated from several factors, e.g., requirements stability may track unresolved, deleted, deferred, and committed items for different software builds. Finally, many software metrics are not peculiar to software as a technical product. Rather they are management and work progress factors, e.g., funds or effort expended, that are useful in managing any kind of product development. Throughout this report, the term "software metric" is used most often and with the customary latitude indicated above.

## 1.4 REPORT OVERVIEW

Substantive information collected from interviews is summarized in Appendix A for DoD organizations, National Aeronautics and Space Administration (NASA), and Software Engineering Institute (SEI), and in Appendix B for private companies. In some cases, information from 2 or 3 interviews for the same organization or chain of command has been merged, yielding summary reports for 13 government organizations and 10 industry organizations. (Summaries for four companies that required legal nondisclosure agreements are published separately and are available only to government officials needing this information. This information was used in this report for composite numerical summaries and to confirm general conclusions.) The private companies are not identified by name, as agreed in order to encourage their information disclosures. The following outline applies for each summary in Appendices A and B.

- Metrics and reports
- Levels of metrics reporting across organizations
- Tools and repositories supporting metrics collection and use
- Best practices and lessons learned from metrics experience
- Future plans and directions of organizations, including research

Using the summaries and pertinent documents, Chapters 2 and 3 assess the state of practice across DoD and industry organizations, respectively, addressing the topics above. Chapter 4 is a comparative analysis across the DoD and industry organizations, and provides answers to the basic questions posed for the study.

# 2. SUMMARY OF DOD METRICS DEVELOPMENT AND USE

This chapter provides an overall analysis of metrics development and use, as reflected in the 11 Appendix A summaries for DoD organizations. Table 2-1 lists the DoD organizations and their acronyms.

The analysis aims to identify trends and conclusions that could be important in formulating future DoD policy or research goals for software measurement. As the DoD analysis is presented, information also may be given for comparison regarding the NASA Software Engineering Laboratory (SEL) experience and the Software Engineering Institute's (SEI's) metrics recommendations or research.

**Table 2-1. DoD Organizations Surveyed**

| Organization Name | Acronym |
|---|---|
| Army Armament Research, Development, and Engineering Center | ARDEC |
| Army Communications - Electronics Command | CECOM |
| Army Missile Command | MICOM |
| Army Operational Test and Evaluation Command | OPTEC |
| Naval Air Systems Command | NAVAIR |
| Naval Air Warfare Center (summarized with NAVAIR) | NAWC |
| Naval Undersea Warfare Center | NUWC |
| Air Force Materiel Command | AFMC |
| Air Force Electronics System Center | ESC |
| Air Force Rome Laboratory | RL |
| Defense Information Systems Agency | DISA |
| Joint Logistics Commanders | JLC |

Metrics have existed and been of interest nearly as long as the software field. For example, a history [Sackman 1967] of the Semi-Automatic Ground Environment (SAGE) project provided some of the earliest metrics on large-scale DoD software development. Some of the organizations contacted in this study can cite metrics experiences dating back that long. Most of their efforts have been motivated by the importance of software in defense acquisitions, problems encountered with software when acquisitions enter the system testing phase, and need for development processes that incorporate process and product measurement.

All of the DoD organizations are engaged in evolving and implementing software metrics to serve software development and acquisition management. DISA, JLC, and AFMC now are jointly involved in developing a software metrics guide under JLC auspices, and other organizations already have issued guidance. The guidance documents define and recommend (implicitly or explicitly) a preferred set of metrics. Table 2-2 describes the metrics sets and guides examined in this survey. More analysis of these metrics sets is given later in this chapter.

**Table 2-2. DoD Metrics Guides Examined**

| Source Organization | Reference to Metrics Guide | Name Used Here for the Metrics Set |
|---|---|---|
| Army | [Army 1992] | STEP (Software Test and Evaluation Panel) |
| Air Force | [SAF/AQ 1994] | USAF |
| CECOM | [Dyson 1991] | CEMSM (CECOM Executive Management Software Metrics)[a] |
| ESC | [Schultz 1988] | ESC |
| MICOM | [MICOM 1991] | MICOM |
| NAVAIR | [NAVAIR 1992] | NAVAIR |
| NAWC-AD | [Rozum 1992b] | NAWC |
| NUWC | [McGarry 1992a] | NUWC |

a. This metrics set was developed prior to the Army's STEP set and no longer is CECOM's primary recommendation; see [CECOM 1993].

## 2.1 GOALS FOR METRICS

Goals to be met through metrics are stated in different ways, as shown by the sample of statements in Table 2-3 taken from metrics guides issued by DoD organizations. A few of these are more definite than others, but most of them signal vague or broad goals by using terms such as "visibility," "insight," or "indications." Most organizations recognize that metrics are only indicators that may assist project monitoring and management decision-making. Lessons learned (see Section 2.4) reinforce this view.

**Table 2-3. Sample of Goals Stated in DoD Metrics Guides**

- Early indications of potential software development problems.
- Visibility to verify the current status of software development.
- Determine and monitor software maturity and readiness for test.
- Visibility and control of software development within a program.
- Management insight into all aspects of the development process & products.
- Insight into software development processes or process improvement efforts.
- Demonstrate achievement of required level of functionality and maturity.
- Development process assessment, with increased quality of deployed software.
- Management insight into developmental status and long-term supportability.
- Consistent and quantifiable insight into software aspects of system development.

## 2.2 METRICS IMPLEMENTATION

Table 2-4 and Table 2-5 tabulate the nature of guidance documents produced by the DoD organizations so far and the year that the most current guidance was issued.

**Table 2-4. Form of Documents Issued**

| Form of Metrics Documents Issued | Number of Organizations (Out of 11) | Number Providing Contracting Guidance |
|---|---|---|
| Research & study reports, or draft reports or guides | 4 | 0 |
| Concise guides describing recommended metrics and application experience | 4 | 0 |
| Extensive handbook for practice within acquisition management | 3 | 3 |

**Table 2-5. Year of Publication**

| Year Range | Number of Organizations (Out of 11) |
|---|---|
| 1985 or earlier | 1 |
| 1986 through 1989 | 1 |
| 1990 or later | 9 |

From this data and other information about the current thrust of these organizations (see Sections 2.4 and 2.5), it is evident that software metrics are still evolving with respect to both technical content and implementation in practice.

Given the significant history of metrics research and experience, it is worth asking whether certain technical precedents have primarily influenced recent metrics efforts. Table 2-6 identifies prior work that is claimed as influencing the DoD organizations contacted. Multiple citations by one or more organizations occur in this table, but it shows no dominance by one or two precedents.

**Table 2-6. Citation of Prior Sources as Major Influence**

| Prior Document or Effort | Number of Citations |
|---|---|
| Software Engineering Institute metrics, e.g., [Rozum 1992a], [Carleton 1993b] | 1 |
| SEI Capability Maturity Model for Process Improvement [Humphrey 1987] | 3 |
| Air Force Systems Command Pamphlets 800-14 or 800-43, or Army Materiel Command Pamphlets P 70-13 or P 70-14 | 4 |
| Rome Lab software quality factors [RL 1985] | 3 |
| NASA SEL [McGarry 1993] | 2 |
| Army STEP metrics [Army 1992] | 3 |

Another consideration in metrics implementation is whether an organization by policy has required software metrics to be collected and used to monitor acquisition efforts. Among DoD organizations contacted, there are three such cases. This includes two Service-wide policies (Army mandate of the STEP metrics and the Air Force software metrics policy), and NAVAIR's policy for its AIR-546 organization. Two other DoD organizations indicated that a draft policy formulation was being considered. However, four stated that collection and use of software metrics should remain an option for program managers, with an acceptable policy providing only guidance to program managers.

## 2.3 BENEFITS

Only a few of the interviews and associated documents provided detailed evidence of widespread application and benefits from recommended metrics. ARDEC, ESC, NASA, and NUWC provided clear evidence by citing specific acquisition programs or displaying actual metrics data from specific programs. NASA, in particular, uses the convincing technique of presenting actual metrics from specific programs to illustrate and explain its metric recommendations and their use.

Several organizations, including those involved in implementation through policy, cited difficulties and opposition encountered in transitioning their recommendations into acquisition practice.

Regarding aid to program management, software metrics are unlikely to provide the earliest information for detecting problems such as an unfulfilled milestone. Because of both collection and trending time lags, indications from metrics usually lag the actual emergence of problems. Nevertheless, quantitative metrics data is valuable for foreshadowing risks [Fife 1993], such as likely missing a future milestone, and assessing their potential impact on program success.

## 2.4 METRICS AND REPORTS

This section aims to assess the degree of commonality among metrics recommended by DoD organizations as indicated by the eight metric sets identified in Table 2-2. Two levels of commonality are considered: first the basic types or categories of metrics data recommended, then attributes of the precise metrics definitions recommended.

### 2.4.1 Common Metric Types

Table 2-7 briefly defines categories of information for which metrics are often defined for collection, and indicates how many of the eight metric sets include one or more metrics for each category. Each set may include additional metrics categories not listed here. Table 2-7 shows 10 categories of metrics that occur in at least 6 out of the 8 sets.

### 2.4.2 Comparison of Size Metric Guidance

To look further at precise metrics defined, this and the sections to follow will compare the definitional guidance provided for only the size, requirements, defects, and progress categories. As an additional allowance for brevity, only five of the eight sets are considered, i.e., CEMSM, STEP, USAF, ESC, and NAVAIR. These were chosen because the guidance was specific and sufficient for comparison. Table 2-8 compares definitional factors and guidance for the size metric.

### Table 2-7. Metrics Categories and Usage

| Category | Brief Description | Number Using It (Out of 8 DoD Sets) |
|---|---|---|
| Cost | expended funds | 6 |
| Effort | expended labor, including number of staff on project | 8 |
| Schedule | status of key events or deliverables relative to due date | 7 |
| Size | size of intermediate or deliverable products (code, documentation, requirements, no. of CSUs, no. of lines of PDL)[a] | 7 |
| Progress | status of software components such as designs, code, and test cases, procedures, and reports | 7 |
| Defects | defects and software change requests | 8 |
| Inspections | information on the process of inspecting software products, such as effort expended, defects identified | 0 |
| Requirements | status of requirements, requirements changes and their impact | 6 |
| Software Quality | design stability, attributes of design or code such as complexity, reliability, maintainability, flexibility | 7 |
| Customer Satisfaction | customer perception of the software product | 0 |
| Cycle Time | duration of specific types of development tasks | 6 |
| Computer Utilization | utilization of target machine resources | 7 |
| Development Environment | status, utilization, and maturity of development tools, processes, and methodologies | 4 |
| Release Capability | functionality delivered per build or release | 3 |
| Product Performance | for example: speed, throughput, reliability | 0 |
| Staffing | number of staff by skill or experience level, unplanned staff losses, training needs & accomplishments of staff | 2 |
| Reuse | amount and impact of reuse | 5 |

a. CSU - Computer Software Unit; PDL - Program Design Language.

Table 2-8. Comparison of Size Metric Guidance[a]

| Size Metric Guidance | Included in | | | | |
|---|---|---|---|---|---|
| | CEMSM | STEP | USAF | ESC | NAVAIR |
| Specifies Source Lines of Code (SLOC) for measuring size | * | * | * | * | * |
| Admits or recommends additional or alternative sizing metrics | * | | * | | |
| Specifies basic criteria for counting SLOC | * | | | * | * |
| Defines extensive SLOC counting options for multiple languages | | | * | | |
| Admits or recommends project tailoring of SLOC counting rules | * | * | * | * | |
| Defines data declarations, but not comments as part of SLOC | * | * | | * | * |
| Recommends distinguishing new, reused, & modified SLOC | * | | * | * | * |
| Advises on risk indicated by SLOC variance over life cycle | * | | | * | |
| Advises on collection & reporting events or intervals | * | * | | * | * |

a. The asterisk (*) denotes that the information is included or the criterion met. A blank space denotes it has not been included or met.

There is a marked difference between the SEI approach adopted in the Air Force policy and that of the other metric sets in this table. The other sets give rather basic definitions of how SLOC should be identified and counted (ESC's report has somewhat more guidance than other documents). The underlying concept of these other sets is that a using organization will be able to refine and specify the SLOC definition in order to use a size metric reliably. However, SEI highlights a multitude of factors and programming language dependencies that may be brought into play for prescribing how to count SLOC. Instead of specifying preferred rules, SEI provides descriptive forms so that each organization can specify precisely what shall be included and excluded in counting SLOC. Even embedded comments, which are to be left out for the other metric sets, are considered countable in several different categories in the SEI guidance. This perspective implies that standardization of counting rules is considered difficult to achieve. The other metrics sets also do not demand standardization

Of these five metric sets, only the USAF set suggests an alternative to SLOC, i.e., function points, as the appropriate size metric. However, the CEMSM set does include produced documentation page count as an additional size metric, but with minimal guidance on its interpretation or monitoring.

### 2.4.3 Comparison of Requirements Status Guidance

Requirements status metrics serve to monitor requirements growth and change over the development life cycle. Table 2-9 lists and compares definitional factors and guidance for requirements status.

**Table 2-9. Comparison of Requirements Status Guidance[a]**

| Requirements Status Metric Guidance | Included in | | | | |
|---|---|---|---|---|---|
| | CEMSM | STEP | USAF | ESC | NAVAIR |
| Specifies requirements count for monitoring growth & change | * | * | | * | * |
| Defines criterion for identifying a countable requirement | | | | * | |
| Requires change request reports | * | * | * | * | |
| Recommends impact or priority classification of change requests | * | | * | * | * |
| Requires SLOC impact assessment | | * | | | |
| Requires change monitoring for each CSCI or CSUs[b] | * | * | | * | * |
| Advises assessment on monthly basis | * | * | | * | * |
| Advises on risk or action indicated by trends over life cycle | * | | | * | |
| Advises on collection & reporting events over life cycle | * | * | | * | * |

a. The asterisk (*) denotes that the information is included or the criterion met. A blank space denotes it has not been included or met.
b. CSCI - Computer Software Configuration Item; CSU Computer Software Unit.

Requirements are monitored through two basic data inputs. The first is a count of requirement statements in the software requirements specification or its amendments (commonly, the "shall" statements from the DOD-STD-2167A Software Requirements

Specification or SRS document). The second are monthly records of change requests for requirements.

The CEMSM and ESC metric sets are very similar in their guidance about this metric. Status of change requests must be monitored and they should be classified to indicate potential impact on effort. The STEP guidance goes further by requiring impact to be assessed in terms of SLOC modifications

### 2.4.4 Comparison of Defect Status Guidance

A defect status metric serves to track the number and age of unresolved issues identified during software testing. The metric gives insight into quality and the developer's capability to resolve and fix defects quickly. Table 2-10 compares definitional characteristics and guidance.

**Table 2-10. Comparison of Defect Status Guidance[a]**

| Defect Status Metric Guidance | Included in | | | | |
|---|---|---|---|---|---|
| | CEMSM | STEP | USAF | ESC | NAVAIR |
| Specifies count of "open" or unresolved defects as status metric | * | * | * | * | * |
| Specifies criteria or method for identifying defects for reporting | | | | | |
| Recommends content of defect or trouble reports | | * | | | * |
| Specifies defect closure requirements, e.g., regression testing | | | | | |
| Describes interplay of testing depth and defect discovery | | * | | | |
| Recommends tracking defects at CSCI or lower CSU levels | * | * | | * | * |
| Advises on priority classes for defect tracking | * | * | | * | |
| Advises monthly collection and assessment interval | * | * | | * | * |
| Advises on coupling collection to test & rework effort | * | | | | |
| Advises on risk or action indicated by defect trends | * | * | * | | |

a. The asterisk (*) denotes that the information is included or the criterion met. A blank space denotes it has not been included or met.

Despite a possible impression from Table 2-10 that there is little commonality, the five sets, in fact, identify the same metric, a count taken periodically (usually monthly) of so-called "open" or unresolved defects. However, Table 2-10 does indicate a significant disparity in the guidance provided by each source for applying this metric. In fact, all of the sources provide minimal guidance and requirements. A particular weakness is lack of guidance for identifying defects and their status in relation to testing, diagnosis, repair or rework, and retesting. Perhaps this is considered beyond the scope of metrics definition, but it is crucial for consistent collection and interpretation of defect reports.

### 2.4.5 Comparison of Development Progress Guidance

A development progress metric serves to monitor continuing development progress in terms of the software product or its components. Table 2-11 compares guidance on this metric. The USAF set does not have this metric. All others use a count of software product units such as CSCIs or CSUs that have completed a designated life cycle stage. Accomplishing a walkthrough or inspection is a significant progress stage in one set.

**Table 2-11. Comparison of Development Progress Guidance[a]**

| Development Progress Metric Guidance | Included in | | | | |
|---|---|---|---|---|---|
| | CEMSM | STEP | USAF | ESC | NAVAIR |
| Specifies progress as count of units in prescribed stages | * | * | | * | * |
| Denotes software status as a DOD-STD-2167A stage or equivalent | * | * | | * | |
| Specifies exit criteria for confirming status in each stage | | * | | | |
| Specifies progress as SLOC delivered via configuration control | | * | | | |
| Recommends walkthrough or inspection as important unit stage | | | | * | |
| Recommends tracking progress for CSCI or lower CSU levels | * | * | | * | * |
| Advises monthly collection and assessment interval | * | * | | * | * |
| Advises on risk or action indicated by progress trends | * | * | | * | |

a. The asterisk (*) denotes that the information is included or the criterion met. A blank space denotes it has not been included or met.

### 2.4.6 Metrics Reporting

DoD organizations seldom report metrics beyond a program manager to higher command or organizational levels. For reporting to a program manager, or for that occasional briefing to other parties on metrics applications, graphic displays are essential. Typical displays are illustrated here. The common use of estimated SLOC is to track its changes over time and correlate the estimates with other engineering and programmatic information, e.g., requirements status. Figure 2-1 illustrates a typical presentation suggested for sizing data. Figure 2-2 is a typical suggestion for graphing requirements status and volatility. Figure 2-3 illustrates a suggested graphic for tracking defects. Figure 2-4 illustrates a typical presentation of development progress information.

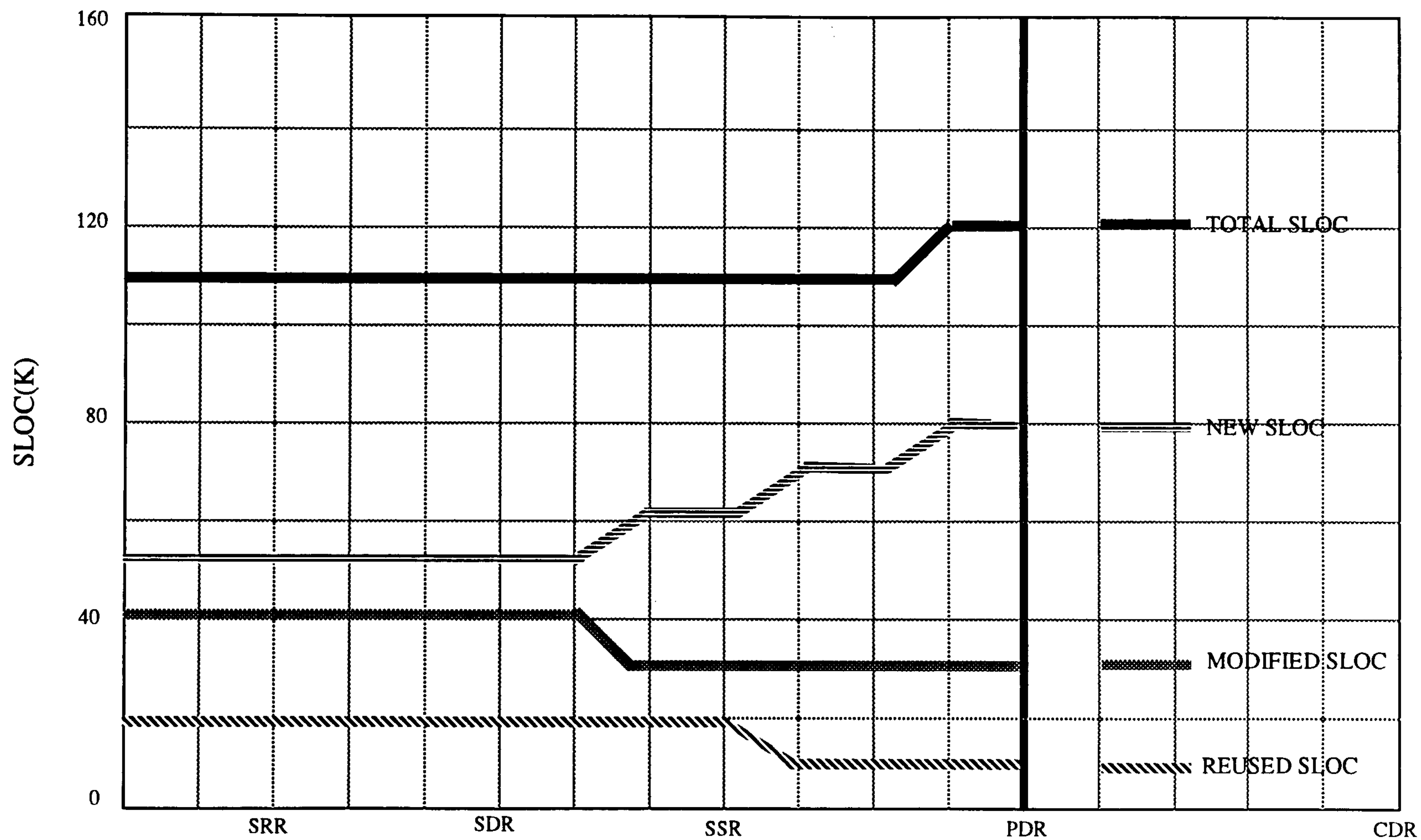
## 2.5 TOOLS AND REPOSITORIES

The goal of this section is to describe viewpoints and trends among the contacted organizations regarding tools for metrics collection and analysis, including data repositories for metrics information. Repositories take on a special interest if they are populated and made accessible across many acquisition programs, to assist in software technology planning or acquisition planning and analysis.

### 2.5.1 Tools

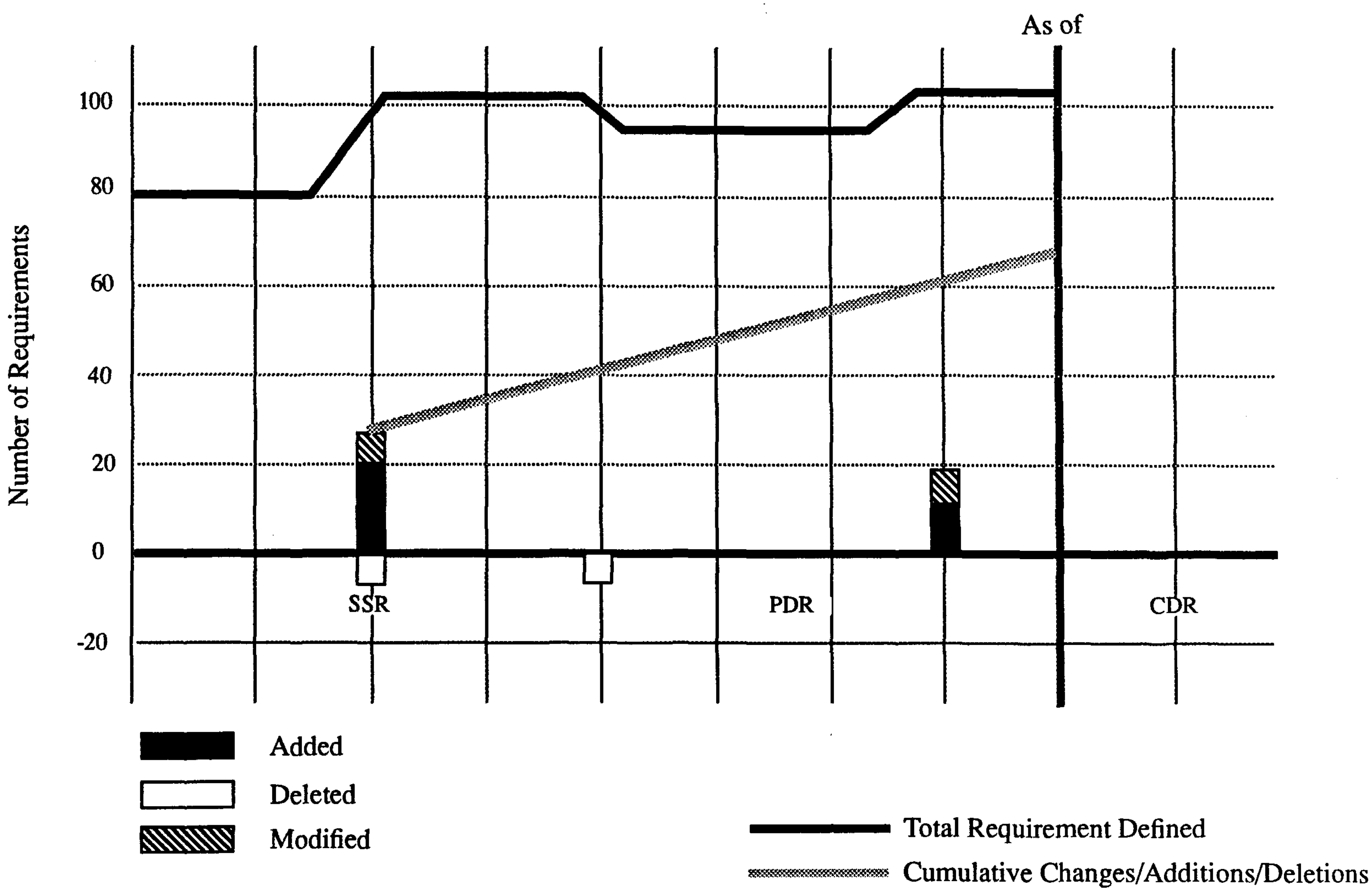
The information available from DoD organizations about preferred tools, tool capabilities, or tool needs was limited. This was somewhat surprising since the need for non-intrusive data gathering is often cited. On the other hand, NASA SEL stated that sufficient data could be collected readily on paper forms filled out once a week. In fact, much of the information listed in Table 2-7 comes directly from human observation and judgement (e.g., what is the state of a given CSU). Only a few of these metrics would involve more effort than occasional manual data recording. Exceptions would include automated analysis of source code statements for certain quality factors and measurement of product performance or throughput. Several organizations stated that widely available spreadsheet computer programs for personal computers serve as their primary tool. A commercial database package also was cited as an appropriate and successful tool for a metrics data repository.

Nevertheless, some tool support and guidance would help organizations to accept and practice a common metrics approach. The Army STEP program for instance has developed a database capability that is being disseminated to Army organizations to assist




| | |
|---|---|
| SDR | System Design Review |
| SSR | Software Specification Review |
| SRR | System Requirements Review |
| PDR | Preliminary Design Review |
| CDR | Critical Design Review |

**Figure 2-1. Size**




SSR Software Specification Review
PDR Preliminary Design Review
CDR Critical Design Review

**Figure 2-2. Requirements Volatility**

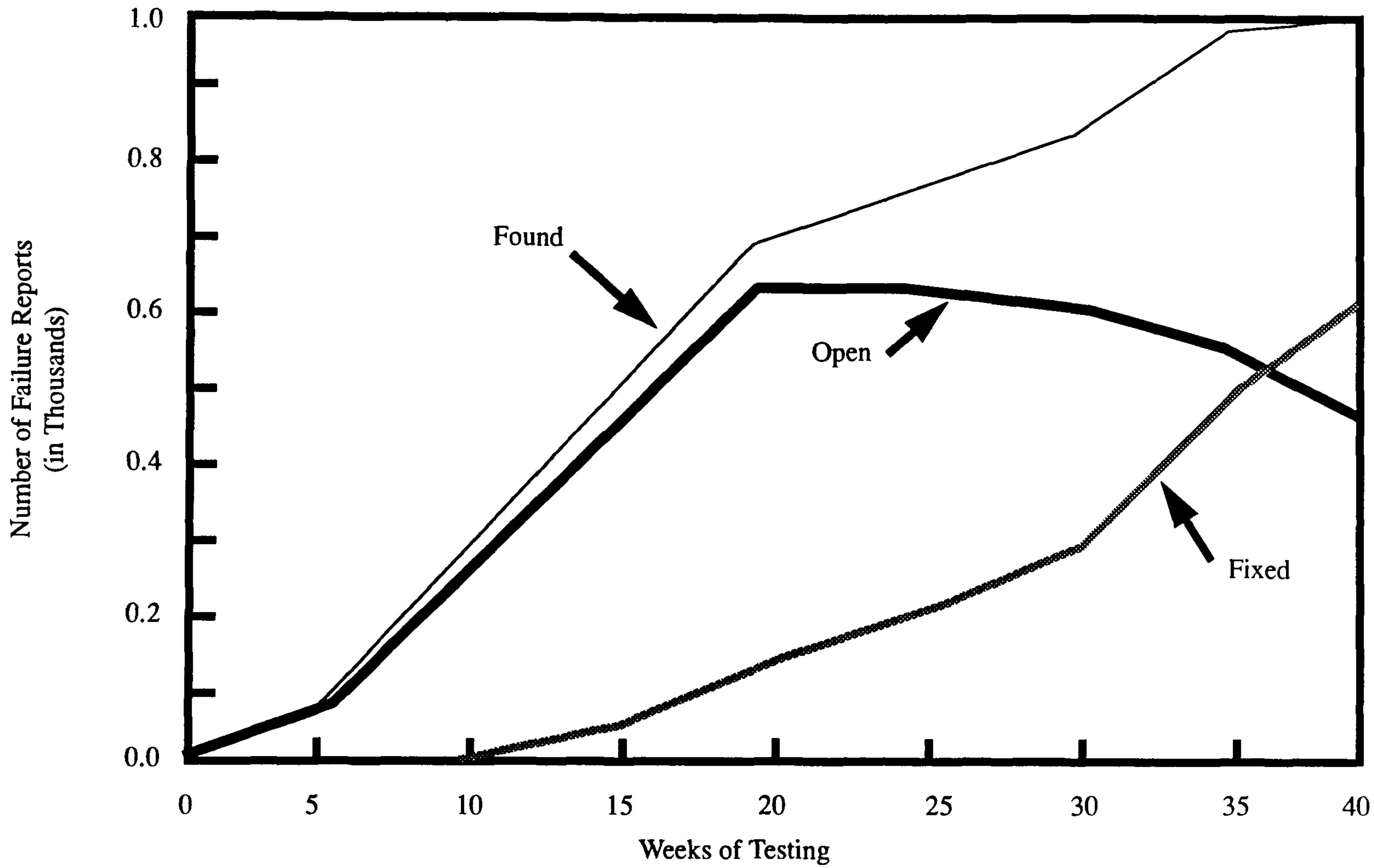



**Figure 2-3. Defects**

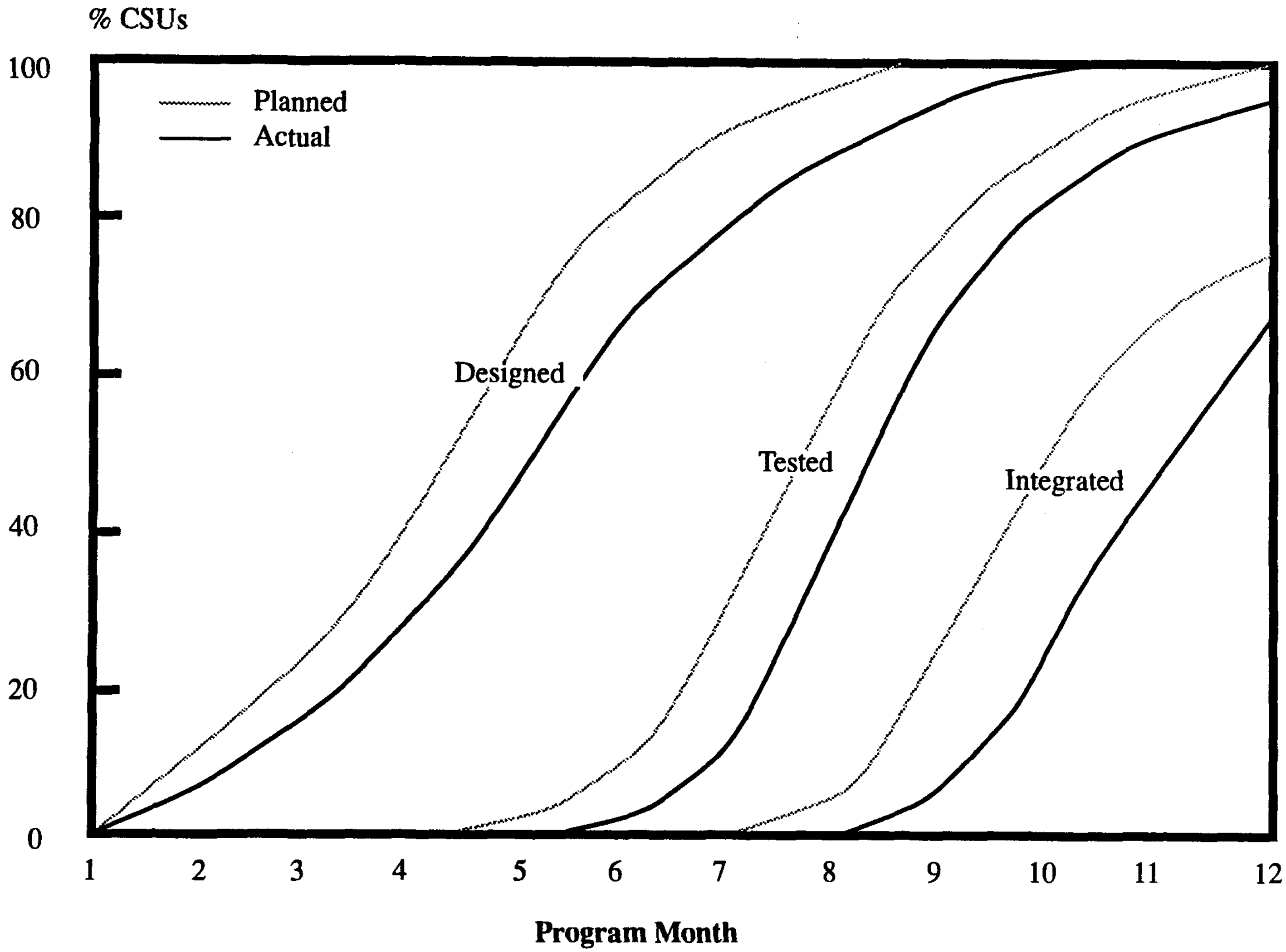



**Figure 2-4. Development Progress**

collecting and storing metrics data for local use. Several organizations mentioned a future goal to investigate tools and make recommendations to users.

### 2.5.2 Repositories

Several organizations retain metrics data from projects for research and analysis purposes, but only three (NASA, MICOM, and SEI) have investigated or prototyped computer repository designs to collect metrics from many projects. Most sources expressed serious opposition to the idea of centrally collecting metrics information. The general fear is misinterpretation and misuse of such data.

The NASA SEL provides one example of collecting metrics data from many projects and using it to establish experience models and performance norms for guiding future programs, rather than controlling current programs or evaluating program managers.

## 2.6 BEST PRACTICES AND LESSONS LEARNED

Observations expressed by the interviewees on best metrics practices and lessons learned are paraphrased and consolidated in the following sections, with major themes identified at the beginning as part of the section headings.

### 2.6.1 Emphasize Simple and Limited Metrics

Metrics-experienced organizations have discovered that their early efforts were mistaken in asking for too much data when they were uncertain of its purpose and value to them. Today, recommended metrics may collect less than half of the data that was sought in prior years. Collecting data is difficult and it is important to have a simple, goal-directed metrics program.

It is also important to provide guidance on practices of applying software metrics, rather than just defining the desired metrics.

A potential barrier to measurement is that companies are not willing to relinquish project and business data that may disclose their competitive advantages or way of doing business. No appropriate solution was identified, but limiting and simplifying required data may be part of one.

One interviewee also recommends emphasis on product metrics, such as defects, as more relevant and effective than process metrics such as staffing or process maturity.

### 2.6.2 Focus on Local Issues

Interviewees recommend adopting small metric sets and collection technology that is appropriate to improve a specific development project. It was argued as well that organizations should be compared relative to their environment or organizational context. The conclusion drawn is that metrics must be defined for project-specific suitability and so they would not be effective for DoD-wide research needs.

### 2.6.3 Gain Executive Commitment

Senior executives such as Program Executive Officers (PEOs) seldom are directly interested in software, since their purview is projects or systems as a whole. Yet it is important for metrics progress and implementation to gain senior management commitment. There is much experience to show that DoD is not willing to pursue metrics data collection due to cost. DoD program offices typically delete software metrics requirements from contracts.

### 2.6.4 Avoid Perfecting Standards

Past experience does not bode well for a goal of standardizing improved metric definitions. As one contact reported, after you have good definitions, no one follows them. There was considerable sentiment that time spent perfecting software metric definitions is not worthwhile.

### 2.6.5 Technical Problems of Metrics

A few comments also were made that are limited to specific technical considerations about metrics and their collection. For one, having several individuals judge a complex factor on an uncalibrated scale, such as 1 to 5, is not recommended because the outcome is not repeatable over time. Other comments involved Ada code quality factors. Cyclomatic complexity and Halstead complexity metrics were considered ineffective for improving Ada programs. Contractors were considered as already delivering better Ada metrics via tools such as Adamat than what are formulated in current DoD metrics sets.

## 2.7 FUTURE PLANS AND DIRECTIONS

This section identifies metrics research areas suggested or planned by the DoD sources and also their recommendations on DoD goals or actions regarding metrics in the acquisition process.

### 2.7.1 Research and Development

One contact had the opinion that too much research already had been put into metrics. The need now is infusion of measurement into the practitioner's world (e.g., provide guidance on how to use metrics data and determine its benefits).

The following suggestions addressed the need for better metrics than what is now available:

- Better measure of software reliability. Reliability of a module in testing phase may not be an accurate measure of its reliability in an operating environment.
- Better quantitative metrics for evaluating software code, other than complexity.
- Better requirements metrics in addition to stability metrics.
- More understanding of the systems engineering process and different ways to analyze problem reports (source of problem, when identified, necessary corrections).

The need for better tools was mentioned:

- Tools to automate collection, analyses, and reporting of metrics.
- Tools for software development that also collect metrics (e.g., configuration management (CM)).

And suggestions were made for other research in support of measurement:

- Better understanding of relationships and impacts of models (e.g., object oriented design vs. functional decomposition, errors vs. different testing strategies, changes in specifications vs. changes in designs).
- Need more operational data to understand how development efforts impact the maintenance phase of the life cycle.
- Greater awareness of goals and progress among current metrics research programs (e.g., SEI, NASA, ARPA).

### 2.7.2 Recommendations for DoD Goals and Action

There was little support for standards or DoD-wide policy, but more support for an approach that would define recommended metrics only in high-level terms and then provide guidance for organizations to tailor those descriptions for each program's needs. (This is similar to the direction represented in SEI's current approach [Carleton 1992] and in CECOM's current guidebook [1993]. However, it was also remarked that the SEI approach is "too grandiose and hard to use.")

The need to validate metrics was stated by a few sources. Before a national effort is established to collect metrics, they say that metrics data quality and consistency needs to be proven within pertinent application domains. Rather than conceiving new measurement frameworks or models, there is a need to develop procedures for using existing models and to refine them from lessons learned. The following opinions were also expressed:

- A guidance and contracting package is needed to easily implement measurement during the software life cycle.
- Metrics should be tied to SEI process maturity levels.
- Metrics need to be added to Defense Systems Management Collage (DSMC) teaching subjects.

# 3. SUMMARY OF INDUSTRY METRICS DEVELOPMENT AND USE

This chapter provides an overall analysis of the software metrics programs of 17 organizations within 15 different companies. Of the 17 organizations, 10 are DoD contractors and 7 are commercial product producers. Four of the seven commercial product organizations are part of large companies that also have divisions involved in DoD contractor work. Throughout this chapter, the trends identified and conclusions drawn apply to the group as a whole unless a difference in pattern between DoD contractors and commercial organizations is specifically noted.

The intent of the industry survey was to focus on organizations with mature metrics programs so as to allow the DoD to capitalize on their experience. Fourteen of the seventeen organizations interviewed reported that the use of metrics was a routine practice within their organization. In comparison with the level of metrics usage reported in the 1992 *Software Measurement Practices in Industry* survey [SPRC 1992], these organizations qualify as being leaders in the use of metrics.

The amount and depth of information for the different organizations varies, based in part on their willingness to share detailed information that is often regarded as proprietary. All of the available information has been made use of in the analyses performed in this chapter. Appendix B contains metrics program summaries for 10 industry organizations. Summaries of the other organizations were omitted for various reasons, including agreements to treat information as proprietary, newness of the metrics program, and scantiness of available information.

## 3.1 GOALS FOR METRICS

Of the organizations interviewed, three gave information primarily on their corporate-level metrics set, that is, on the metrics set that is reported to the corporate-level of the company. The remainder gave information on the metrics set used at the project level. Regardless of which metrics set was described, many gave information on the goals and usage of metrics at several levels of their organization.

Table 3-1 shows the goals or uses to which metrics are put at different organizational levels and the number of organizations espousing each goal at the given organizational level (note that most organizations reported multiple goals). Goals that are similar have been grouped together and their counts combined.

**Table 3-1. Uses and Goals of Metrics**

| Uses/Goals of Metrics | Numbers Reporting at | | |
|---|---|---|---|
| | Project Level | Divisional Level | Corporate Level |
| Project management<br>Risk management<br>Project status summary | 14 | 1 | 1 |
| Cost estimation<br>Schedule estimation<br>Estimation model calibration<br>Estimation process improvement<br>New proposal estimation | 10 | 3 | 1 |
| Identify/Track process improvement<br>Validate process improvement<br>Increase SEI Capability Maturity Model (CMM) level<br>Benchmark | 6 | 4 | 4 |
| Improve product quality<br>Reduce costs<br>Increase productivity<br>Improve cycle time<br>Improve delivery time<br>Increase on-time delivery<br>Provide basis for management by objectives | 8 | 9 | 5 |

In looking for trends, the data should be examined within an organizational level because many more organizations gave goals for project level metrics than did for divisional or corporate level metrics. At the project level, the heaviest emphasis is on project management and estimation goals, although there is interest in process improvement and other business improvement objectives as well. At the divisional level, there is still some emphasis on estimation and process improvement, but over 50% of the emphasis is on the group of specific business improvements that affect profitability. At the corporate level, process improvements and specific business improvements are almost exclusively emphasized. These trends hold in both the DoD and strictly commercial organizations.

## 3.2 METRICS IMPLEMENTATION

Several influences have factored in the establishment or revitalization of the metrics programs reported on in this study, as shown in Table 3-2.

**Table 3-2. Influences on Start-up of Metrics Programs**

| Metrics Program Influences | Number of Programs Citing Influence |
|---|---|
| Benefits to be gained | 6 |
| Software process improvement efforts | 5 |
| SEI assessments | 2 |
| Quality initiatives | 2 |
| Integrated product teams | 1 |

While several of the metrics programs originated in the 1980s or before, eight have started within the past five years. Table 3-3 shows startup dates for the various metrics initiatives. Different initiatives started with different activities. Some started in conjunction with the startup of software process improvement activities. Others started by defining a metrics set, forming a metrics working group, creating a metrics handbook, or simply by starting to use metrics.

**Table 3-3. Significant Dates for Metrics Initiatives**

| Date | Number Initiatives Started |
|---|---|
| before 1978 | 2 |
| 1978 -1988 | 3 |
| 1989 | 1 |
| 1990 | 4 |
| 1991 | 2 |
| 1992 or later | 1 |

Ten of the programs have policy requiring the use of metrics, six have documented corporate metrics standards, six have documented organizational-level metrics standards, and several have documents providing guidance for the collection and reporting of metrics. Only one organization reported that they did not use any form of metrics document.

Most of the organizations reported the early formation of a group focused on metrics. This group typically provided metrics support to project groups in one or more of the following ways: metrics training, tool development, repository maintenance, and high-level analyses of metrics data across projects. Often the metrics group was composed of representatives from across the organization, who were instrumental in bringing metrics practices back to their home group. Very often the metrics group authored the metrics standards and guidance documents, incorporating project metrics experience into later revisions. The formation of such a dedicated group appears to be a "best practice" across the organizations interviewed.

As far as the metrics sets themselves, Table 3-4 shows the technical influences reported. The Goal-Question-Metric paradigm is more influential than the four explicit citings of it suggest; several additional interviewees stated that their metrics are chosen to support the achievement of organizational goals. The table suggests that government-sponsored metrics efforts, especially the SEI CMM, have been a significant influence on the metrics sets adopted by industry. A few metrics sets were mentioned as being similar to an organization's own metrics set, although not a direct influence on it. These include the acquisition metrics set described by Rozum [1992a], the management metrics set described by Schultz [1988], and the NAVAIR metrics set [1992].

**Table 3-4. Citation of Prior Sources as Influences**

| Prior Document or Effort | Number of Citations |
|---|---|
| Software Engineering Institute metrics recommendations (e.g., [Carleton 1992]) | 2 |
| Software Engineering Institute Capability Maturity Model for Process Improvement [Paulk 1993] | 6 |
| Air Force Systems Command Pamphlets 800-14 or 800-43 [AFSC 1987] [AFSC 1986] | 1 |
| Rome Labs software quality factors [RL 1985] | 1 |
| Goal-Question-Metric Paradigm [Basili 1984] | 4 |
| Mitre Report [Mitre 1985] | 1 |
| Quantitative Software Management (QSM) Metrics [Putnam 1992] | 1 |

## 3.3 BENEFITS

Companies start up metrics programs because they are convinced that metrics will be beneficial to them. Any metrics program of long standing must be presumed to actually deliver benefits beyond their cost, or they would not be supported. The trend among the organizations interviewed was toward an increased usage of metrics. Support and enthusiasm for metrics usage appears to be strengthening at the highest levels of their organizations.

The organizations reported realized benefits from metrics usage (Table 3-5). The second column of the table shows the number of organizations reporting a particular benefit. Some of the entries are somewhat redundant, but are included to reflect the way the benefit was stated. The benefits are derived not from metrics per se, but from the actions that are taken in response to the information provided by metrics. The metrics are crucial to provide direction and feedback, but without action no benefits will occur. A few organizations were able to report on benefits in numerical terms.

- A 30% reduction in defect rate found during formal test; 60% reduction found during system test.
- A 50-times reduction in defect density of released software over three and a half years.
- Progress rate improvement from 50% to 95% in three and a half years (a progress rate of 50% means that it takes twice as long to complete a project as originally estimated).

One organization reported the following return-on-investment information: software process improvement (resulting partly from metrics information) increased productivity 12% and saved $1.26 for every $1.00 invested, from 1991 to 1993. Another reported that the cost of the metrics program in two divisions was 1% of personnel costs.

## 3.4 METRICS AND REPORTS

This section will present information about the basic and calculated metrics in use in the organizations surveyed, metrics set sizes, metrics reports, and the metrics that organizations found most beneficial.

Different organizations use the word "metric" differently. One organization might refer to a LOC count as one metric, while another might refer to a graph containing estimated, budgeted, and actual LOC counts for developed, reused, and commercial off-the-shelf (COTS) software as one metric. To bring a little consistency to comparisons between different metrics sets, the following definitions for basic metrics, calculated metrics, and reports will be adopted:

- "Basic metric" will be used to denote a measurement of something real, that can be directly measured. Budgeted or estimated values are not counted as basic metrics because they aren't measuring anything that exists.
- "Calculated metric" will be used to denote a function of two or more different basic metrics. Productivity and error density are two common calculated metrics. Note that a percentage is considered to be a basic metric because it is a function (ratio) of two values of the same basic metric.
- "Reports" are collections of basic metrics, calculated metrics, and/or budgeted, estimated, or other non-measured information appearing together.

**Table 3-5. Realized Benefits of Metrics Usage**

| Benefits | Number of Programs Citing Benefit |
|---|---|
| Calibration of cost models | 2 |
| Estimating new proposals or planning new projects (effort, time, quality) | 5 |
| Basis for policy on sizing of new projects | 1 |
| Determining best mix of people | 2 |
| Assessing impact (for customers) of requirements changes | 1 |
| Assessing impact of different design techniques | 1 |
| Basis for process improvement | 4 |
| Improved ship-acceptance criteria | 2 |
| Cost reductions due to improved product quality | 1 |
| Motivation for good performance from projects (corporate-level benefit) | 1 |

### 3.4.1 Basic and Calculated Metrics

When broad categories of metrics are considered, commonality of measurements used in different organizations is apparent. At the detail level of individual metrics, however, there is a great deal of diversity. Table 3-6 shows 17 categories of basic metrics, the total number of organizations using one or more metrics in the category (out of 17 total), the number of DoD contractors using one or more metrics in the category (out of 10 total), the number of commercial organizations using one or more metrics in the category (out of 7 total), and the number of usages of metrics in that category across all the organizations. The number of usages of metrics in a given category is counted as the number of distinct metrics within that category used by an organization. Thus the first entry in the table gives the following information: 12 out of 17 organizations are using some form of cost metric. That number includes 8 out of the 10 DoD contractors and 4 out of the 7 commercial organizations. Further, the number of usages of all forms of cost metrics, across all organizations, is 13. Note that a few organizations reported their metrics usage only by broad category (e.g., "we measure cost"), in which case their usage in that category is counted as 1.

Table 3-6 shows that most organizations are using cost, effort, schedule, size, and defects metrics. The fifth column indicates that defect metrics are the most numerous, followed by size and progress metrics.

Examination of Table 3-6 shows that there may be some differences between DoD contractor and purely commercial organizations in the patterns of their metrics usage.

- Progress: All but two of the DoD contractors use progress metrics while only one commercial organization uses them.
- Requirements: Six out of seven of the organizations using these metrics are DoD contractors. A very plausible explanation for this pattern would be greater instability in DoD requirements versus commercial product requirements.
- Computer utilization: All of the organizations using these metrics are DoD contractors. Further, 80% of all DoD contractors surveyed use this category of metric.
- Release capability: These metrics measure the amount of functionality delivered per release or build. All of the organizations using them are DoD contractors.
- Staffing: All of the organizations using these metrics are DoD contractors. Of all the DoD contractors, 90% use this category of metric.

Table 3-6. Basic Metrics Usage by Category

| Metrics Category | Total No. of Organizations | No. of DoD Contractors | No. of Commercial | No. of Usages |
|---|---|---|---|---|
| Cost | 12 | 8 | 4 | 13 |
| Effort | 13 | 9 | 4 | 24 |
| Schedule | 17 | 10 | 7 | 23 |
| Size | 15 | 10 | 5 | 38 |
| Progress | 9 | 8 | 1 | 48 |
| Defects | 16 | 9 | 7 | 69 |
| Inspections | 5 | 4 | 1 | 6 |
| Requirements | 7 | 6 | 1 | 32 |
| Software Quality | 4 | 2 | 2 | 9 |
| Customer Satisfaction | 3 | 1 | 2 | 22 |
| Cycle Time | 6 | 3 | 3 | 11 |
| Computer Utilization | 8 | 8 | 0 | 42 |
| Development Environment | 8 | 6 | 2 | 29 |
| Release Capability | 4 | 4 | 0 | 5 |
| Product Performance | 1 | 0 | 1 | 1 |
| Staffing | 9 | 9 | 0 | 19 |
| Reuse | 1 | 0 | 1 | 3 |

Table 3-7 expands on the usage of size metrics by listing the variations encountered and the number of organizations using each variation. It illustrates the diversity in metrics definitions and usage among commercial organizations.

The category of defect metrics contains the greatest diversity of individual metrics. The basic defect metric is a count of the number of problem reports, but this metric is refined in various ways. First, problem reports, which refer to any type of requested change including enhancements, may be distinguished from defects, which refer to actual implementation errors or bugs. Other distinguishing factors include status (e.g., open, approved, implemented, closed), age (calendar time from initial report to closure or to date), the phase the error was introduced (e.g., requirements, design, code, problem fix), the phase the error was detected (with some organizations placing special emphasis on errors detected after release by customers), and the severity (or priority) of the problem. Very often, multiple factors are used to refine a defect count. Some examples of these types of defect metrics include number of open problem reports by severity level, number of customer-reported problem reports with age greater than *x* months, and number of open problem reports found during integration test by priority.

**Table 3-7. Size Metrics Usage**

| Size Metrics in Use | Number of Uses |
|---|---|
| Size (no further details given) | 2 |
| Number of Requirements (typically 'shalls') | 2 |
| Number of Design Units | 2 |
| Number of Lines of PDL | 1 |
| Number of LOC (generic) | 8 |
| Number of Developed, Non-Comment Logical LOC | 4 |
| Number of Modified, Non-Comment Logical LOC | 2 |
| Number of Reused Non-Comment Logical LOC | 3 |
| Number of Unchanged Non-Comment Logical LOC | 1 |
| Number of Maintained Non-Comment Logical LOC | 1 |
| Number of Delivered Non-Comment Logical LOC | 1 |
| Number of COTS Non-Comment Logical LOC | 1 |
| Number of Developed Non-Comment Physical LOC | 1 |
| Number of Reused Non-Comment Physical LOC | 1 |
| Number of COTS Non-Comment Physical LOC | 1 |
| Number of Total Assembler-Equivalent Non-Comment Physical LOC | 1 |
| Number of Function Points | 5 |
| Documentation Size | 1 |

Table 3-8 lists some of the calculated metrics used by the organizations, along with their usage counts. Error density and productivity metrics are the two single calculated metrics most commonly used. The majority of the calculated metrics deal with defects. Metrics are collected periodically, with monthly collection being most often reported. Corporate-level metrics are likely to be collected less frequently, e.g., quarterly. In some cases, frequency of collection varies based on the particular metric while one organization bases it on the CMM maturity level of the project.

**Table 3-8. Calculated Metrics Usage**

| Sample Calculated Metrics in Use | Number of Uses |
|---|---|
| Error density | 14 |
| Productivity | 7 |
| Schedule variance (e.g., budget for work performed minus budget for work scheduled) | 3 |
| Cost variance (e.g., budgeted cost of work performed minus actual cost of work performed) | 2 |
| Customer severity days (severity of customer problem multiplied by days open, summed by severity level) | 2 |
| Problems per user-month | 2 |
| Cycle time rate (e.g., calendar months divided by code size) | 2 |
| Effort estimation accuracy | 1 |
| Schedule estimation accuracy (actual duration divided by estimated duration) | 1 |
| Review effectiveness | 1 |
| Mean time to defect after release (takes into account exposure time) | 1 |
| Defect containment effectiveness (no. of defects removed after internal review but before release divided by (no. of defects removed after internal review but before release + no. of defects remaining after release) | 1 |
| Phase containment effectiveness (no. of defects found during phase review divided by (no. of defects found during phase review + no. of defects found after phase review) | 1 |

Granularity of the metrics collected also varies. For example, size can be collected for an entire software product, for each CSCI, for each CSC, for each CSU, and so forth. Effort and cost may be broken out by software life cycle product; productivity by language. Defect density may be collected over a whole life cycle or broken up by life cycle phase (requirements, design, code, test). Often, granularity decreases as reports are generated for higher levels of management. Granularity also may be affected by organizational maturity.

### 3.4.2 Metrics Set Size

Variations in metrics set size are shown in Table 3-9. To be consistent across organizations, metrics set size was determined by counting the number of basic and calculated metrics, as defined above, contained in the metrics set. The result rarely coincided with the size of the metrics set as reported by the organization. Fairly exact counts were possible for seven organizations, based on internal documents. Their counts are shown in the second column of the table. Their counts plus the counts for eight additional organizations that described their metrics set during interviews are shown in the third column. Data for two organizations were too incomplete to count.

**Table 3-9. Metrics Set Size**

| Size Range | Number of Metrics Sets (exact count) | Number of Metrics Sets (total count) |
|---|---|---|
| 1- 10 | 0 | 4 |
| 11 - 20 | 0 | 3 |
| 21 - 30 | 1 | 1 |
| 31 - 40 | 3 | 4 |
| 41 - 50 | 3 | 3 |

### 3.4.3 Metrics Reports

In some organizations, metrics are reported in combination with planned, budgeted, estimated, or otherwise generated values for the same objects. The first two of the three sample reports below illustrate the use of metrics in combination with planned values.

Metrics reports often summarize a large amount of basic measurement data in a manner that aids in the comprehension of basic trends. Figure 3-1 and Figure 3-2 show sample staff hour reports, broken down by month. The planned number of staff hours is plotted against the actual numbers of straight staff hours, compensated overtime staff hours,

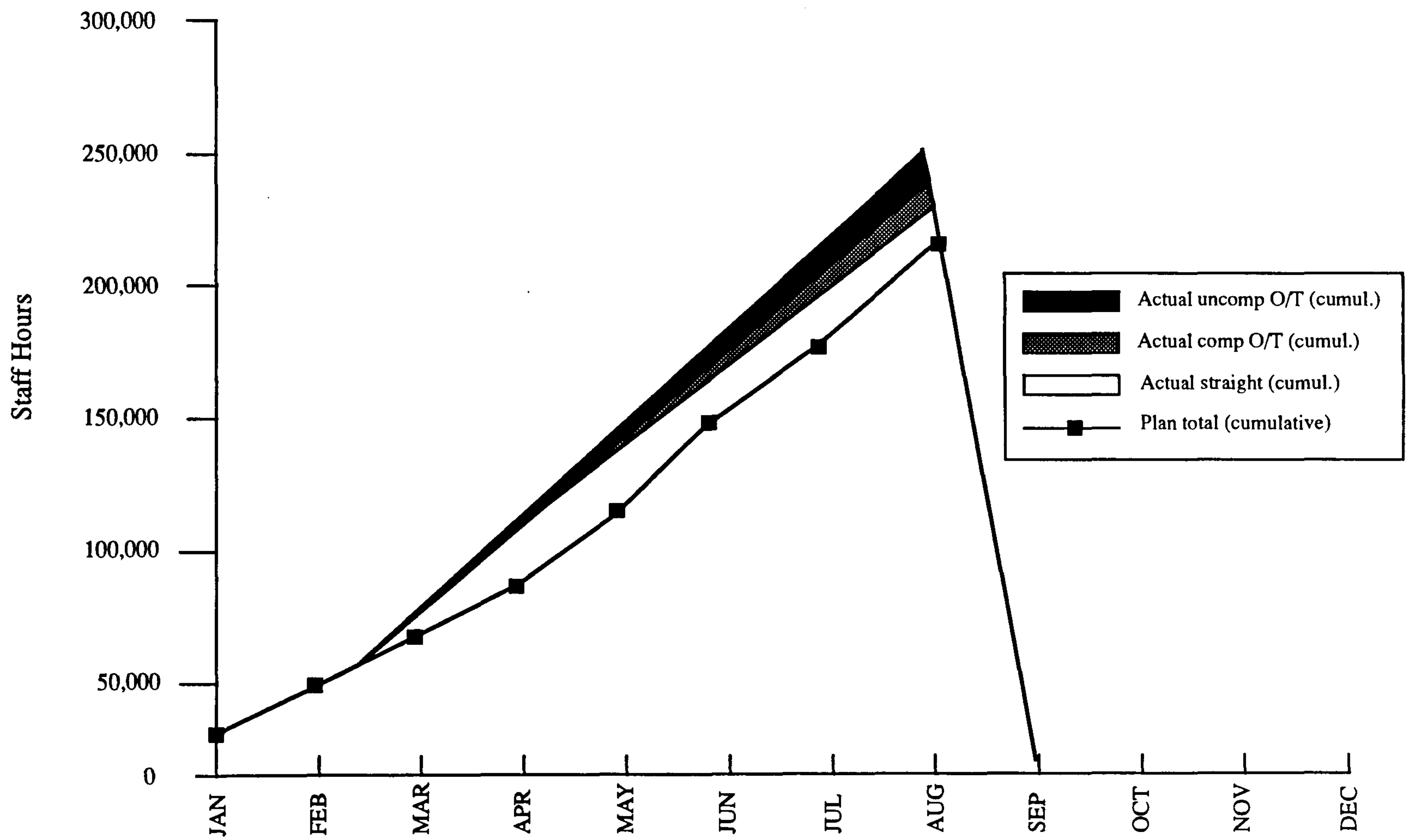



Figure 3-1. Software Staff Hours by Month

OT Overtime

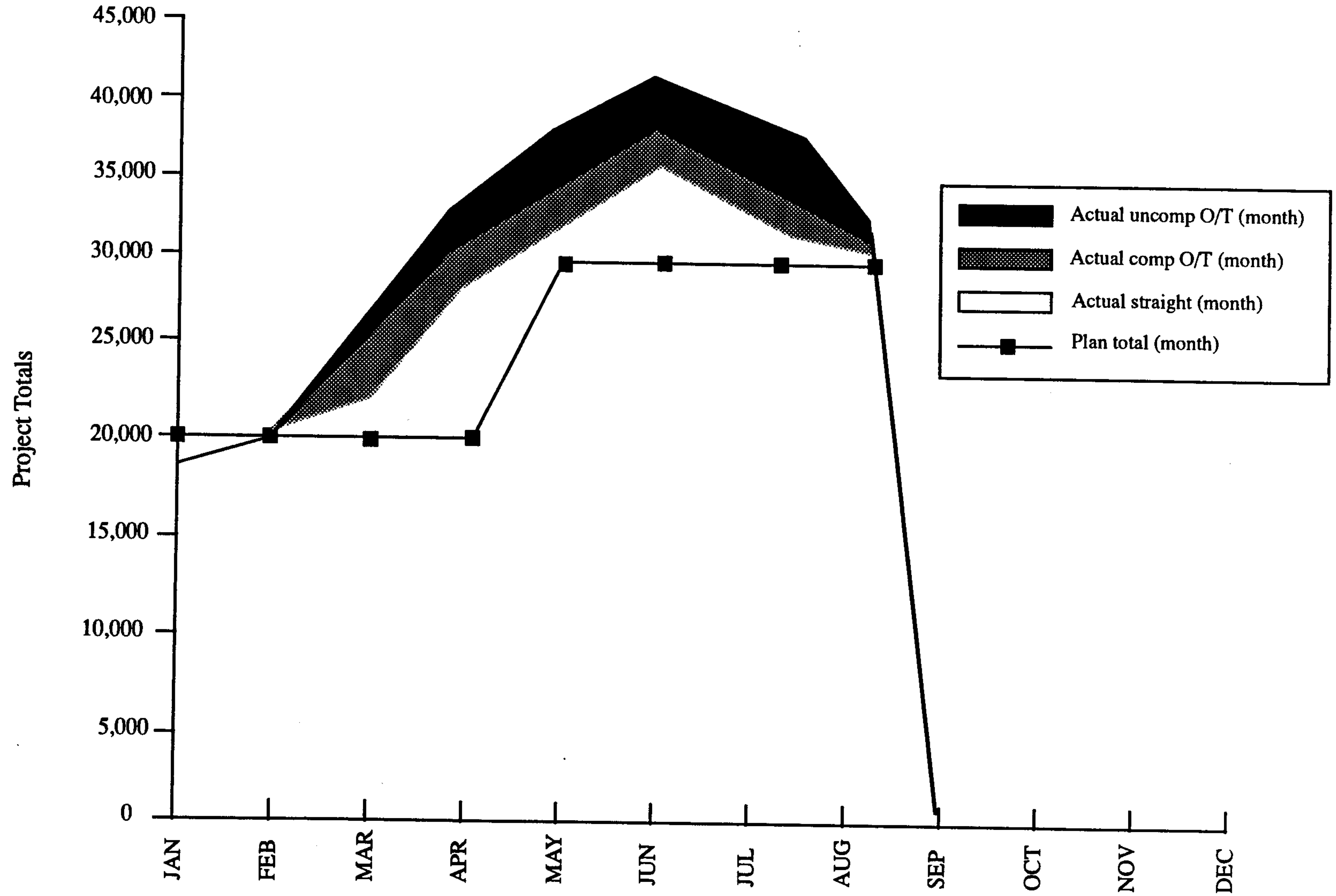



**Figure 3-2. Cumulative Software Staff Hours**

OT Overtime

and uncompensated overtime staff hours.

The report in Figure 3-3 shows problem resolution time. The histogram portion shows the number of open problems per month, broken down by the length of time they have been open. The numbers along the left axis provide the scale. The line plots show the average number of days to reach a solution for problems that have been closed during the month, broken down by problem severity level. The number along the right axis provide the scale.

### 3.4.4 Most Beneficial Metrics

Table 3-10 shows the metrics and reports mentioned as being most beneficial. Some organizations did not value any particular metrics above the others. One interviewee reiterated that the most valuable metrics would depend on an organization's goals. The metrics shown in this table offer no surprises and indicate that basic metrics, rather than esoteric, are considered most valuable.

## 3.5 TOOLS AND REPOSITORIES

This section describes the use of data collection and analysis tools and data repositories among the industry organizations contacted.

### 3.5.1 Tools

Tools are not universally perceived to be a problem. Some organizations are well served by simple tools, as evidenced by the following remarks:

- Much of the required metrics set can be collected using routinely available tools. For example, time and effort is collected from time cards and tracked using the corporate MIS [Management Information System] tool; the current WBS [Work Breakdown Structure] covers 90% of the information needed. The problem reporting system contains the information needed to produce defect metrics.
- Simple tools are used. The emphasis is on keeping the data organized and using it daily).

Others use more complex tools, but are able to purchase them (e.g., Putnam's QSM tool set) or are able to develop them. One organization recently completed a two-year development effort on an in-house tool to aid in data entry, metrics calculations, and chart and report production.

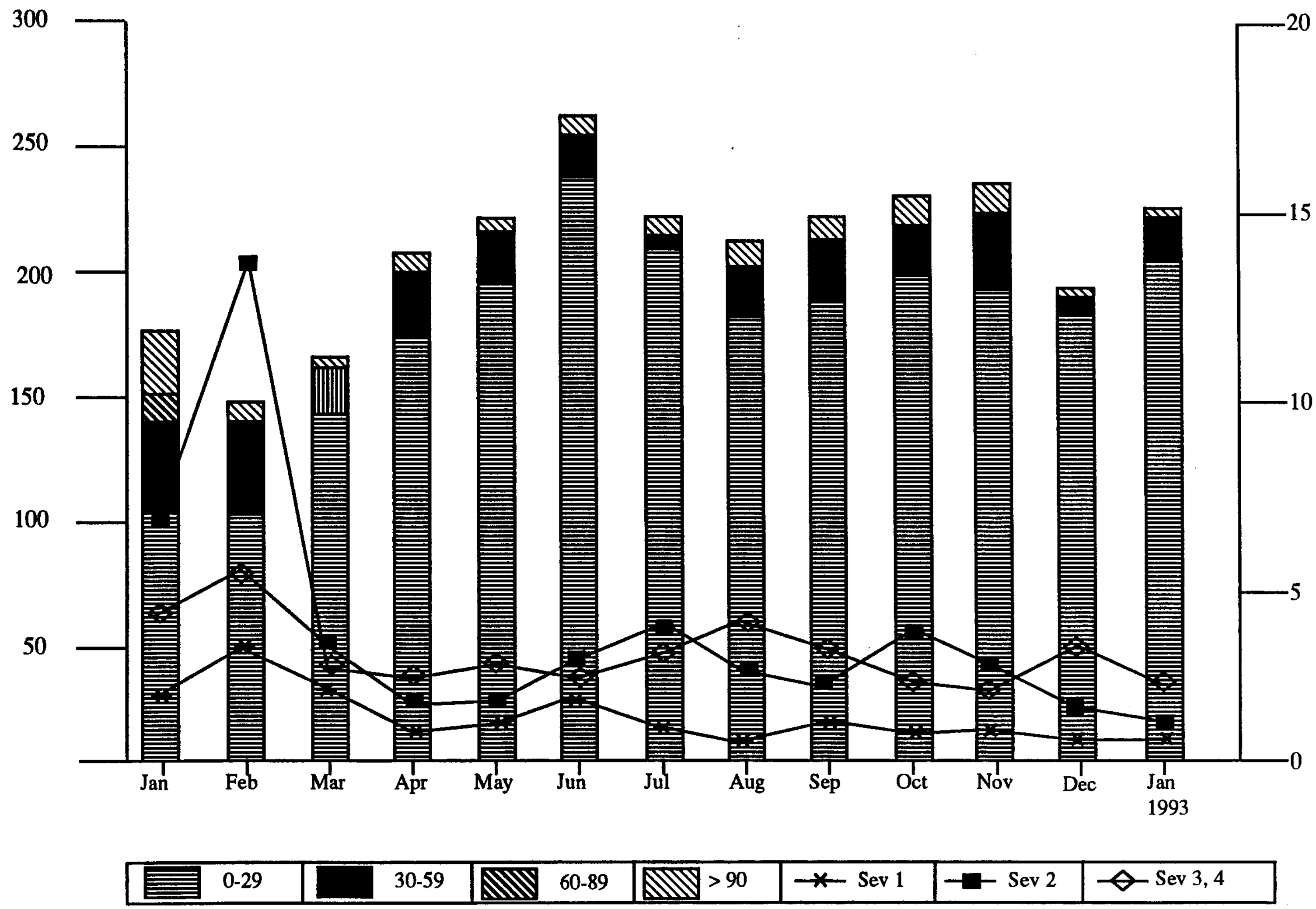



**Figure 3-3. Problem Resolution Time**

On the other hand, many organizations are not able to purchase the tools they want and have not yet developed them. By their remarks, they identified tools as one of the major problem areas in their metrics program (see Section 3.4.3) and an area for further research and development (see Section 3.5).

**Table 3-10. Most Beneficial Metrics**

| Metrics/Reports | Number of Responses |
|---|---|
| Progress indicators | 2 |
| Rate chart - planned vs. actual production of units, for all life cycle phases | 1 |
| Time, schedule | 1 |
| Cost | 2 |
| Manpower loading | 1 |
| Software size | 4 |
| Requirements stability | 1 |
| Defects | 3 |
| Inspection Metrics | 1 |
| Resource Utilization | 1 |
| Customer Satisfaction | 1 |

Tools are needed to collect basic data, to calculate metrics, and to generate charts, graphs, and other forms of metrics reports. Table 3-11 shows some of the commercial tools and Table 3-12 some of the proprietary tools currently being used for these purposes. Of the commercial tools available, spreadsheets, databases, and cost estimation models have the most widespread usage (scheduling tools are not perceived to be metrics tools), while the proprietary tools most often developed are LOC counters and report generation aids. One major problem in producing good commercial metrics tools is the large number of proprietary systems and procedures with which they have to interface.

### 3.5.2 Repositories

Most organizations report storing their metrics data in either paper or electronic form. Table 3-13 shows the numbers reporting usage of paper or electronic storage at either a project level, divisional level, or corporate level. Some organizations reported storage at multiple levels. Most electronic storage is within a database, but some storage is kept in spreadsheets or flat files.

**Table 3-11. Commercial Tools in Use**

| Commercial Tool | Number of Usages Cited |
|---|---|
| Spreadsheets (e.g., Excel, Lotus 123) | 7 |
| Databases (e.g., Paradox, Oracle) | 3 |
| Cost Estimation Models (SEER, COCOMO) | 4 |
| QSM Tools: SLIM, SLIM Control, Size Planner, PADS | 1 (+1 investigation for future use) |
| Requirements Tracking (RTRACE) | 1 |
| Filemaker Pro | 1 |
| Harvard Graphics | 1 |
| Complexity Measures (Logiscope) | 1 |

**Table 3-12. Proprietary Tools in Use**

| Type of Proprietary Tool | Number of Usages Cited |
|---|---|
| Report Generation Aid | 5 |
| LOC Counter | 4 |
| Tools That Aid in Data Entry | 2 |
| Tools That Aid in Data Collection | 2 |

**Table 13. Metrics Data Storage**

| Type of Storage | Project Level | Divisional Level | Corporate Level |
|---|---|---|---|
| Paper | 0 | 2 | 0 |
| Electronic | 5 | 9 | 3 |

Project-level metrics data is the primary type of information stored at the project level, although one organization also reported storage of historical data at this level. At the divisional or intermediate level, the types of information stored include historical data, financial data, trend analyses, divisional-level metrics reports, planned values for required metrics, and in one case, project inspections information and problem change report information. At the corporate level, project historical data, project characteristics data, project cost data, and corporate-level metrics reports are stored.

Historical data provides a basis for estimating the cost, effort, and duration of new projects. Estimations are made by examining data from past projects with similar characteristics to the one being estimated. It is therefore important to store project characteristics data along with historical metrics data.

The types of characteristics data reported being stored include sponsoring agency, contract type (e.g., cost plus, firm fixed price), project size, application domain, name of life cycle process required, standards used, development environment, implementation languages used, and target hardware. One organization reported using 53 items of characteristics data.

## 3.6 BEST PRACTICES AND LESSONS LEARNED

This section presents the best practices and lessons learned by the organizations interviewed. The best practices recommendations have been paraphrased and organized around five themes: goals, motivation and trust, improvement focus, simplicity, and guidance and support. Following the best practices sections, the main problems or issues faced by the organizations in establishing a metrics program are discussed.

### 3.6.1 Goals

Metrics are not the goal. The first thing that an organization must do in starting a metrics program is to clarify its real organizational goals. The metrics program must then be focused to support achievement of those goals. In particular, only metrics that contribute to the achievement of organizational goals should be included in the metrics set. Only data that is relevant to the goals should be collected, and all data that is collected must be used. Many metrics efforts founder when the people who must put effort into collecting data see that the data is not used or that it is not relevant to anything important.

### 3.6.2 Motivation and Trust

Motivation and trust are important and must be established. Most metrics programs do not start off having them. Because of people's fears regarding measurement, just one misuse of metrics data can destroy all trust that has been built up. One recommended way to start building motivation and trust is to involve the user community in the metrics start-up activities, including metrics set selection. This has the added advantage of bringing hands-on experience to the initiative.

Motivation can also be supplied by having a metrics edict in place (the stick) and by being able to provide quick benefits from the metrics program (the carrot). Any metrics analysis results should be shared with the people who contributed to the underlying data collection. To motivate someone to participate and collect data, one person recommended using actual metrics data and showing the person how the data has been used and how it has been beneficial.

### 3.6.3 Improvement Focus

Metrics data should not be used to compare dissimilar projects or separate individuals. For one thing, comparisons (of productivity measures, for example) across application domains are not valid. The large differences in the things being compared make the comparison meaningless. Most organizations that are using historical data to estimate new projects use only similar projects to formulate their estimations. The best comparisons to make are to compare self against self over time.

Most importantly, a metrics program should make clear that its commitment is not to comparison but to improvement. Problems illuminated by metrics data should be discussed in terms of how to best remove them, and in terms of process causes rather than people causes.

### 3.6.4 Simplicity

Several people recommended a small metrics set of basic measures, especially at the start of a metrics initiative. A small metrics set lessens the impact of data collection and reinforces a focus on the most relevant metrics. This in turn makes it easier to get buy-in for a new metrics program.

Several people recommended using simple metrics. For example, using straight defect totals was cited as being preferable to calculating defect density. Reasons given for eliminating the LOC divisor included LOC differences among different languages and simple confusion. Larry Putnam, president of Quantitative Software Management, warns against productivity measures such as LOC/person-months and function points/person-months because of the non-linear relationships of effort and schedule to LOC. He generalizes by stating that ratios of metrics tend to be dangerous when used in isolation (as productivity measures often are).

Several best practices dealt with the idea of graduated metrics sets. Larger projects or more mature organizations may want or need larger metrics sets. Several organizations have evolved rather large metrics sets. One, whose organization has used metrics for over 20 years, recommends sets such as Rozum or Schultz have defined. "Anything less is not enough to run projects."

Finally, metrics sets evolve over time, as organizations evolve and mature. Metrics organizations should not waste time trying to come up with the perfect metrics set that satisfies everyone. They should adopt a reasonable set that addresses their goals and let time and experience perfect it further.

### 3.6.5 Guidance and Support

Most organizations have established a focus group for metrics to provide guidance and support in several areas including training, tools, and general consulting. As a first step in helping groups automate their data collections and analysis, one group surveyed its company to determine which tools were already being used successfully, and then provided this information to the projects.

One important component of guidance is to develop simple, crisp, and exact metrics definitions, to develop guidance for metrics usage, and to disseminate this information in documents. When people are inexperienced in metrics, they need clarity and simplicity to be able to participate in metrics collection and usage. Documentation contributes to uniform practices across a group and facilitates communication within the group.

### 3.6.6 Issues and Problems

Tools were identified as one of the largest problem areas in metrics. Many groups expend a lot of resources in developing tools to collect data, count LOC, and generate the reports they need. Tools that are available commercially often collect data and do analyses that are not relevant to the organization. Tools that are not available commercially include tools to count function points and tools that would interface seamlessly into the group's development environment.

Several programmatic problems were mentioned. Getting buy-in for the metrics program and for data collection in particular is difficult. There are problems with getting subcontractors who are also competitors to share their metrics data. Complying with different metrics requirements from different agencies also causes difficulties within an organization.

Finally, some problems with specific metrics were mentioned:

- It is difficult to predict LOC and difficult to reestimate them when requirements change.
- LOC is not satisfactory for counting fourth generation languages (4GL) or object-oriented languages.
- Quantifying staff experience is difficult.

## 3.7 FUTURE PLANS AND DIRECTIONS

This section identifies research and development areas that the organizations are addressing or recommending for consideration by some other organization to address. It also contains advice offered to the DoD regarding the use of metrics in its software acquisition process.

### 3.7.1 Research and Development

The future plans and directions reported by the organizations can be categorized as (1) plans or ideas for progressing or evolving their metrics program, (2) recommendations for investigating new or revised metrics definitions, and (3) recommendations for other metrics research and development activities. However, a strong caveat was also issued concerning research and development in metrics: *Metrics buy-in, training, and support, rather than new metrics definitions or research, need attention and provide the greatest benefit.*

Examples of organizational plans for evolving a metrics program giving insight into the current status of metrics programs are listed in Table 3-14.

Suggestions and needs were identified for investigating new metrics in the following areas:

- Object-oriented metrics.
- An adaptation of or successor to function points or feature points to support real-time software, and to provide a measure of size independent of language.
- Reliability and reusability metrics.

Finally, general metrics problems needing further research and development efforts were identified:

- Identifying standard reports for the corporate level. How can information be abstracted in a valid manner across different application domains, and how could it best be reported?
- Investigating the impact of reuse and COTS software on metrics data and analysis.
- Evaluating the pay back of measurement. You can over measure. What metrics really matter?
- Creating tools and procedures to make measurement non-intrusive.

**Table 3-14. Organizational Plans**

| | |
|---|---|
| • Creating a corporate-level metrics standard for project-level metrics. | • Developing a guidance document on the interpretation of metrics. |
| • Collecting data in a central repository to create benchmarking ranges. | • Collecting and analyzing development costs and recurring costs across projects. |
| • Using metrics across programs to identify process improvement benefits. | • Best tools, and best methodologies. |
| • Creating a division-level electronic database. | • Expanding the use of metrics to hardware development and systems engineering. |
| • Creating integrated measurements, not broken out into system engineering, hardware engineering, or software engineering, to support integrated product teams | • Developing documentation metrics (from a producer of shrink-wrap commercial software; the only organization interviewed currently using a documentation metric is also part of a strictly commercial company). |
| • Looking for a productivity measure for object-oriented metrics, maybe function points. | • Analyzing feature points to see if they should be used. |
| • Looking at adding function points to the metrics set. | • Breaking metrics into sets based on SEI CMM level. |
| • Enlarging the metrics set to support Level 3 organizations. | • Identifying which metrics should be collected after the SEI core. |
| • Investigating algorithmic estimation tools. | • Building the ultimate logical LOC counter. |

### 3.7.2 Recommendations for DoD Goals and Actions

Interviewees were asked what advice they would give to the DoD regarding the use of metrics to improve the DoD software acquisition process. Some of the advice given was generally applicable to any metrics effort and has been included in the best practices section. The following advice is DoD specific.

- The best role of the DoD would be to offer guidance rather than requirements, and to standardize its guidance across agencies and services. The DoD's impact on contractors is often perceived as negative.
- A contractor's own process and process improvement should take precedence over a DoD mandate. Process improvement is difficult when different customers impose different standards on different projects.
- While a minimal amount of standardization might be beneficial, it should be at a high level. For example, the DoD could standardize on a core *set* of charts to be used on every program, but shouldn't standardize on the *definition* of the charts. Standardization at the level of atomic data and central repositories was strongly discouraged.
- Ensure that contracts have a WBS that supports whatever metrics the DoD requires.
- Do a pilot to verify usefulness of information you're collecting before disseminating DoD-wide.
- Use and acceptance of project level metrics depends greatly on the customer and the customer- induced culture. DoD Sub-Program Offices, personnel having two-year tours of duty, tend to require metrics showing good performance on their shifts. They require management to cost. Commercial customers typically are interested in both cost and time to market. Some non-DoD government customers, however, have long-term relationships with their projects. They focus on delivery of the right product rather than schedule or cost. Within this subculture, there is a lack of incentive to manage to cost and an accompanying lackadaisical attitude to metrics collection.

# 4. COMPARISON OF DOD AND COMPANY EFFORTS

This chapter compares DoD and company metrics activities in regard to the state of practice and lessons learned, with consideration of potential benefits to DoD weapon system acquisition. Successive sections address goals, implementation, benefits, metrics used and their reporting, tools and repositories, and research and development recommendations. A final section draws conclusions on the basic questions identified for this study.

## 4.1 GOALS FOR SOFTWARE METRICS

This study shows an important difference between DoD and company practices in regard to the intended users of metrics. DoD organizations predominantly consider their metrics users to be acquisition program managers and their office staff. Because program offices spend much effort in oversight and direction of a development contractor or contractor team, metrics recommended by DoD organizations focus on assessing a contractor's software development status. Consequently, DoD metrics are similar in scope to contractors' practices, yielding high potential for conflict but little if any added information that is clearly pertinent to acquisition decisions. Guidance accompanying DoD metric definitions is minimal to nil regarding metrics use in program management decisions. DoD metrics proponents do not recognize adequately that an acquisition program manager has a different perspective and decision-making role than a contractor's software development manager.

But it should not be forgotten that metrics may be important also for DoD in-house developers doing PDSS (post-deployment software support) on weapon system software or building engineering and management applications that are crucial for weapon system acquisition. This user community is not particularly recognized in DoD metrics guidance. Thus, DoD organizations have not effectively identified and accommodated various classes of metrics users.

In industry, individual project managers now are the primary metrics users. Usually they retain their project's voluminous and detailed metrics data for their project monitoring, control, and planning purposes. But business sector (or application domain) managers are emerging as important and major users also. Company metrics increasingly contain productivity and quality factors that are derived from project data and reported to higher organizational levels in the company to assist process and product improvement planning. Companies are using metrics to assess their return on investments on such things as software capabilities, training, and measurement itself.

The DoD-versus-industry differences are most visible when practical metrics uses are examined rather than goals stated in guidelines and published papers. Figure 4-1 summarizes the survey information regarding how metrics are actually being used by the two groups of organizations.

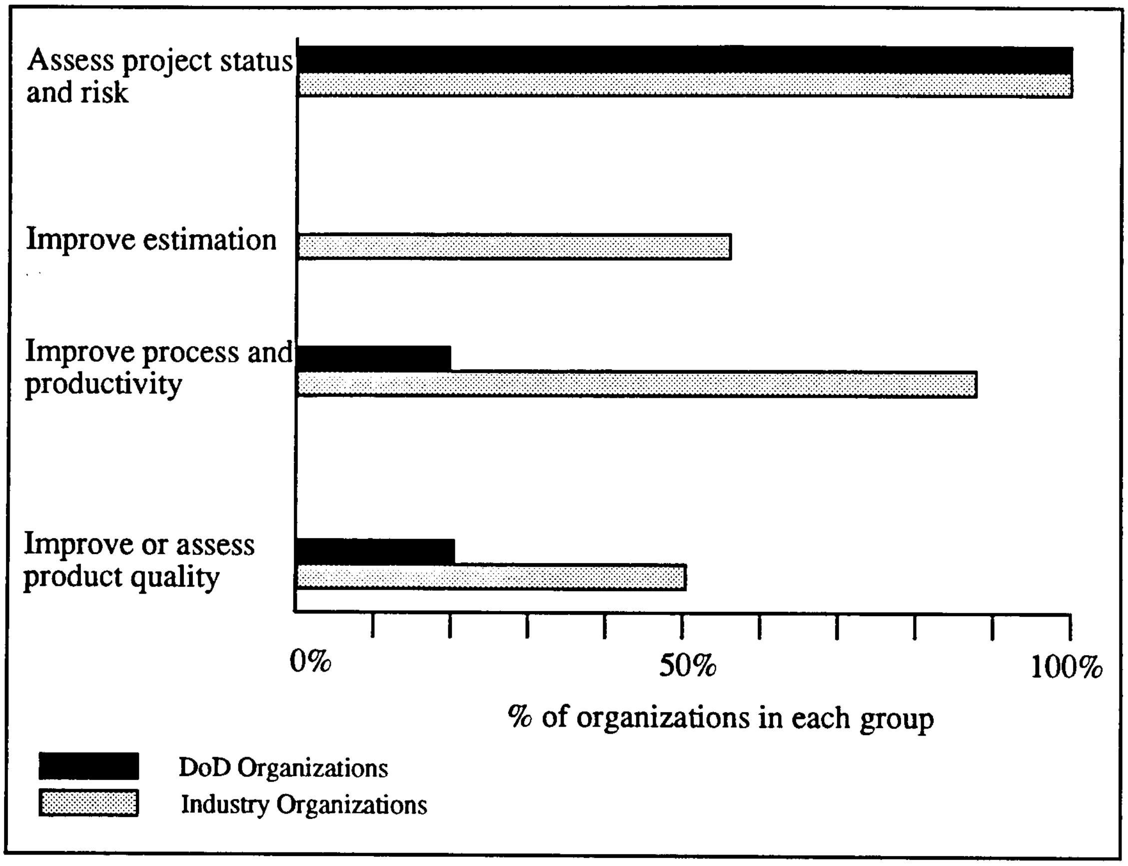



**Figure 4-1. Comparison of Reported Metrics Usage**

Companies appear to be stressing the practical goal of increasing their business through development process improvement, including improved cost estimation and contract bidding support. Among DoD contractors, this trend may be due partly to DoD's use of software process improvement or total quality management criteria in awarding contracts.

No comparable objectives regarding DoD software acquisition are motivating greater scrutiny and use of metrics. But evidently there is concern that the evaluation of program manager performance would become the rationale behind expanding software metrics collection throughout DoD.

## 4.2 IMPLEMENTATION

This study also indicates a significant difference in implementation maturity between DoD and company metrics efforts. Companies have longer continuous experience and more evolved implementation artifacts such as handbooks, documented policies, and repositories. Only a small portion of DoD programs represent a continuous thread of experience predating 1985. Most can be characterized as recent attempts, repeated in some cases, to find a limited metrics set acceptable to many program offices in order to insert contractor measurement into acquisition practice.

No unequivocal successes were found in DoD metrics implementation through policy. About half of the DoD personnel contacted were convinced that a DoD or Service-wide policy to mandate standard metrics collection and reporting would be unacceptable. Program offices oppose reporting current measurements to outside organizations and oppose standard requirements that may conflict with their contractors' practices or capabilities, thereby introducing additional program costs.

Several contacts expressed the view that contractors should be defining metrics and not the DoD. But most also agreed that basic guidance on collecting and using metrics would be helpful to program offices.

Candid opinions given to us suggest that far less success is being achieved by the DoD activities contacted than was anticipated, even with a voluntary approach to putting metrics into practice. Along with lack of policy commitment and executive interest, comments pointed to decreasing funding for organizations trying to provide technical advisory and technology transfer services. Lack of training opportunities and resources also was identified as a problem.

Companies are not concerned with evolving finely honed metrics standards, although some have well-developed handbooks or well-organized standardization efforts. Companies are focused more on putting measurement into practice expeditiously, relying on centralized technical support groups rather than high quality guidance and specifications. They have an advantage in organizational unity and corporate-level performance goals that motivate progress.

## 4.3 BENEFITS

Although this study accumulated many software metrics publications, few of these publications documented noteworthy successes in DoD program management attributable to using software metrics. Some DoD activities are vague about specific programs that have applied their recommended metrics and received concrete benefits. ARDEC, ESC, NASA, and NUWC cited significant programs as examples of applying software metrics, e.g., JSTARS, Granite Sentry, CCPDS-R. It is not always clear whether these applications are succeeding as a direct result of government requirements and program manager initiative, or instead, is DoD simply accepting the benefits of independent contractor achievements and initiative? For JSTARS and CCPDS-R, the latter seems to be the case.

On the other hand, a number of successful experience reports have been published from industry describing achieved benefits in process improvement [Andres 1990, Royce 1990, Dion 1992, Daskalantonakis 1992, Kan, Wohlwend 1993]. The company interviews revealed concerted effort on assessing benefits and payoff, often through pilot projects, along with ambitious, time-phased, quantitative goals for process improvement. Actual improvement achievements may not be often published, but the evident extent of corporate investment is some reason to believe that actual benefits are worthwhile.

The overall conclusion is that there is little available evidence of government metrics recommendations leading to significantly improved program management. Perhaps the evidence simply has not been gathered and documented. Industry, on the other hand, is publishing benefit experiences and demonstrating that competitive advantages can be realized by applying measurements to improve development processes and products.

## 4.4 METRICS AND REPORTING

DoD organizations have been led by unsuccessful prior experience to downsize their metrics recommendations in order to find a minimal requirement that is acceptable to program managers. NASA related some experience that others also suggest—well-founded metrics definitions are ignored or followed inconsistently, while the most simply formulated metrics are argued to be unusable for lack of definition. Industry experience also suggests that simple metrics work out better, and that it is very important to understand who needs metrics and for what purpose.

The metrics currently recommended by DoD organizations are fairly similar in the underlying input data required. Differences arise in computed metrics, presentations, or reporting criteria, such as using a percentage rather than an absolute ordinal scale, or reporting certain metrics at weekly versus monthly intervals. Some metrics sets have requirements that are costly and hard to accept as a program management need. An example is requiring a SLOC impact assessment on all reported defects. An estimate of modified SLOC to repair a defect will not be accurate without doing most of the repair analysis. Also, it is not as valuable for setting priorities for defects for repair as the determination (also required) of a defect's impact on users and operations. The SLOC impact requirement perhaps illustrates government policy makers trying to micromanage an individual developer.

Although more data collection should not be a goal in itself, companies collect more metrics information than requested in current DoD recommendations. Some of the addition is management and financial data, and much of it is software technical or product data that indicates quality or productivity. More company projects depend on metrics for managing development than appears the case for organizations managing DoD programs.

DoD metrics are seldom reported above the individual program office. Companies are reporting summarized or derived metrics, not raw data, to the corporate or division level, where it is used for business and process research purposes. A prevalent use is calibrating software cost estimation models in order to fine-tune pricing for proposals. Figure 4-2 compares metrics reporting within DoD organizations and industry, with strictly commercial companies separated from DoD contractors. From such evidence it appears that DoD contractors have at least equally mature metrics programs as strictly commercial organizations.

For an overall picture of metrics use and reporting in industry, a representative taxonomy was developed during this study from several industry examples. This is given in Table 4-1 to illustrate the typical number and scope of metrics collected at a project level, a division level (multiple projects in a similar application domain or business line), and corporate level.

## 4.5 TOOLS AND REPOSITORIES

Companies are more active in assessing available tools and configuring metrics tool sets than DoD organizations appear to be. Those with more mature measurement programs believe that available tools are limited or unsuitable in some capabilities, and they are investing in new tool development. No significant metrics users among DoD organizations are investing in new tool development, with the exception of the Army STEP program.

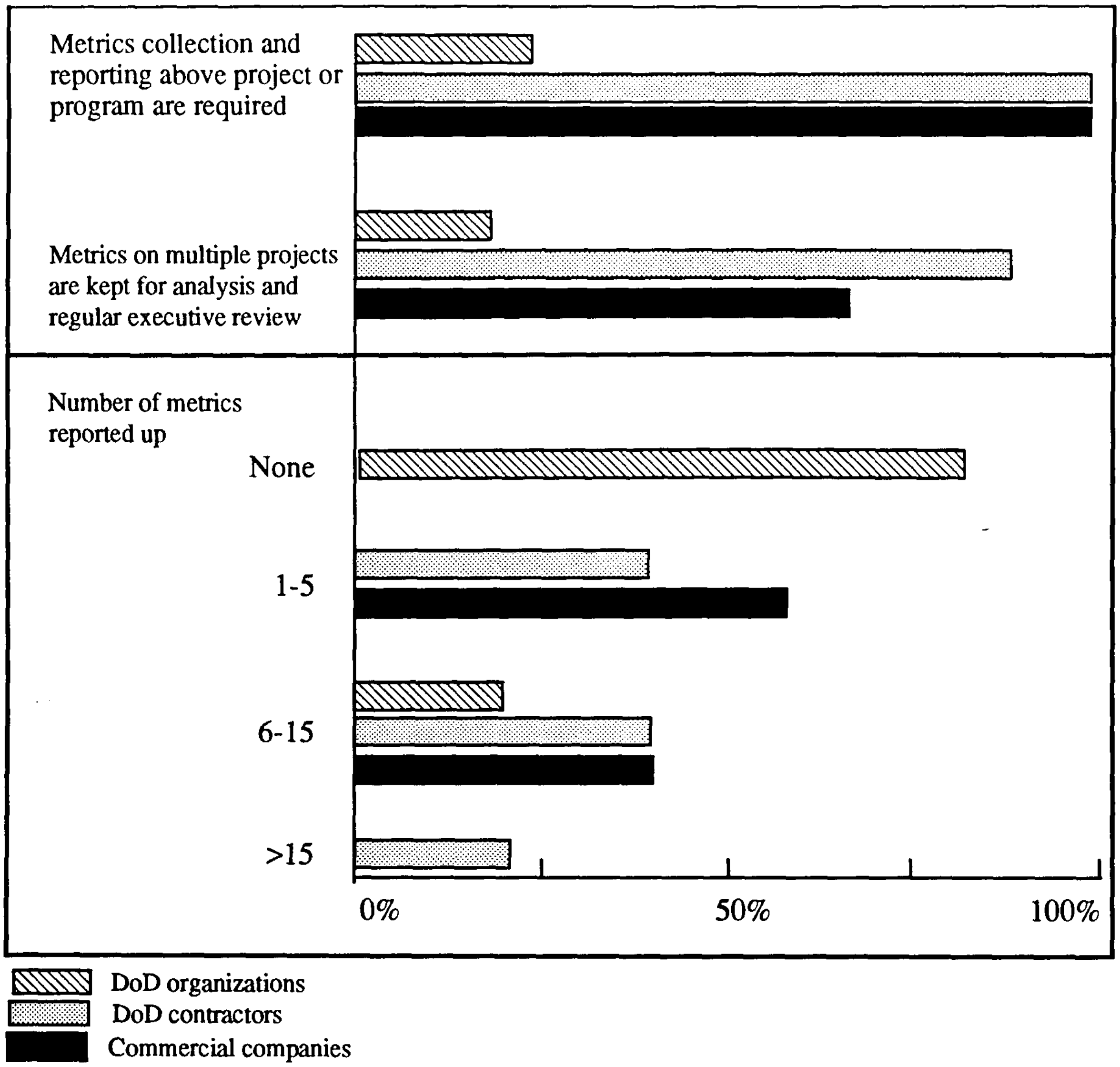



**Figure 4-2. Comparison of Metrics Reporting Practice**

Available COTS tools appear to be adequate for initiating metrics support for acquisition programs. A limited number of general suggestions were made for future tool research and development needs.

Only NASA SEL, among this study's contacts, is maintaining a computer repository across multiple programs. It is used for research and baseline planning data, not program evaluation. A COTS database system was used satisfactorily in that experience.

Companies tend to use multiple repositories to hold specific segments of metrics data for particular analysis and decision purposes. The more voluminous amount of data for project control and management is held within project organizations. COTS tools appear to suffice for repository needs.

**Table 4-1. Illustrative Taxonomy of Industry Metrics Usage**

| User - Purpose - Review Cycle | Assessment Areas and Metrics | | |
|---|---|---|---|
| | Project | Product | Process |
| Corporate Executive<br>• assess corporate capability<br>• quarterly | • productivity | • customer satisfaction<br>• shipped defect rate | • divisional maturity level<br>• improvement progress |
| Division or Business Sector Manager<br>• assess team capability and project performance<br>• monthly | • size<br>• effort<br>• cost<br>• schedule | • customer satisfaction<br>• shipped defect rate<br>• release capability<br>• reuse | • problem/ defect aging<br>• defect containment<br>• cycle time<br>• process improvement status |
| Software Project Manager<br>• assess product & process<br>• weekly | • size<br>• effort<br>• cost<br>• schedule<br>• staffing | • release capability<br>• traceability<br>• shipped defect rate<br>• stability<br>• code quality<br>• test coverage<br>• target computer utilization | • inspections<br>• in-process defects<br>• fix quality & cycle time<br>• reliability<br>• unit progress<br>• test progress<br>• documentation progress<br>• environment maturity |

## 4.6 RESEARCH AND DEVELOPMENT RECOMMENDATIONS

### 4.6.1 Recommended Research Toward Better Metrics Capabilities

This list highlights the suggested areas that look most important.

- Validation of metrics as to their benefits and relationships to other models
- Training
- Metrics for object-oriented methodology

- Tools to assist metrics collection and distribution across various developmental and managerial participants

### 4.6.2 Recommendations for DoD Action To Improve Measurement

The following succinct statements appear to capture the essence of recommendations offered on future DoD action.

- Improve available training and guidance on use of metrics.
- Provide wide latitude for local choices within any measurement policy.

## 4.7 CONCLUSIONS ON BASIC STUDY QUESTIONS

This section provides concise answers to the basic questions posed for the first phase.

**What factors distinguish company and DoD practice in software metrics?**

Within DoD organizations, the state of practice appears spotty and immature. It is not likely to improve significantly because of major barriers. These include lack of clear and practical goals consistent with user responsibilities, lack of documented success, lack of support for technology transfer and training, lack of understanding of appropriate metrics and their effective use, and apparently strong program office opposition to external reporting and to cost-adding requirements. Company practice is more mature and further evolved for both DoD contractors and strictly commercial companies. It is motivated by a clear and widely supportable purpose, i.e., improve competitiveness within company application or business domains for the sake of gaining new business. Companies are doing the ground work to understand their metrics users and their different purposes for having metrics.

**What problems are managers trying to address through metrics collected during the software development process?**

DoD managers are trying to assess the status of development contractors' work and to foretell problem situations. But metrics should not be the primary basis for detecting program management difficulties. Instead they should serve as auxiliary, quantitative data for assessing the eventual impact of undesirable trends or for assessing the benefits of a deliberately instituted development approach. DoD goals for software metrics appear to be inadequately conceived, and the proper role of metrics for program management is not well understood and communicated to users.

Companies are using metrics for different purposes at different organizational levels. At the performing project level, metrics describe work status, resource expenditures, and the myriad factors that apply in daily decision making and planning. At a division or domain level, metrics are used to synthesize trends related to narrow improvement issues. The prevalent example is calibrating cost estimation models. At the corporate level, metrics are used to assess status, benefits, and return on investment for process improvement effort.

**To what extent has the application of software metrics proven successful?**

Software metrics include basic management and status data that is essential to effective project management on a daily basis. Many acquisition programs can be cited where contractors collect and use metrics and report them to their DoD program offices. Analysis would show similarities with DoD metrics recommendations. Companies have published a number of success stories on using metrics to guide improved development processes and practices. The extent of metrics use within commercial companies and DoD contractors is convincing evidence that software developers see value in software measurement.

It was not a study objective to determine whether government requirements or contractor initiative has been the main force behind metrics use in defense software development. However, little evidence was found of successful acquisition program management directly attributable to use of software metrics.

**Does company practice reveal metrics capabilities, concepts, or tools that could benefit DoD's software acquisition process?**

The trend in company metrics use is to gain long-range and company-wide marketplace benefits, not merely to solve short-range problems in each project. As part of their approach, companies are establishing policies for metrics collection and technical support groups for institutionalizing measurement. They also are investing in building or acquiring better metrics tools.

Company trends may appear at odds with recommendations the study heard relative to the status quo in DoD acquisition practice. Company practice aims to collect useful data for defined purposes, to integrate measurement with development process innovations, and to exploit metrics productively in an overall business perspective. DoD staff in contrast are recommending limited data collection and measurement programs tailorable to the constraints and goals of each acquisition program.

To pursue a measurement concept comparable to that emerging among companies, DoD would have to have acquisition process objectives that transcend individual acquisition programs. DoD also would have to provide incentives for performance or achievement, similar to what the marketplace offers to company practice.

**What lessons have been learned in applying software metrics?**

The major DoD lesson is that policies and standards are unlikely to be accepted unless acquisition program offices are given latitude to control external reporting and cost impact, as well as an incentive to participate, say in the form of some benefit not available to them by other means. Choices and definitions of metrics should be goal or issue driven, and tailorable to each project and its working environment. Company experience correlates with that view insofar as telling us that metrics prove beneficial when a clear and rewarding goal is identified for their use and they have been carefully designed for well-defined purposes.

**Which metrics have been most beneficial?**

The minimal metrics set being recommended by most DoD programs, e.g., cost or effort, defects, change reports, schedule, development progress, is widely used among companies. Indeed, one company contact exclaimed that a project could not be managed without them. But companies collect more metrics than DoD programs recommend, as it suits their defined purposes for project management and process improvement. Much of the added data is management and financial data related to productivity, competitiveness, and quality management.

**Has the use of metrics influenced the software development process or planning of future efforts?**

This question does not apply to the stated province of DoD metrics efforts, i.e., to assist program management, because program management must be distinguished from software project management. From the company perspective, measurement is having a significant impact and becoming an integral part of the software development process. Investments are being made to expand measurement capabilities.

In DoD, little or nothing is being done to improve metrics tools or establish repositories of data from many projects. Most of the currently recommended metrics do not require significant tools beyond widely available spreadsheet and database packages. Current planning toward a National Software Council may lead to a repository proposal. However, this study indicates that a comprehensive plan is needed for effectively addressing all metrics uses and users across DoD.

In contrast, companies are establishing repositories and using them for process and product improvement. Process maturity criteria are being taken seriously by DoD contractors.

**What human factors issues have been taken into consideration in applying metrics?**

In DoD, the technical complexity of metrics has driven effort to simplify metrics and evolve clear graphical presentations of measurements on a project history time line. Not enough has been accomplished to understand the relevant metrics and to communicate their effective use for acquisition program management or for other DoD responsibilities, e.g., PDSS, that are associated with weapon system acquisition. Training is a recognized problem and a need for action by the Defense Systems Management College was voiced by one contact at least.

None of the DoD recommended metrics will intrude on an individual developer's work unless improperly implemented. Also, with a reasonable level of configuration management and labor accounting support, none should require much time to report. Much of the data is determined and collected periodically at intervals of a week or month. Metrics involving voluminous and tedious counting, e.g., SLOC, are readily automated with tools as has been amply demonstrated.

# APPENDIX A.
# METRICS PROGRAMS OF DOD ORGANIZATIONS

## Table of Contents

## A.1 ARMY ARMAMENT RESEARCH, DEVELOPMENT AND ENGINEERING CENTER (ARDEC)

The Army Armament Research, Development and Engineering Center (ARDEC) at Picatinny Arsenal, New Jersey, is a component of the Army Materiel Command (AMC). ARDEC's mission involves research and engineering on direct fire, close combat systems ranging from bayonets to tank cannons, and indirect fire support systems such as artillery, mortars, mines, and demolitions. ARDEC provides independent evaluation capability to assist Army acquisitions and certify software under its quality assurance, test, and evaluation efforts.

ARDEC's software metrics effort began in 1983 as a research program. In 1990, ARDEC developed the Readiness Growth Model (RGM) for software. This result was an outgrowth of an effort chartered by the Army Vice Chief of Staff and the Undersecretary of the Army to improve the overall quality of software, prevent immature software from entering user system tests or deployment, and to integrate test and evaluation (T&E) into software development processes.

ARDEC also has collaborated with Army Communications - Electronics Command (CECOM) to develop AMC guidance for the implementation of the Army Software Test and Evaluation Panel (STEP) metrics in conjunction with use of RGM.

### A.1.1 Metrics and Reports

The RGM is a cumulative evaluation framework for system indicators or characteristics that have a measurable impact on software maturity. Primary areas such as requirements traceability, stability of requirements, design and code, and fault profiles, are subdivided into quantitatively assessed factors (typically, 26 factors). Point allocations are assigned to each factor and weighted to produce an RGM score that would total 100 points if the software merited a perfect rating for each factor. Assessing the RGM factors periodically or at significant milestones in a system's life cycle yields a visual perception of improving maturity called the RGM curve.

RGM becomes a risk assessment and decision-making tool once specific scores or scoring zones are established for a program to proceed past given milestones such as Preliminary Design Review (PDR), Critical Design Review (CDR) and Formal Qualification Testing (FQT). The trend shown by the RGM curve helps a program manager or acquisition executive to focus on system issues requiring increased attention.

Typical factors used in computing the RGM maturity score are listed in Table A-1. However, RGM is intended as a tailorable methodology to be adjusted by a user to the unique characteristics of a software system and its development methodology.

**Table A-1. Typical RGM Metrics**

| | |
|---|---|
| Requirements allocation/ tracing | Fault profile |
| Requirements stability | Ada implementation |
| Design stability | Code complexity |
| Code stability | FQT test coverage |
| Test coverage | FQT tests passed |
| Tests passed | FQT regression testing |
| PQT test coverage breadth | Technical test coverage |
| PQT tests passed | Technical tests passed |
| | TT regression testing |

Army use of RGM continues as AMC staff effort proceeds toward integration with STEP metrics and their implementation. Draft AMC guidance for the latter, published in 1992, identifies seven baseline metrics and six optional metrics, listed in Table A-2.

**Table A-2. Candidate AMC STEP Metrics**

| Baseline Metrics | Optional Metrics |
|---|---|
| Requirements traceability | Computer resource utilization |
| Fault density prediction | Software engineering environment |
| Fault profiles | Development progress |
| Depth of testing | Manpower |
| Breadth of testing | Cost |
| Code complexity | Reliability |
| Cumulative stability | |

RGM has been applied in about 20 acquisitions of systems such as armored systems, howitzer systems, and smart mines. Specific examples include the M1A2 Abrams battle tank, the Advanced Field Artillery System (AFAS), the Future Armored Resupply Vehicle (FARV), and the Nuclear, Biological, and Chemical Reconnaissance System (NBCRS). DoD program reviews by the Defense Acquisition Board, for Army and other Service programs, also have benefited from use of RGM.

### A.1.2 Levels of Reporting

At present, there is no AMC directive or standard requiring use of RGM or reporting of results to higher command levels.

### A.1.3 Tools and Repositories

Widely available commercial spread sheet packages are suitable for tracking RGM factors and producing the RGM curve in graphic form.

ARDEC maintains a paper repository of RGM metrics for ongoing programs, to serve research purposes. AMC has chartered Army Test and Evaluation Command (TECOM) at Aberdeen Proving Grounds to collect Army STEP metrics data over as many programs as possible. This effort will use a commercial database software package.

### A.1.4 Best Practices and Lessons Learned

Program managers and executives may not readily understand individual software metrics, but they do seem to easily understand the RGM curve. The RGM curve is analogous to the Reliability Growth Curve on which they are trained in systems engineering.

ARDEC staff advocate DoD-wide improvement in software measurement capabilities and are participating in the Joint Logistics Commanders project to develop a measurement handbook to support program managers in use of MIL-STD-498.

A perceived problem with software metrics programs is that contractors appear to need funds earmarked for collecting metrics. Computer-assisted software engineering (CASE) tools should help ease the burden of metrics data collection.

ARDEC staff recommend flexibility in selecting acceptable metrics, e.g., allowing each contractor to define goals and measurements themselves. They also would emphasize product rather than process measurements.

### A.1.5 Future Plans and Directions

ARDEC staff would like to validate RGM with assistance from the Software Engineering Institute (SEI), and also determine if there is any relationship between SEI maturity level scores and the shape of RGM curves.

ARDEC continues to seek experience input for establishing RGM weights, shape of the curve, and other rules of thumb.

ARDEC staff recommend research toward better measures of software reliability. The reliability of a software module in a testing phase may not be an accurate measure of its reliability in the intended operational environment.

## A.2 ARMY COMMUNICATIONS - ELECTRONICS COMMAND (CECOM)

Army Communications - Electronics Command (CECOM), located at Ft. Monmouth, New Jersey, provides engineering and acquisition support services for communications, training and simulation, command and control, intelligence and electronic warfare, fire support, tactical fusion, and avionics systems.

CECOM's software metrics program has evolved since the late 1980s at its Software Engineering Directorate (SED). One early result was a metrics-based quality assurance methodology for the Advanced Field Artillery Tactical Data System (AFATDS). CECOM's efforts have leveraged results from other DoD programs, such as Software Technology for Adaptable and Reliable Systems (STARS) measurement methodology, Air Force Rome Laboratory's software quality framework, and Air Force Systems Command management and quality indicators.

CECOM has produced a guidebook for executive managers describing an issue-driven, data-based metrics methodology called Streamlined Integrated Software Metrics Approach (SISMA). SISMA is a comprehensive approach to acquisition oversight, timely process and product insight, risk management, and process improvement. The SISMA guidebook provides a specific, user-friendly path for applying the recently mandated Army STEP metrics, as well as other metrics that program managers may find useful.

### A.2.1 Metrics and Reports

The SISMA guidebook identifies the STEP metrics and extensively discusses graphical presentations of the metrics and their derivation from original or raw data input. The guidebook includes a series of concise discussions focused on common software management issues.

SISMA supports an organization's development process improvement up through Level 4 of the SEI Capability Maturity Model (CMM). The guidebook includes sample Request for Proposal (RFP) text and Contract Data Requirements List (CDRL) descriptions to implement the SISMA process contractually.

### A.2.2 Levels of Reporting

The SISMA guidebook discusses Army STEP implementation requirements, referencing Part Seven of draft DA Pamphlet 73-1. The technical staff in a program office would develop reports as suggested in the guidebook and provide an analysis for review by the program manager. The analysis would lead to corrective action alternatives for control of the current project, and to lessons learned to be applied to management of future projects.

The CECOM SED has a draft technical operating procedure as guidance for all their supported projects to implement the SISMA and STEP metrics.

No CECOM directives or regulations were identified that concern reporting of software metrics information beyond a program office. However, the draft SED procedure contemplates accumulation of metrics information in a repository. CECOM offers complete support to Army customers in the application of SISMA, and is also developing a standard operating procedure and other guidance documents for measurement programs.

### A.2.3 Tools and Repositories

The use of tools and repositories is advocated in the SISMA guidebook. No specific tools are recommended so as to allow flexibility and to avoid commercial bias. A CECOM effort is mentioned that will develop and integrate a set of tools to support SISMA. The capability to model the development process in which metrics collection and analysis occurs is strongly advocated.

### A.2.4 Best Practices and Lessons Learned

CECOM's SISMA guidebook contains numerous admonitions on best practice and perspectives for software metrics. Here is a partial list of such lessons for users:

a. Metrics are not a silver bullet but serve as basic indicators to provide early insight into a program's trends and status. Program assessments should be based on integration of other information as well.

b. Metrics-based assessments are only as valid as the input data. It is important to identify an efficient data set, viable data sources, and validation techniques.

c. Metrics are only as good as the development process that produces them.

d. Metrics should not be used as a stick to beat the developer or other support organization.

e. The metrics process cannot be conducted exclusively by the developer. A team effort is needed among program office and contractors for greatest benefit.

f. A fixed metrics set is not recommended. Adaptability to existing metrics and reconfigurability to different software process models and mission domains are highly desirable.

g. The metrics process is issue driven and applied throughout the development life cycle.

### A.2.5 Future Plans and Directions

CECOM is now heavily focused on transferring metrics technology to program office customers and providing needed training and support. User surveys and benefit analysis work have been done, and current applications or pilot projects are being tracked. A working group for interchange and planning meets bi-annually, bringing together defense industry metrics experts, CECOM software support staff, and CECOM customers. A specific goal mentioned in the SISMA guidebook is to upgrade the SISMA process to support organizational improvement to CMM Level 5 (Optimizing).

## A.3 ARMY MISSILE COMMAND (MICOM)

Army Missile Command (MICOM) at Redstone Arsenal, Huntsville, Alabama, has a mission for engineering, acquisition, and post-deployment support of Army offensive and defensive missile systems.

MICOM's metrics program began in late FY 89 and has evolved within MICOM's Software Engineering Directorate (SED). Initial direction was given by the Fire Support Program Executive Officer, who requested a minimal set of software management indicators for use in all Fire Support programs. The Software Engineering Directorate considered measurement methods then in practice for Air Force and Army programs, e.g., AFSCP 800-43 and AMC-P 70-13, but sought to improve on them by addressing early development activity and specific means to identify critical problems. The result, a set of Software Management Indicators, is documented in a 1991 user's manual.

No information was available about specific MICOM programs that have adopted and used this metrics set. However, MICOM SED has responsibility for 44 software projects involving 40 different computer languages and has been a strong advocate of software metrics. It has an active metrics working group which recommends policies and procedures for the use of metrics on all projects. The SED continues to perform research, e.g., in metrics data validation and modeling of metrics applications, and has considered the Army's STEP metrics in the light of MICOM experience.

### A.3.1 Metrics and Reports

The Software Management Indicators comprise the following eight metrics:

- Requirements stability
- Software development manpower
- Software development progress
- Computer resource utilization
- Schedule risk
- Trouble report resolution
- Software product delivery
- Supportability

The MICOM user's manual provides instructions on computing the metrics from raw source data (e.g., the specific requirements changes recorded each month) and presenting the data as time history plots and as red-amber-green "stoplight" charts during various time periods during the project life cycle (per DOD-STD-2167A phases). These indicators do attempt to account for object-oriented design and Ada applications

### A.3.2 Levels of Reporting

Management-oriented visual displays are recommended in the user's manual, but no explicit requirements or MICOM regulations were indicated for reporting of Software Management Indicators beyond a program office. There is no current MICOM policy to support or mandate the Software Management Indicators, but MICOM's SED requires all projects to collect metrics.

### A.3.3 Tools and Repositories

MICOM SED has considerable usage experience with metrics tools, with several commercial tools in place for ready use and evaluation experience with many others, including software cost-estimation tools such as COCOMO (COnstructive COst MOdel). MICOM has experience in establishing computer database files for project tracking, evidenced by a relational database description for the Software Engineering Evaluation System (SEES). SEES records the basic project schedule, DOD-STD-2167A deliverable document identification, and basic counts or totals for discrepancies noted, resources expended, and other factors in reviewing project results. Other in-house developed tools include a spreadsheet for collecting and analyzing software architecture metrics as part of research on object-oriented Ada design methodology.

### A.3.4 Best Practices and Lessons Learned

MICOM staff consider it very difficult to find software development tools to collect and support metrics (e.g., as part of configuration management or software maintenance). Most of those evaluated or in use have limited scope of application and none are integrated with development tools to adequately support the range of projects under SED responsibility.

Metrics have played a key role in MICOM SED's software process maturity improvement efforts in cooperation with the Software Engineering Institute (SEI). MICOM SED will soon become the first Army organization to be rated as high as Level 2 on the maturity scale.

Review of Army STEP plans and requirements has raised the following concerns which were identified in working papers:

- Conflict with established metrics programs of individual Army commands.
- Potential cost added by requiring Software Engineering Institute's capability maturity level.
- Complexity metrics, e.g., Halstead metrics, that are not effective with Ada and other development paradigms.
- Metrics that are not mature or well defined, e.g., software reliability.
- Lack of contractual implementation.
- Potential for excessive cost.
- Potential for misinterpretation and misuse of a "national database" of program metrics.
- Little value added in converting Software Management Indicators to STEP format when same indicator is being measured.
- No funds are provided to support STEP implementation.

### A.3.5 Future Plans and Directions

Current research projects include correlating results of complexity and management indicators with project outcomes, and defining better metrics such as coupling and cohesion that are effective for Ada and object-oriented development.

MICOM staff recommend configuring metrics programs in stages correlated to SEI maturity levels and supporting improvements in available tools for metrics data collection.

## A.4 ARMY OPERATIONAL TEST AND EVALUATION (OT&E) AGENCY

The Army Operational Test and Evaluation (OT&E) Agency is responsible for evaluating the acceptability of systems for delivery to its contracting agency, via the mechanism of an operational test. By the 1980s they were discovering that software problems caused most delays in operational test. Between 1986 and 1990 OT&E found that over 90% of the delays experienced in the initial operational tests were due to immature software [Paul a]. Hence, the Software Test and Evaluation Panel (STEP) was formed in September 1989 to address these problems.

One of the principle recommendations of the STEP panel was to "establish a set of core software metrics and a centralized metrics database" [Dubin 1992]. The metrics would help place management into a continuous evaluation mode and thus prevent the failure of software at the last moment, as was occurring in OT&E during the 1980s. The selection criteria for the STEP metric set included the following: each metric must be useful and unambiguous; each must be simple and non-labor intensive to collect and evaluate; each must be capable of consistent interpretation; and each must have intrinsic worth and demonstrate added value to the software development process.

A software metrics program is mandated for all DoD software acquisition programs by DoD Instruction 5000.2, "Defense Acquisition Management Policies and Procedures," February 23, 1991. An Army policy letter dated June 4, 1993, states that the use of the STEP metrics is required for all software-intensive programs. Chapter 17 of DA Pamphlet 73-1, Part 7 defines the STEP metric set and gives extensive information on each metric, including purpose and description of the metric, life cycle application, algorithm and graphical display, data requirements, frequency of reporting, and use and interpretation. Data Item Descriptions (DIDs) have been drafted, formally staffed for comments through the Army Materiel Command, and are awaiting approval.

Although compliance with the STEP metrics is not universal, several program offices are currently collecting STEP metrics data, including the program offices for the Patriot Missile, THAAD (THeatre High-Altitude Defense), and FATDS (Field Artillery Tactical Data Systems). A training program will be funded and provided for all Army program executive officers and program managers beginning in June 1994.

### A.4.1 Metrics and Reports

The STEP metrics set contains 12 software measurements, which are classified as management metrics, requirements metrics, or quality metrics. The management metrics include cost, schedule, computer resource utilization, and software engineering environment. The requirements metrics include requirements traceability and requirements stability. The quality metrics include design stability, fault profiles, breadth of testing, depth of testing, reliability, and complexity. Following is a more detailed description of each of these measurements.

The **cost metric** deals with the dollars spent versus the dollars allocated. It is based on three data items, the budgeted cost of work scheduled, the budgeted cost for work performed, and the actual cost of work performed. These data items are collected for each activity (e.g., requirements analysis, design, code and unit test, computer software configuration item (CSCI) integration and test, problem resolution, software management, quality assurance, configuration management) for each CSCI.

The **schedule metric** indicates major milestones, activities, and key software deliverables. It requires data collection on planned and actual schedules for milestones (e.g., System Design Review (SDR), System Requirements Review (SRR), Software Specification Review (SSR), Preliminary Design Review (PDR), Critical Design Review (CDR), Test Readiness Review (TRR), Formal Qualification Testing (FQT), Technical Test (TT), Functional Configuration Audit (FCA), Physical Configuration Audit (PCA), Operational Test (OT)) and deliverables (e.g., Software Development PLan (SDP), System/Segment Specification (SSS), Software Requirements Specification (SRS), Interface Requirements Specification (IRS), Software Development and Documentation (SDD), Software Test Plan (STP), Software Test Description (STD), Software Test Report (STR), Version Description Document (VDD), Software Product Specification (SPS)).

The **computer resource utilization metric** indicates the current status of resources and the rate that resources are approaching their limits. For each resource (central processing unit (CPU), memory, storage, and input/output (I/O) data) is collected on the capacity of the resource, the target upper bound usage, the projected usage, and the actual usage. For each CSCI (measured in bytes) data is collected on the RAM (random access memory) allocation and actual usage, the mass storage allocation and actual usage, and the capacity of RAM and mass storage.

The **software engineering environment metric** measures the degree of software development process maturity. The data collected includes the maturity level, process strengths and weaknesses, and the date of evaluation.

The **requirements traceability metric** measures the adherence of the software specification, design, and code to their requirements. The metric requires collection of the total number of requirements identified at each level (Operational Requirements Document (ORD), SRS, and IRS) and the number of requirements traceable to the next lower level. (SRS and IRS requirements are traced to CSCI design, computer software component (CSC) design, computer software unit (CSU) design, code, and test cases.)

The **requirements stability metric** measures the degree to which changes in software requirements or changes in the software developers understanding of the requirements, affect the development effort. It requires collection of the following data for each CSCI: number of discrepancies against software requirements for each review (SSR, PDR, and CDR); number of software engineering change proposals (ECPs) against software requirements; for each ECP, number of source lines of code (SLOC) affected, number of modules affected, and number of SRS requirements added, modified, or deleted; and total SLOC for the CSCI.

The **design stability metric** indicates the amount of changes made to the design of the software, as well as the completeness of the design. Design stability is defined as the ratio of modules in the current delivery design that have been modified, added, or deleted from the previous delivery design. Design progress is defined as the ratio of the number of modules in the current delivery design to the number of modules projected for the project.

The **fault profiles metric** provides insight into the quality and maturity of the software. Information collected from software trouble reports includes the number of software faults detected and resolved during testing, and analysis of software faults to determine priority, problem category (when discovered) and status.

The **breadth of testing metric** indicates the degree to which user functions and software requirements have been successfully demonstrated (black box testing). It is defined as the ratio of the number of requirements that have passed their test to the total number of requirements.

The **depth of testing metric** indicates the degree to which CSUs have been tested (white box testing). This metric is composed of coverage calculations:

- Number of paths executed/total number of paths
- Decision points exercised/ total number of decision points

- Number of executable statements exercised/total number of executable statements
- Number of inputs tested /total number of inputs) and success calculations (number of paths successfully executed/total number of paths)
- Number of decision points successfully exercised/total number of decision points
- Number of executable statements successfully exercised/total number of executable statements
- Number of inputs successfully tested/total number of inputs

The **reliability metric** measures the contribution of the software to the overall system mission failure rate in the system's intended environment, assesses the length of system downtime due to software failures, and predicts the number of faults remaining in the software.

The **complexity metric** measures the quality of the evolving design and code in terms of the control structure and size of the software. It includes McCabe's cyclomatic complexity, Halstead's complexity metric, control flow metric (number of times control paths cross), lines of code, and percentage of comment lines.

### A.4.2 Levels of Reporting

There is current discussion on whether OT&E should be involved in tracking test readiness by accessing a centralized metrics database, or whether the acquisition community should control access to their individual project-level metrics databases and hence be the primary users. While OT&E could maintain independence as compared to the end users, current perception is that metrics databases belong under acquisition control.

STEP metrics are to be reported at major acquisition milestone reviews, including Major Automated Information System Review Committee (MAISRC)/Army System Acquisition Review Council (ASARC) reviews and test readiness reviews. The exact reporting requirements, including formats, are still undecided.

### A.4.3 Tools and Repositories

There are several efforts underway to assist in the automation of the data collection and reporting of the STEP metrics. A survey of commercial off the shelf (COTS) metrics data collection tools has been conducted and a summary is included in the Army Pamphlet 73-1, Part 7.

An Army-wide metrics database, designated the Software Metrics Management Information System (SMMIS), will serve as the repository of lessons learned from the use of the STEP metrics. SMMIS output reports will assist Army managers in tracking their projects, evaluating the quality of the resulting software products, and in general, identifying deficiencies or anomalies based upon expected and actual values of the reported metrics. SMMIS is currently available in both a Unix version and a stand-alone DOS version.

STEP has also sponsored the development of a series of software tools, provided as an overall analytical system called the Decision Analysis System (DAS). These tools support senior Army program executive officers (PEOs) in various analytical tasks including statistical analysis, reliability evaluation and prediction, and risk analysis.

### A.4.4 Best Practices and Lessons Learned

When the STEP metrics initiative was proposed, several benefits were foreseen [Paul b]. In the short term, program managers would be able to improve their monitoring of software development; senior Army officials would have measurable criteria for milestone decisions. In the mid term, metrics collection would improve Army software management practices; the central database would allow definition of the most effective metrics and correlations between metrics. In the long term, an established metrics set would encourage development of more concise metrics tools to predict software development progress, measure software product maturity, and define exit criteria for each stage.

The following issues concerning the STEP program reflect opinions gathered from various sources, including the STEP metrics training manual [US Army 1992].

- Many Army managers are nervous about the use of a central repository for a standard set of metrics.
- Many program offices prefer to use their own existing metrics set.
- Some program offices want the freedom to tailor the STEP metrics set to better suit their projects.
- Better definition and clarification of the metrics is needed.
- Guidance for implementation, transition, and support is needed.
- Funding to support data collection and analysis is desired.

While universal compliance is a challenging goal, several Army program offices have adopted and used the STEP metrics set in their projects. A General Accounting Office (GAO) Report, *DOD Slow in Improving Operational Testing of Software-Intensive System*, observes that "the Army has made substantial progress in developing enforceable policy guidance. The Army also implemented . . . 12 Servicewide . . . software metrics."

### A.4.5 Future Plans and Directions

During 1994, the Army has several efforts scheduled that will focus on validating, implementing, and improving the research and development needs of the STEP measurement program. The Army plans to set up a formal validation program for the STEP metrics to determine whether they measure what is intended and if there is a statistically significant relationship between the metrics values and management objectives.

Implementation plans call for installing personal computer based databases in approximately 200 program offices by March 1994 and starting the necessary training by June 1994. By Fall 1994, the program office databases are scheduled to be networked and to be able to deliver data to more centralized databases at the Program Executive Offices. Eventually, it is envisioned that a hierarchy of networked databases will culminate in an Army-wide repository that will be accessible to senior Army officials.

Future research and development efforts support the development of models to help interpret the STEP metrics. The Army is sponsoring the development of a model, based on a neural network, that uses information from previous software projects to predict aspects of a current software project. For example, the neural network may help to identify CSCs that require high development effort or the CSCs that will be error prone [Paul 1992]. In addition, the Army is sponsoring a study whose purpose is to "develop a systematic and comprehensive approach for analyzing and interpreting metrics data." The approach is to develop a multivariate equation based on metrics that are highly correlated. A second objective of the study is to "develop a framework for objectively grouping these similar metrics to obtain a small set of super metrics" [Goel 1993, page 1].

## A.5 NAVAL AIR SYSTEMS COMMAND (NAVAIR) AND NAVAL AIR WARFARE CENTER AIRCRAFT DIVISION (NAWC-AD)

The primary Naval Air Systems Command (NAVAIR) metrics effort is sponsored by the Avionics Systems Engineering Division. AVION Instruction 5235.1 [NAVAIR 1992] applies to all AIR-546 organizational elements supporting NAVAIR weapon system procurements and requires the use of software metrics for all NAVAIR weapons systems. NAVAIR is currently working on policy and procedures for all software efforts, to include software metrics.

The Naval Air Warfare Center (NAWC) is actively participating in these activities. In a separate effort pursuing metrics needs, NAWC Aircraft Division (NAWC-AD) has used a blue ribbon panel to survey its managers, and based on those results and other lessons and experiences, has prioritized a list of metrics into four increments that address management questions. NAWC-AD developed a Software Measurement Guide [Rozum 1992b] with the help of the Software Engineering Institute (SEI). The first draft was released in October 1992 and an update is imminent [Koch 1994]. The revision adds all remaining metrics required for a maturity level 2 organization (e.g., complexity, computer resource utilization, requirements stability, cost, and build release content).

### A.5.1 Metrics and Reports

AVION Instruction 5235.1 requires that software metrics be collected and analyzed to aid the software management process. The instruction defines a specific set of metrics, but NAVAIR did not dictate the use of only that set. Rather it required that set or its equivalent. The goal is to aid the management process and establish a database for projecting development costs for future programs. The instruction's effect has been slow due to the low rate of new Request for Proposals (RFPs) for applicable systems.

The NAVAIR Avionics Systems Engineering Division currently is developing a handbook for implementing software metrics based on the old AVION Instruction 5235.1. It has added a format for collecting and reporting data, information from the SEI, MITRE, and Software Test and Evaluation Plan (STEP) metrics guidebooks, and some new metrics of its own. Some of the metrics include staffing (by labor category, full and part time), effort (staff hours by functional areas and labor category), size (Source Lines of Code or SLOC), product and milestone data (planned and actual dates), work progress (defined work items with exit criteria), complexity, and numbers of units coded per month.

Software metrics are included as a part of the software management handbook and a half-day training course offered on management metrics. The course, although not required, emphasizes systems engineering and includes a portion on software measurement, with role playing, sample metrics, and reports as part of course techniques. It is taped and publicized among software program managers.

### A.5.2 Levels of Reporting

NAVAIR does not require specific data collection mechanisms. There is significant opposition to central reporting of software measures. Metrics and reports are used at different organizational levels. Since metrics are not part of any milestone review, the level to which software measurements are reported varies. The program manager is normally the highest level receiving reports, but it is expected that metrics are useful to the engineering community outside of a program office. NAVAIR is trying to utilize the current Navy management process and the contractor's development process, augmenting them only where necessary and with minimal impact. It hopes to collect data from Contract Data Requirements Lists (CDRLs) and to automate collection and analysis in order to make measurement painless for program manager and contractor. Improvements to CDRL and Statement of Work (SOW) definitions are being made to do so.

Currently, only three or four programs have used the NAWC-AD (Warminster) metrics guide, and there are no new contracts using it or incorporating metrics into their requirements. To get more metric users, NAVAIR is trying to have program managers include metrics on contracts for upgrades, not just new developments. The NAWC-AD guide addresses how to define, how to collect, and how to analyze software metrics. For definitions, it uses the SEI checklist approach, which does not require one standard definition. The most detailed metrics cover support and development progress, and problem report status.

### A.5.3 Tools and Repositories

Although a repository was planned, it did not get included in the first NAVAIR instruction, partly because of a realignment of responsibilities. When NAVAIR mandated reporting of software size, there was concern from laboratories because they were afraid their funding would be affected by the reported metrics. As a result, there are currently only two projects in the NAWC-AD database and comparison is difficult. There are no tools used currently that can be provided to contractors, although some support activities have SLOC counters. The challenge is how to weight and evaluate different metrics once they are available (e.g., how does complexity of units affect the project staffing).

### A.5.4 Best Practices and Lessons Learned

NAVAIR Avionics Systems Engineering Division successfully identified and defined a set of nine metrics for use in monitoring the technical acquisition of software associated with avionics systems. Application of these metrics to contracts has initiated discussions with industry on the acceptability of alternate metrics that provide similar information but may be more readily available. More attention needs to be given to real metrics needs and how easy metrics are to collect. Some issues that need to be addressed are unambiguous definitions, linking data collection to the contracting and software process, making sure the analysis process addresses real issues, and that the feedback and reporting process involves the contractor and project or program manager.

### A.5.5 Future Plans and Directions

NAVAIR recommends that the research community (1) look for measurements in the requirements area (need more than the stability measurement); (2) look for measurements in the systems engineering process;(3) look for different ways to analyze problem reports (source of problem, when identified, necessary corrections), and (4) calibrate cost models with the metrics currently being collected. NAVAIR would like to have a CDRL that would require contractors to provide metrics data on magnetic media in a transferable format such as for a spreadsheet. It also feels that it is very important for government program managers to become better educated in the importance and use of metrics.

## A.6 NAVAL UNDERSEA WARFARE CENTER (NUWC)

The Naval Underwater Systems Command (NUWC) has eight years experience with the use of metrics. By 1991 NUWC was experimenting with a software assessment process based on the use of software metrics [NUSC 1991]. Its goal was to obtain an objective evaluation of software development processes and products for large-scale DoD and commercial programs, based upon quantitative measures [McGarryJ 1993]. NUWC metrics objectives are to manage software processes and products, to improve software system quality, and to refine software cost models. By mid-1992, it extended its metrics objectives to include software reuse with particular emphasis on design, code, commercial off-the-shelf (COTS) and non-development item (NDI) software, and firmware reuse [McGarryJ 1992a] [McGarryJ 1992b].

NUWC places metrics on contract and performs independent data analysis for programs, which it shares with its contractors. It has no training policy and no funding for training.

### A.6.1 Metrics and Reports

NUWC advocates the use of metrics within the context of its quantitative software development assessment process. The software development assessment process has four subprocesses:

- Software issue definition: Software process and product issues are identified and prioritized, so that subsequent measurement and analysis efforts can be focused and cost-effective. Issues define the measurements that will be applied to the program.
- Software attribute measurement: Defined software measures are applied to the software development processes and products.
- Software indicator generation: Reports and graphs are developed based on analysis of the measurement data.
- Software quantitative assessment: Software development issues are clarified and their impact evaluated. Correlations are made between process issues and product characteristics, and recommendations for improvement are generated.

Because the set of measures used for a particular program is based on unique program characteristics and resulting development issues, there is no standard metrics set. However, measurements dealing with software size, development effort, schedule, and errors have been identified as core measures appropriate for all programs. An example set of process and product measures for a typical DoD software development program includes process measures of staffing, effort, facilities, cost performance, productivity, progress (milestones, activities, functions, products), stability, dependencies, and standards conformance; and product measures of size, defects, completeness, stability (requirements, design, code, interfaces, product allocations), consistency, complexity, traceability, reliability, and efficiency.

### A.6.2 Levels of Reporting

The primary intended user of the metrics program is the program manager—the metrics captured and the target audience are at that level.

### A.6.3 Tools and Repositories

NUWC has developed a software metrics database for those systems it is monitoring. It operates on multiple platforms and has been built using commercially available database engines and support applications. It includes a proprietary software measurement schema design that allows the database functions to be adapted to different data sets and measurement methodologies.

The NUWC work is beginning to identify a number of tools and tool categories that support the four phases of its software assessment process [McGarryJ 1993, McGarryJ 1994].

### A.6.4 Best Practices and Lessons Learned

There are several factors that can have an adverse effect on the implementation of a metrics program:

- Metrics assessments that do not support program objectives
- Lack of understanding of metrics by users
- Metrics mistrust

Best practices and lessons learned during software metrics implementation include the following [NUSC 1991, McGarryJ 1992b]:

- Metrics requirements should be issue driven and tailored to specific program characteristics.
- Metrics are part of the overall program management process.

- Access to the software parameter data and interim software products (code) of contractors is mandatory.
- The focus should be on source level (CSU-level) software measurements.
- Plans, actuals, and changes should be measured.
- Metrics are indicators, not absolutes.
- Metrics must be understood and used properly.
- Multiple metrics should be used.
- Quantitative and subjective assessments should be integrated.
- Direct subcontractor insight is important.
- Program-to-program comparisons must be carefully addressed.

NUWC experience indicates that software product quality is related directly to the nature of the development process and that metrics must be tailored to the software development approach (program acquisition, technical and management characteristics), not the reverse. Part of the reason that program offices have not implemented metrics policy may be that metrics collection and use are not correlated to a return on investment. Finally, guidance documents should not be too low level: they must be adaptable to the processes of individual software development organizations. Higher-level descriptions with good tailoring guidelines are more beneficial.

### A.6.5 Future Plans and Directions

In the future the Navy hopes to get metrics comprehensively implemented through Flag-level readiness reviews. Program executive officers will establish their own policies for collecting metrics across their programs.

## A.7 AIR FORCE MATERIEL COMMAND (AFMC)

This summary addresses activity of the Air Force Materiel Command (AFMC) Metrics Task Team. It conveys certain perceptions of command-wide metrics activities, but it is not intended as an overall report on AFMC's use and implementation of software metrics.

The AFMC Embedded Software Management Plan ("balloting draft," dated 1993) drives the work of the AFMC Metrics Task Team. The Plan has evolved since 1992 in response to embedded software management issues identified within AFMC's predecessor components, the Air Force Systems Command and the Air Force Logistics Command. The Plan establishes 24 priorities, and priority number 5 is the establishment of a metrics program for software development and sustainment. AFMC centers (e.g., logistics centers, product centers such as the Electronic Systems Center or the Space and Missile Center, and laboratories such as the Arnold Engineering Development Center) participate on the Metrics Task Team.

The Task Team's objectives, as documented in the Plan, are to define specific software metrics requirements, compile metrics examples, and develop application guidelines to support the following:

- Management visibility and control of software development within a program development.
- Product quality.
- Process improvement.

### A.7.1 Metrics and Reports

The Task Team addresses only embedded software, not management information or automated information systems (i.e., not management information systems (MIS) or automated information systems (AIS)). DOD-STD-2167A and DOD-STD-2168 are considered the principal extant policies that imply but do not dictate use of software metrics.

The Task Team's effort is influenced by significant experience data and perceptions about the state of software measurements. From workshops and survey data, the Task Team determined that little was being done with metrics within AFMC. If metrics are being collected according to the pertinent AF pamphlet 800-48, the resulting data is not being used in acquisition and program management.

Although there may be substantial metrics effort in the technical community at large, much of it is driven by academic concerns, and end-user needs and experience are not being accommodated. The Task Team is therefore seeking to join or influence other metrics efforts (e.g., Defense Information Systems Agency (DISA) and Joint Logistics Commanders (JLC)), to gain leadership in metrics evolution where important to AFMC centers, and to evolve experience-based metrics application guidance for AFMC's embedded software programs. The Team is working to influence revision of the draft Software Development and Documentation standard (MIL-STD-498) so that it will adequately address measurement needs.

### A.7.2 Levels of Reporting

The Task Team perceives that top-down direction, e.g., from DoD, is counter-productive and time is needed at the working level for total quality management ideas to succeed. One manifestation of this view was the Team's influence on a draft policy memorandum that was contemplated by Air Force Headquarters for implementing "core metrics" recommendations based on Software Engineering Institute and Army Software Test and Evaluation (STEP) results. The policy has been completed with recommended changes from the Task Team, and has been released.

### A.7.3 Tools and Repositories

Neither AFMC or the Task Team plans to recommend a command-wide repository to compare organizations and programs.

### A.7.4 Best Practices and Lessons Learned

"Buy-in" at the user level and endorsement at command level are essential for a successful metrics program. Lack of these explains the limited use of software metrics today.

### A.7.5 Future Plans and Directions

A significant Task Team objective is to have a common guide for metrics use across all Air Force development and maintenance programs. Its work also addresses the issue of a common "core" set of software metrics. But the Team has not decided whether a common set is feasible, given the organizational and program differences across the Air Force.

This is the Team's current schedule of work and expected completion dates:

- Researching historical measurement programs (March 1994)

- Collecting metrics on two pilot programs at each AFMC center and sharing resulting data (April 1994)
- Developing draft metrics handbook based on centers' experience (September 1994)
- Coordinating and releasing the prototype metrics handbook (January 1995)
- Using the prototype metrics handbook (January 1995 to January 1996)
- Building training plans for Air Force-wide implementation (October 1995)
- Offering training through the Air Force Institute of Technology (AFIT) and the Defense Systems Management College (DSMC) (January 1996)
- Evaluating and updating the handbook (March to June 1996)
- Releasing revised handbook (October 1996)

## A.8 AIR FORCE ELECTRONICS SYSTEM CENTER (ESC)

The Electronics System Center (ESC) has had a crucial role in development and acquisition of major Air Force systems dating back to the beginning of the Cold War. Consequently, it has a lengthy history of evolving and attempting to implement software measurements. ESC provided the seminal metrics work of the mid-1980s, ESD-TR-85-145 [Mitre 1985], which is still being used.

### A.8.1 Metrics and Reports

For lack of senior management emphasis in recent years, little has been done in new metrics development, technology transfer, and implementation for program management. No Air Force or ESC regulations exist that mandate the use and reporting of metrics. Nonetheless, acquisition programs are "advised" to apply measurement as described in AF pamphlet 800-48 (superceding AF 800-43) or to monitor software development per management indicators described in ESD TR 88-001 [Schultz 1988].

The recommended metrics address software size, personnel, volatility, computer resources, design complexity, schedule, design progress, development progress (computer software component (CSC) and computer software unit (CSU)), testing progress, incremental release content

### A.8.2 Reporting of Metrics

The available guidance recommends reporting metrics to the program executive officer (PEO) level but this is not done in practice. Metrics are only reported to the individual program office level. It is perceived that PEOs are not interested in software as a distinct issue, and instead focus on system-level issues or projects as a whole.

### A.8.3 Tools and Repository

Three ESC program offices are starting to use a metrics package called Decision Vision 1. This tool works through a development environment and collects data unintrusively, covering more metrics than AF pamphlet 800-43.

### A.8.4 Best Practices and Lessons Learned

- Noteworthy Air Force programs that are collecting metrics using ESD TR 88-001 guidance include JSTARS (Joint Surveillance Target Attack Radar System) and CCPDS-R (Command Center Processing and Display System Replacement).

- Air Force has not provided for an adequate level of metrics training.
- Manual data collection and metrics calculations are problems requiring better tools and automation.

### A.8.5 Future Plans and Directions

Information on the future plans of ESC was not available at the time of publication.

## A.9 AIR FORCE ROME LABORATORY

Rome Laboratory is one of the Air Force's principal engineering and technology research organizations. Rome Lab has actively supported software metrics research since the mid-1970s. An early program produced the Software Quality Framework [RL 1985]. This work formed the basis for the *IEEE Standard for a Software Quality Metrics Methodology* [IEEE 1992] and *ISO Software Product Evaluation— Quality Characteristics and Guidelines for Their Use* [ISO 1990]. The Data Analysis Center for Software (DACS) also started out as part of an early Rome Lab software metrics data collection program.

The cornerstone of Rome Lab's current metrics work is the Software Quality Technology Transfer Consortium. This consortium was formed as a joint applied research and development initiative between Rome Lab and several corporations as a mechanism to transfer software engineering measurement technology into the U.S. defense industry. The consortium promotes the application of software quality methods and tools, and the exchange of experiences and data among participating organizations. Data collected by the participants will be used to validate and modify software quality models.

The current software measurement program is one of several programs under the Software Technology Division's Engineering Branch. Other research areas within the Software Technology Division include artificial intelligence, prototype tool development, software requirements, and high performance computing.

### A.9.1 Metrics and Reports

Rome Lab is a research organization rather than a production software developer and, therefore, does not routinely use software metrics to manage its programs. It has produced a procedures guide for implementing the Software Quality Framework that outlines approaches to metrics data collection and includes descriptions of useful reports [RL 1992].

Rome Lab currently sponsors the following areas of software metrics research:

- Reliability metrics for distributed systems.
- Statistical methods for designing measurement experiments and analyzing software metrics data.
- Measures of scrap and rework within the software development process.
- A quality inspection process that adds quality factors to the checklists of defects used in Fagan-style software inspections.

Rome Lab also sponsors annual workshops on software quality measurement, where researchers share results and review the progress of their work.

### A.9.2 Levels of Reporting

Rome Lab's current focus is on software quality measurements that Air Force contractors would report to program offices, and derived metrics that program offices would report to headquarters.

### A.9.3 Tools and Repositories

The Quality Evaluation System (QUES) [RL 1991] is a Rome Laboratory tool that automates aspects of the Software Quality Framework. QUES accepts collected data, computes quality metrics, and displays the results. Automated data collection tools include static analyzers for Ada and Fortran.

Software Quality Technology Transfer Consortium members are currently building a database of software reliability and maintainability factors gathered from systems they have developed.

### A.9.4 Best Practices and Lessons Learned

It is not clear how some quality models can be applied to improve Air Force software development. Most models were developed and tested on incomplete, ideal, or old metrics data sets. These models need to be validated and updated. Procedures for using them also need to be developed.

Many software development organizations are not mature enough to use models and resulting process adjustment recommendations.

### A.9.5 Future Plans and Directions

The current direction of Rome Lab's software metrics research is to continue to develop and validate procedures and tools to support models of software quality, refine existing models, extend software reliability models, collect more operational data, and develop novel measures based on code.

A prototype of the Air Force's National Software Data and Information Repository is currently in the planning stage.

## A.10 DEFENSE INFORMATION SYSTEMS AGENCY (DISA)

Current Defense Information Systems Agency (DISA) activities in software metrics are being conducted in the Center for Standards (CFS) and the Center for Information Management (CIM).

DISA is the Executive Agent for the Department of Defense (DoD) Information Technology (IT) Standards Program, and as such, DISA is responsible for integrating, coordinating, and managing all DoD IT standards activities. Within DISA, the Joint Interoperability and Engineering Organization (JIEO) CFS is assigned Executive Agency responsibilities for the IT Standards Program [DISA 1993a], including software metrics standards.

In conjunction with the Joint Logistics Commanders, DISA JIEO/CFS is drafting a DoD handbook (MIL-HDBK-SWM) to support the new SDD (MIL-STD-498, Software Development and Documentation) which will require the use of metrics. Enforcement of the metrics handbook is currently not planned. The handbook will provide acquisition managers with a reference guideline and software engineering personnel with rationale for metrics selection when establishing measurement efforts. Intended users include program offices, in-house development activities, and contractors conducting software-related activities. The handbook is for use by all DoD components, and would be applicable throughout the software life cycle. It will include both process (management) and product (quality) measurements. The handbook's sponsor, the Joint Logistics Commanders, has stated that the first priority is to support the acquisition of weapon systems, so the handbook's impact on automated information systems and general information management applications remains to be seen.

Activities in CIM address needs of the Central Design Agencies (CDAs), program managers, and also the DISA Software Reuse Program (SRP). DISA is working to understand the current state of metrics practice within DoD, DISA contractors, and industry at large. Much of the internal DISA motivation comes from a recognition that it needs to improve the processes by which it develops and maintains software intensive systems, but to do so it must be able to measure where it needs to improve and how well it is improving. There is a belief that the DoD IT community lags behind the weapon system community in understanding the need for metrics and in desire to overcome the shortfall between measurement capabilities and need.

DISA has established a Software Metrics Standards Working Group under the authority of DoD 4120.3, Defense Standardization and Specification Program, Policies, Procedures, and Instructions, to promote information flow beyond the routine exchange of correspondence and expedite the Software Metrics Initiative. It includes members from DISA CIM, the Army, Air Force, Navy, Defense Mapping Agency, Defense Intelligence Agency, the Software Productivity Consortium, and a support contractor, Logicon.

DISA is working with CDAs to pilot the use of the Software Engineering Institute's (SEI) core metrics (size, effort, schedule, quality) at eight volunteer CDA locations. The current DISA role is seen as infusing good practices into the CDAs, rather than oversight. The metrics data is intended to be helpful to the CDAs in managing their projects better and improving their software practices. Each CDA typically has three to four projects in the metrics program. DISA had a four-day workshop for metrics champions from each CDA site, giving training on the SEI core metrics. Site champions meet with DISA CIM monthly. The pilot program is expected to run for one year, with the data collection having begun in July 1993.

DISA also has a reuse metrics program operated by the DISA/JIEO/CIM Center for Reuse Operations as part of the Software Reuse Program. DISA's Software Reuse Metrics Plan [Chubin 1993] describes a three-phase approach. The first phase focuses primarily on a repository and the nature and scope of interaction between a user and the repository. The second phase, lasting through FY1994, focuses on identifying the costs and benefits of the SRP through pilot projects. The third phase will focus on a deeper analyses of the reuse process, including the evolving areas of domain analysis and design. The plan defines collection forms and reports that should be produced from the data collected. The goal of the program is to improve the Reuse program through measurements that can (1) enable understanding of reuse processes, (2) evaluate the quality of products of software development and maintenance, (3) control the activities of the reuse process, (4) forecast future reuse needs based on current needs.

### A.10.1 Levels of Reporting

DISA recognizes three levels of reporting: oversight, program management and technical. The initial focus is to satisfy the needs of program managers [DISA 1993b]. A high priority is to keep the collected metrics at the CDA level. The Reuse Plan is consistent with this view.

### A.10.2 Tools and Repositories

Except for the SRP, DISA is not currently pursuing metrics repository activities or tools. The intent is to work with others already active in these areas, such as the Air Force, and to focus on the understanding and guidance needed for starting small, identifying the metrics user audience, and knowing how the information will be used prior to its collection.

### A.10.3 Best Practices and Lessons Learned

One problem identified as part of early attempts to introduce metrics into the CDAs is a lack of infrastructure for collecting metrics, e.g., CDAs often have no method for collecting information on hours worked on software—they use manual data collection processes, they do not track defects, and data is collected differently by each center. DISA feels that the research and development community needs to work on additional tools to automate collection, analyses, and reporting of measurements.

### A.10.4 Future Plans and Directions

DISA plans a sequence of working group meetings to establish sample goals and questions to be answered by metrics collection, identification of 10 to 15 priority metrics, and suggestions for metrics presentation. It plans to have a complete draft metrics handbook available in the third quarter of 1994. DISA feels that the DoD needs to establish a policy to encourage agencies to collect and use metrics, and it thinks that the Defense Systems Management College (DSMC) needs to add metrics to its teaching subjects.

## A.11 JOINT LOGISTICS COMMANDERS (JLC)

Joint Logistics Commanders (JLC) are responsible for logistical operations within the Army, Navy, and Air Force. They have established a Computer Resource Management (CRM) group for developing solutions to problems that the JLC face in the area of computer and software intensive systems. The JLC have had a continuing interest in metrics. The JLC-CRM produced a draft metrics handbook in 1992 to accompany DOD-STD-2167A. They have conducted two workshops, Orlando II and San Antonio I, where software measurement was the subject of several panels. The workshop reports documented the consensus that a measurement approach similar to that of the Air Force indicators would be a good foundation for metrics in acquisition, development, and support [JLC1993a]. The individual Services currently have metrics initiatives which are based on the JLC workshop information as modified by their own needs.

The CRM has three significant metrics activities currently. The JLC-CRM and the Defense Information Systems Agency (DISA) are jointly involved in developing a handbook to support measurement in conjunction with MIL-STD-498. This handbook, known as MIL-HDBK-SWM, also is mentioned in the DISA report in this Appendix. Two other efforts not detailed here are reliability/maintainability metrics validation, and work on using metrics to define system readiness for test.

### A.11.1 Metrics and Reports

To assist in the MIL-HDBK-SWM handbook effort, the CRM has established a working group composed of members from Army, Navy, Air Force, Marines, DISA, and the Defense Logistics Agency (DLA). JLC prefers the handbook to emphasize weapon system software acquisition. Another JLC objective is for the handbook to contain complete instructions for computer acquisition personnel across all of the Services and logistics agencies.

The handbook is evolving with four distinct areas of content. One area provides an overview of measurement from a program manager's viewpoint, identifying typical issues, constraints, and questions. The second area identifies workable metrics in three levels: a basic set, an enhanced set, and an advanced set. A third area addresses step by step detailed instructions for implementation. The fourth content area provides statement of work and data item description guidance for implementing the handbook in solicitations.

Currently, Virginia Polytechnical and State University, Blacksburg, Virginia, is validating metrics on reliability and maintainability by following several software development contracts. It expects to validate design indicators through statistical comparison [JLC1991a, JLC1993a]. The work started under the Navy, but was transitioned to the JLC in order to obtain Tri-Service and industry perspectives. The validation project is due to complete in 1994, though not tied to release of the Software Development and Documentation (SDD) metrics handbook.

### A.11.2 Levels of Reporting

Currently, there is no JLC policy prescribing use of metrics, and enforcement of standard metrics is not a JLC-CRM interest. JLC has no definite plan to define organizational levels that will collect metrics, although the scope of the working group's charter calls for metrics at various levels (e.g., oversight, buy-off).

### A.11.3 Tools and Repository

The JLC is not considering a central repository for collecting metrics under the new handbook. They believe that data would not be comparable because projects would be dissimilar. The JLC-CRM has another working group addressing tools, and Virginia Polytechnical has also developed a language analyzer for Ada, a DOD-STD-2167A document analyzer, and a report generator to support the subset of metrics it is validating. It is too early to tell if tools will be included in the SDD metrics handbook.

### A.11.4 Best Practices and Lessons Learned

Spreadsheets were provided for collecting and reporting measurement data for the DOD-STD-2167A handbook but they were not used. There was no senior management support, commitment, or enforcement. Some commands tasked contractors to collect the data, while others used the DLA offices. It required very time-consuming data collection efforts, and payback was minimal as contractors and program managers were not using the data. While the metrics were not validated, it was determined that Halstead's complexity measures were not effective.

### A.11.5 Future Plans and Directions

The JLC/DISA working group plans to finalize the Charter of the Working Group at its January 1994 meeting. Research and development is not in the scope of the CRM charter.

Training needs have not been addressed yet, but there is a JLC-CRM working group on training. The current plan is to use the same training support as provided to SDD.

Working group participants do recommend establishing a DoD-wide metrics policy to encourage agencies to collect and use metrics. Some also believe that the Defense Systems Management College (DSMC) should add software metrics to its courses. Working group members also recommend that DoD identify or provide additional tools to automate metrics collection, analysis, and reporting.

## A.12 SOFTWARE ENGINEERING INSTITUTE (SEI)

The Software Engineering Institute (SEI) is a federally funded research and development center (FFRDC) affiliated with Carnegie Mellon University. Both are located in Pittsburgh, Pennsylvania. The SEI's mission is to "advance the state of the software engineering practice to improve the quality of mission-critical computer systems." One of its major program areas is the "software engineering process" that includes the "software process measurement project." The objective of this project is to promote and improve the use of measurement in managing, acquiring, and supporting software systems [SEI 1993].

As part of SEI's efforts to promote and improve the use of measurement, SEI has published a number of documents on software metrics. These include a curriculum module [Mills 1988], an explanation of how program managers can use metrics [Rozum 1992a], a comparison of software metrics and SEI's Capability Maturity Model (CMM) [McWhinney 1992], and even a concept study for a national software engineering database [Van Verth 1992].

In 1992, SEI's software process measurement project was asked by the Department of Defense (DoD) to develop materials and guidelines for a core set of measures to be used in DoD software acquisition, development, and support programs. The objective was to produce measures that would serve as a basis for collecting well-understood and consistent data throughout the DoD. This core set was of four metrics developed and provided as a capstone document [Carleton 1992], with three documents providing additional detail.

### A.12.1 Metrics and Reports

The SEI recommends that at least four basic measures should be used by the DoD for acquiring, developing, and maintaining software systems. These measures are shown in Table A-3.

Each of these metrics can be defined in slightly different ways. To resolve this problem, SEI provides a set of checklists and forms to define each measure more precisely. SEI also provides recommended basic definitions for each term using these forms. The checklist method of defining key terms is very useful because it clearly shows definition alternatives.

SEI's definition of source lines of code (SLOC) is possibly the most contentious because there are so many different methods for measuring software size. SEI recommends measuring physical lines of code, excluding blank and comment-only lines. Advantages of this approach include the ease of application, ease of definition, reduced dependency on a particular programming language, and the fact that many cost model databases use this measure. A significant disadvantage is that this definition is much more subject to variations in programming style than logical lines of code. SEI did not include function points; one reason is that although they appear to be effective in business environments, they have not enjoyed widespread success in other environments (such as embedded systems or computation-intensive systems).

**Table A-3. SEI's Recommended Core Metrics[a]**

| Unit of Measure | Software Characteristics Addressed | Described In |
|---|---|---|
| Counts of physical source lines of code (SLOC) | Size, progress, reuse | [Park 92] |
| Counts of staff-hours expended | Effort, cost, resource allocations | [Goethert 92] |
| Calendar dates (tied to milestones, reviews and audits, and deliverable products) | Schedule | |
| Counts of software problems and defects | Quality, readiness for delivery, improvement trends | [Florac 92] |

a. Source: [Carleton 1993b]

SEI's definition of effort is the total hours of time expended by a member of the staff. It includes direct labor and excludes indirect labor. It includes regular time and overtime, whether or not it is compensated, by all workers (including part-time and subcontracted). All product-related effort is included, including system requirements analysis through deployment, management, software quality assurance, configuration management, training, and support. Hours were selected instead of weeks or months, since there is no "standard" work-week or work-month. Overtime is to be separately reported. Including unpaid overtime is interesting and in many ways beneficial, since unpaid overtime often hides productivity improvements. However, including this value makes this measurement more difficult to obtain, since this information is rarely collected. It also makes the metric's use for cost estimation more problematic since this cost is not directly borne by the contractor or government. Significant amounts of unpaid overtime may be unreported in spite of this requirement, causing mismeasurement.

SEI's definition of schedule metrics are the actual and planned dates for various milestones, reviews, and audits (such as Software Specification Review and code complete) for a selected set of exit completion criteria (such as internal review complete, formal review with customer complete, all action items closed, and quality assurance sign-off). SEI's key contribution here is the definition of multiple dates for some milestones (for example, separating review complete from action items complete).

SEI's definition of counts of software problems and defects includes a definition of a software problem. A software problem is defined as a defect in the requirements, design, code, or operational document. Excluded are test case defects, hardware or operating system problems, and new (or enhanced) requirements.

SEI provides sample reporting forms for all of these metrics.

### A.12.2 Levels of Reporting

The SEI intends for these metrics to be used by the DoD, implying that the SEI expects these metrics to be reported by software development contractors to their respective DoD sponsors.

### A.12.3 Tools and Repositories

The SEI does not provide any specific tools, though it does recommend that the DoD develop and distribute such tools. SEI does note that their recommended size measure (physical lines of code) is easier to develop tools for than size measures based on logical lines of code. The SEI has developed a concept study for a national software engineering database, but has not committed itself to plan or implement such a database [Van Verth 92].

### A.12.4 Best Practices and Lessons Learned

SEI recommends that the DoD consider the following actions for implementing a DoD metrics program [Carleton 1993a]:

- Develop instructional and training material for the metrics set.
- Offer an initial pilot test period before a metrics set is mandated.
- Designate and staff an organization to respond to user questions and provide advice and assistance to organizations implementing these practices.
- Prepare and distribute software to assist in automatically collecting the required measures.

- Provide for a review and revision cycle for updating the framework metrics documents, and plan for republishing and redistributing these update.
- Consider extending the set of measures. In particular, defining counts of software units (or functions), rules for logical source statements for principle DoD languages, and dollar cost measures (as an analog to staff-hours).

Rifkin and Cox [Rifkin 1991] performed site surveys of organizations with reputations for excellent measurement practices, and their summary provides a number of useful lessons learned. They found that in all successful organizations the following:

- Errors had been decriminalized. Defects were made public—no one was surprised by them, and everyone was working to eliminate them.
- Measurement was part of a larger effort for software improvement. Measurement was not added on, appended, or made to stand alone.

The following patterns were true for most successful organizations and can be considered suggestions:

- Measurement Content: Relatively few (1 to 20) measures were collected which were useful and easy to obtain. Starting small was viewed as important, and many only collected one metric (number of defects). Measures used were obviously practical, incurred low collection cost and effort, could be presented simply, and were rigorously defined. Most provided automated tool support to minimize the impact of measurement on software developers. Common metrics included number of defects, effort (labor hours) and size (lines of code).
- People: Managers must be motivated, measurement goals must be focused and articulated, all stakeholders must be involved, all affected persons must be educated and trained, and trust must be earned. In particular, no harm should come to the bearer of bad news. One common goal was to ascertain where to apply additional resources to improve the software product or process.
- Program: It is best to take an evolutionary approach, plan to "throw one away," get the right information to the right people, and strive for an initial success.
- Implementation: Measurement should "add value," developers should be empowered to use measurement information, a "whole process" view should be taken, and it must be understood that adoption takes time.

### A.12.5 Future Plans and Directions

SEI is considering a number of future directions. SEI is considering defining measurements for reliability, complexity, and process understanding (such as process control and cause identification and defect prevention). Methods for improving software cost estimation are also being considered. Planned future technical reports subjects include instructions for establishing a measurement process and a description of cost estimating practices.

SEI plans to assist the metrics programs of specific organizations. SEI is also considering participation in the software metric standards efforts of the Federal government, the Institute of Electrical and Electronics Engineers (IEEE), Inc., and the International Organization for Standardization (ISO).

## A.13 NATIONAL AERONAUTICS AND SPACE ADMINISTRATION (NASA)

The National Aeronautics and Space Administration (NASA) consists of nine autonomous centers. The Goddard Space Flight Center in Greenbelt, Maryland, is the largest and most diverse. Goddard's primary mission is scientific investigation, largely supported through unmanned, earth-orbiting space craft missions.

NASA's Software Engineering Laboratory (SEL) was started in 1976 as a cooperative effort between NASA-Goddard, the University of Maryland (Computer Science Department), College Park, Maryland, and Computer Sciences Corporation. The SEL is an integral part of Goddard's Flight Dynamics Division, which develops ground support systems (e.g., mission planning, altitude and orbit determination) on an annual software budget in the $20 to $30 million range. Most of the systems are non-embedded and mid-sized, in the 100 KSLOC range. SEL's objective is to learn from every software project in the Flight Dynamics Division. To date, 115 software projects have been tracked.

There are some software standards at NASA-Goddard but they do not include the use of metrics, except in the SEL. Everyone within the Flight Dynamics Division and SEL follows additional requirements set forth in a series of standards, including the overall policy document called *Manager's Handbook for Software Development.* This handbook is based on data collected during SEL experiments. Individual project data is summarized to produce a model for the software development process (e.g., errors per KSLOC is four at code and unit test, two at integration test, one at functional test). If a project comes in with a different profile than the model suggests, it is examined more closely. In general, metrics goals are achieved by strong personal leadership and management, rather than by policy.

In 1992 a NASA-wide effort was started to encourage local organizations to collect measurement data to benefit their software development programs. Five NASA centers currently participate, each having a development-oriented person as a process improvement champion. Current efforts include the development of a NASA-wide Software Measurement Guidebook to provide guidance in starting a measurement program. The guidebook recommends a core set of measurements but leaves the detailed definition of each measure to the local organizations. This document was completed in April 1994.

### A.13.1 Metrics and Reports

From the perspective of the SEL, there are three reasons to measure: to attain an understanding of the engineering principles of software (e.g. the relationship between size and cost, comparison of flowcharts vs. program design language (PDL), evaluation of structured programming methodology), to manage ongoing projects, and to guide process improvement. More particular objectives of a measurement program will depend on the particular center's mission. For example, at Goddard the objective may be to reduce software costs while at the Johnson Space Center, Houston, Texas, the objective may be to increase software reliability.

Data collected during an SEL experiment is based on the particular goals of the experiment, in accordance with the Goal-Question-Metric paradigm developed at the University of Maryland. Development teams are informed of the study's goals, of the derived questions, and of the types of data to be collected during development. They provide much of the required data through the use of data collection forms. Their prime goal, however is to complete the project on time and within budget. Non-development staff personnel are responsible for metrics data entry, quality assurance, and analysis.

Certain data have been identified as critical and are recommended for collection on all projects [Valett 1989]. These include schedule dates for phase completion, size and effort estimates, methodology, environment, and tools used on the project. Also included are actual effort data, computer utilization data, and counts of software changes and errors and the relative effort to implement them.

### A.13.2 Levels of Reporting

Every project within the SEL is responsible for producing a report that includes statistics, context, and lessons learned on the project. The report is started during acceptance test and usually takes about three staff-months of effort to complete. Data and lessons learned from individual projects have been incorporated in the *Manager's Handbook for Software Development* and made available to all personnel within SEL and the Flight Dynamics Division of NASA-Goddard.

### A.13.3 Tools and Repositories

Most data is collected manually, which does not appear to pose any problems. The Personnel Resource form, which is filled out weekly by each project member, contains information on the number of hours spend on each activity (predesign, design, read/review of design, code, read/review of code, unit test, debug, integration test, acceptance test, and other). The Change Report form, which is completed each time a change is made to code that has already passed unit test, contains information about the change that was made, including error information if applicable. The Components Origination form, which is filled out after a component is unit tested, gives detailed data about the component such as its purpose, whether it is new vs. modified vs. old, and the difficulty of developing it. An Estimation form, which is filled out by project managers after each significant phase or milestone, gives projections of phase dates, system size, and effort to build the system. Automated tools are available to collect information on computer utilization, component change history and growth, and product characteristics such as lines of code (LOC) per module and the number of statements of each statement type [Valett 1989].

Oracle is used for the SEL data base and for the generation of monthly reports. Proprietary data is not a problem within SEL because of the support contractor relationship. Since NASA centers have different measurement objectives, NASA does not have a central database and does not require common metrics.

### A.13.4 Best Practices and Lessons Learned

Best practices and lessons learned from over 17 years of experience within the SEL include the following [McGarryF 1993].

- Data collection must not be the dominant element of process improvement: application of measures is the goal. Organizational focus should be on establishing measurement goals and planning for the usage, benefits, and applications of the collected measures. Too often, organizations focus on defining a list of measures and the data collection forms that will be used to collect data. At least three times as much effort should be spent on analyzing and using collected data as is spent in data collection.

- Measurement must be focused and well defined to get what you want. SEL's measurement program today collects 50% less data than it did 10 years ago. The original data collection effort was too difficult and SEL was uncertain what to do with the data.

- A measurement program must focus on local improvement, not on comparison with external organizations. Different organizations have different emphases (e.g., cost vs. quality vs. security), different product domains (e.g., embedded vs. non-embedded), and different goals. Because of such differences, it is difficult if not impossible to derive meaningful results from combining data across dissimilar domains. There are two corollaries to this emphasis on local usage:

  - Define standard local terminology instead of attempting to generate widely accepted standard definitions.

  - Use measurement data locally: do not expend effort formulating broad, national databases.

- Measurement data will be inexact, incomplete, and fallible. This fact must be accepted and data analyses designed accordingly.

- The process of measurement cannot be significantly automated. Tools can automate a limited set of routine processes for counting measures such as code size, code growth, errors, and computer usage. However, humans are necessary to provide insights into the reasons for errors, changes, and problems.

### A.13.5 Future Plans and Directions

The NASA-wide measurement effort is focusing on giving guidance to local organizations that are interested in metrics. Trial tests are being performed in five centers with "champion" developers. Once these measurements have been proven successful, others will follow. NASA plans to publish a *Software Measurement Guidebook* in April 1994, which includes guidance for starting a measurement program. The core measures that will be included include LOC, dates, and schedules. It plans to collect these measures for each Center.

SEL feels that what is needed is less basic research and more infusion of measurements into the practitioners world (e.g., guidance, how to utilize, how to determine benefits). A stronger awareness should be established between the various programs (e.g., SEI, NASA, the Advanced Research Projects Agency (ARPA)), without expecting collaboration. They also recommend identifying the impacts and defining better relationships between various software engineering models (e.g., object-oriented design vs. functional decomposition, errors vs. different testing strategies, changes in specifications vs. changes in designs).

# APPENDIX B.
# METRICS PROGRAMS OF INDUSTRY ORGANIZATIONS

Table of Contents

## B.1 COMPANY A

The information in this report pertains to an electronic systems group within a large company that is preeminent in both commercial and defense markets. The group, which contains multiple divisions employing over 450 software engineers, specializes in developing mission critical software systems for the Department of Defense (DoD).

The group's metrics program began in the 1980s and was influenced by several outside metrics initiatives as it evolved. The management metrics defined in AFSCP 800-43 [AFSC 1986] was an early influence. Later influences include the Goal-question-metric Paradigm of Basili [1984], the Mitre ESD report TR-88-001 [Schultz 1988], and the Software Engineering Institute (SEI) process metrics questions in the SEI report SEI-87-TR-23 [Humphrey 1987].

In 1988, an SEI-assisted assessment found weaknesses in both metrics and process. As a result, a Software Engineering Process Organization (SEPO) was formed, which spawned off 15 working groups, including one on metrics. Since mid-1988, the working group has been meeting weekly.

A Software Management Manual Policy requires the use of software metrics for all mission-critical software projects of greater than five labor years. A software metrics handbook completed in 1993 describes the metrics set and the metrics-related process and procedures for implementing the policy.

### B.1.1 Metrics and Reports

The metrics set consists of 27 measures that support the attainment of 5 goals established by the SEPO:

- To obtain an SEI maturity level 4 rating
- To increase software development productivity by 30%.
- To achieve reliable forecasting of the electronic systems group's ability to meet project cost and schedule objectives.
- To increase software product quality by 30%.
- To improve customer satisfaction by increasing on-time deliveries by 30%.

The first 19 measures in the metrics set derive directly from the 1987 SEI questionnaire and relate directly to SEPO's first goal of obtaining a level 4 SEI rating. Metrics 20 and

20a relate to its second goal; metrics 22 and 23 to the third goal; metrics 20b, 20c, 20d, and 24 to the fourth goal; and metric 20e to the fifth goal.

Metrics usage is optional for projects with less than five labor years of software effort. Mission-critical software projects between 5 and 30 labor-years in size must use 8 of the 27 metrics (numbers 8, 9, 10, 11, 20, 22, 24, and 25 in the metrics list that follows). Mission-critical software projects larger than 30 labor years in size are responsible for the entire set of metrics, although 4 of the metrics are tailorable (numbers 5, 6, 13, and 14) and may be treated as optional.

Metrics are to be generated monthly and maintained for the duration of a program. The same metrics may also be generated quarterly and annually to help identify trends. Quarterly metrics are calculated as a weighted average of a consecutive three-month interval and annual metrics as a weighted average of quarterly data.

During the proposal phase of a program, software engineering managers are encouraged to get customer approval for using the group's defined metrics set. If such approval cannot be obtained, software project managers are in the position of having to obtain software steering committee release from the required internal metrics set.

**The 27 Measures**

1. **Profiles of actual versus planned staffing.** The suggested reporting format is to graph both planned and actual staffing data on one chart, where the *y* axis represents the number of staff and the *x* axis represents time since the start of the contract, in one month intervals. Optional, additional information could include actual versus planned experienced staff and unplanned staff losses.

2. **Profiles of software size for each CSCI.** The suggested format is planned versus actual size, in Delivered Source Instructions (DSI) for each CSCI (computer software configuration item), reported in monthly intervals. Data can optionally be refined by breaking total DSI into new code, modified code, and reused code.

3. **Statistics on software design errors.** Design errors are Software Errors or Defects (SEDs) that are traced to the design task. (A software error is a problem that is detected before submission to formal test, whereas a defect is a problem detected after submission to formal test.) Cumulative design errors are to be plotted monthly and compared against historical benchmarks and project events such as requirements changes.

4. **Statistics on code and test errors.** Cumulative code and test errors are to be plotted monthly and compared against historical benchmarks and project events.

5. **Project and compare design errors to actuals (optional).** Estimate the number of expected design errors to be detected each month of the project, based on historical data and current project environment. Plot expected and actual data in monthly intervals.

6. **Project and compare code and test errors to actuals (optional).** Estimate the number of expected code and test errors to be detected each month of the project, based on historical data and current project environment. Plot expected and actual data in monthly intervals.

7. **Profiles of actual versus planned CSUs designed.** Plot the number of actual CSUs (computer software units) designed versus the number of CSUs planned to be designed, in monthly intervals.

8. **Profiles of actual versus planned CSUs tested.** Plot the number of actual CSUs completing CSU test versus the number of CSUs planned to be completely tested, in monthly intervals.

9. **Profiles of actual versus planned CSUs integrated.** Plot the number of actual versus planned CSUs integrated at the CSCI level, in monthly intervals.

10. **Track target computer memory utilization.** Graph the actual versus planned target computer memory utilization in monthly intervals. (Computer memory utilization is defined as the percent of memory that is unavailable for additional software—worst case.)

11. **Track target computer throughput utilization.** Graph the actual versus planned target computer throughput utilization in monthly intervals. (Computer throughput utilization is defined as the percent of active CPU time—worst case.)

12. **Track target computer input/output (I/O) channel utilization.** Graph the actual versus planned target computer I/O channel utilization in monthly intervals. (Computer I/O channel utilization is defined as the percent of active I/O channel time—worst case.)

13. **Measure and record design and code review coverage (optional).** Track the percentage of CSUs completing the review process each month. This data can be compared to the percent of elapsed allocated design and code time. (For the review of a CSU to be complete, all associated phase reviews, including Software Requirements Specification (SRS)/Interface Requirements Specification (IRS), CSU design, interfaces, timing constraints, test cases, Software Development Files (SDFs), and test results, must be completed, and all resulting action items must be closed.)

14. **Measure and record test coverage for each phase of functional testing (optional).** Track the test coverage for each phase of testing each month. Compare the percent coverage to the elapsed allocated test time per phase of functional testing. (Test coverage is defined as the percent of requirements in the SRS and IRS verified by test.)

15. **Track to closure action items resulting from design reviews.** Track the number of open and closed action items resulting from formal design reviews (SSR (Software Specification Review), PDR (Preliminary Design Review), and CDR (Critical Design review)) each month.

16. **Track to closure software trouble reports resulting from testing.** Track the number of open and closed software trouble reports resulting from testing each month.

17. **Track to closure action items resulting from code reviews.** Track the number of open and closed action items resulting from code reviews each month.

18. **Track test progress by deliverable software component and compare to plan.** Track monthly the actual versus planned percent of CSC requirements (from SRS/IRS) successfully verified by test for each deliverable CSC.

19. **Profile of software build/release content versus time.** Track monthly the actual versus planned percent of SRS/IRS requirements implemented in the deliverable CSCI build.

20. **Pulse point charts (produced only for software formally tested).**

    a. Productivity (DSI/person month) for both developed and reused code each quarter

(b) Design quality (no. of errors/thousand DSIs) for both developed and reused code each quarter.

(c) Defect density (no. of defects/thousand DSIs) for both developed and reused code each month. (No. of defects is calculated one year after formal testing. Defects are tracked for a period of one year after delivery.).

(d) The number of on-time and late software deliveries each month and the number of on-time and late documentation deliveries each month.

21. **Labor months expended per CSC over time and DSIs completed for each CSC.** Graph the planned versus actual values for both the total monthly labor months expended per CSC and the total KDSIs (thousands of DSI) completed per month per CSC.

22. **Track project cost and schedule data both planned and actual.** Track planned versus actual data for project cost and schedule each month. Schedule information can be represented as percent of completed milestones versus percent of schedule time elapsed.

23. **Profile of the number of items changed in the SRS and IRS over time.** Track the cumulative number of added, deleted, and modified requirements in the SRS or IRS for each deliverable CSCI each month.

24. **Track the total number of open and closed SPRs.** Track the cumulative number of both open and closed software problem reports (SPRs) each month. SPRs should be categorized by the phase is which they were introduced and the phase in which they were detected.

25. **Track CSCI progress with respect to the SRS and SDD.** Track the actual versus planned number of requirements completed in the SRS, SDD (Software Design Document), IRS, and IDD (Interface Design Document) each month.

26. **Track the software development tools.** Track the actual versus required date of availability of each software development tool.

27. **Track the progress of the documentation.** Track the actual versus planned number of pages of deliverable documentation completed, in monthly increments.

### B.1.2 Levels of Reporting

Most of the metrics data is used at the project level in day-to-day management of the project. The most useful of these metrics are those involving timing, size, manpower loading, cost, schedule, defects and errors, and trend analysis within a program.

Certain measures are collected across all divisions, analyzed, and reported quarterly to upper management. These measures include productivity and quality measures for the group as a whole (see metric 20, pulse point charts). Upper management is enthusiastic about this information, which it is currently getting from its software, manufacturing, engineering, and procurement organizations. However, some software project managers do not feel it is worth the level of effort it takes to produce the reports. From their perspective, the reports are often not insightful because they combine across many projects, not helping a specific project. This is especially true for productivity reports across product domains. Productivity comparisons are much more valid within a domain.

### B.1.3 Tools and Repositories

Standard forms exist to aid in the collection of the standard metrics. Completed forms are sent monthly to the SEPO Software Metrics Working Group and to the project Software Engineering Manager. The full set of 27 metrics requires 90 database cells, as well as an additional 53 cells devoted to background information. There are three types of databases for storing the metrics data:

1. All project status metrics data (metrics 1-2, 5-19, 22, 24-27) are kept in a project database that must be maintained by someone from the project.
2. All process metrics data (metrics 3, 4, 20, 23) are retained across projects in a process metrics database.
3. All financial metrics data (metric 21) are retained across projects in a financial metrics database.

The SEPO Software Metrics Working Group is responsible for maintaining the latter two databases.

Several data collection and support tools are available, including an internal LOC counting tool that supports Ada, Jovial, Fortran, VRTL, TRL, Pascal, and C, a requirements tracking tool (RTRACE), a program management tool (Microframe Program Manager), Paradox and FoxPro relational database tools, Excel and Lotus 123, and Harvard Graphics.

### B.1.4 Best Practices and Lessons Learned

This group had several problems and lessons learned in trying to standardize metrics. Examples include the following:

- Lines of code (LOC) is used to normalize productivity and quality data, but LOC always changes and is difficult to collect monthly. (This is why the "pulse points" are only reported for projects whose software has been formally tested.)
- For quality reports, design quality is defined based on the number of problems found from the start of integration to functional quality testing. Defect density is defined based on the number of problems found up to one year after functional quality testing. A problem occurs in defining design quality versus defect density for programs that do not have a formal software test.
- The software work breakdown structure needs four-digit charge numbers to accurately account for where effort is spent, but there is resistance by individual engineers to recall too many charge numbers or to spend too much time accounting for such specific details of their time.

Other problem areas include the following:

- Data collection is manual and hence problematic. There was much resistance to collecting the amount of data desired by the SEPO. Consequently, an Excel spreadsheet was developed to aid in reporting and summarizing the metrics data. However, the spreadsheet was also too complicated. It has been simplified and a pilot is underway to see how it will be accepted.
- It is difficult to get metrics information on software that was developed before the metrics program was in place, but which is still being evolved and maintained today.

The group offered the following advice to the DoD in regard to setting up a software metrics program:

- A contractor's own process and process improvement efforts should take precedence over DoD-wide mandates. It is difficult to improve when different processes and tools are imposed by various customers on different projects since you cannot maintain organizational consistency. This includes metrics requirements and work breakdown structures. Therefore, the government should issue guidance rather than requirements in the area of metrics collection. It should be acceptable to use the contractor's metrics set, assuming a basic level of adequacy.

### B.1.5 Future Plans and Directions

The group identified several areas of need and several future directions:

- One large area is the need for better tools to help collect data seamlessly, as part of the development process. Metrics data should flow directly from computer-assisted software engineering (CASE) and software engineering environment (SEE) tools, without human intervention.
- The tools that are available are not necessarily suited to the group's environment. It believes that it is going to have to create its own tools.
- Tools that are available sometimes do not measure what the group really needs to know. For example, reuse tools measure check-ins and check-outs rather than the amount of actual reuse or the amount of rework done on a reusable product.
- Function points are being investigated as an alternative to the LOC measure.
- The group wants to be able to use measures across programs to determine the benefits from process improvements and to identify the best tools and methodologies.

## B.2 COMPANY B

This report covers metrics activities in the defense segment of a large aerospace company. Programs within the defense segment are geographically dispersed. A software process group, which is part of a centralized software engineering organization, has responsibility for defining and promulgating improved software engineering practices, including metrics, throughout the entire defense group. It created and documented an initial set of measures in 1990. Programs were (and still are) encouraged to provide feedback to the process group reflecting their experience and judgement as to the benefit of each of the metrics. This feedback formed the basis for a significant revision of the metrics set, completed in 1993. The update also incorporated improvements derived from the Capability Maturity Model (CMM) of the Software Engineering Institute (SEI). The current version of the metrics standard defines the metrics set, gives guidance for implementing and using the metrics, and establishes requirements for consistent collecting and reporting of metric data.

Compliance to the standard is based upon the technical credibility of the software process group, and its ability to acquire and communicate "best practices" to programs throughout the defense segment. The programs have found that the return from their metrics activities is worth the investment they make. The software process group does high-level analyses of metrics data across the defense groups and feeds the results back to the programs. The software process group also supports programs in their use of metrics, provides electronic templates, gets feedback on which metrics are appropriate, and modifies the metrics set based on program feedback.

Corporate-level metrics activity at Company B is currently focused on the definition of a corporate metrics standard based on group experience. The corporate metrics set is a proper subset of the metrics set defined by the defense group. The Naval Air Systems Command (NAVAIR) metrics set also is a subset of the defense group's set.

### B.2.1 Metrics and Reports

There are three primary focuses of the metrics used by the defense group: estimation, program management, and process improvement.

For cost estimation and pricing, it uses parametric models such as SEER, COCOMO, and PRICE-S. The group uses a process called "similar to". Project characteristics data is collected and stored, along with project cost data, in a defense group program experience repository (example characteristics data: degree of customer involvement with development details, references to design documentation, skills of people who worked on project). The characteristics data is used to determine whether one project is "similar to" another. Cost data for past, similar projects provide the basis for software cost estimating of new projects and for calibration of cost models.

For program management and process improvement, the group uses a set of 21 indicators, where each indicator is typically composed of several metrics. A brief description of the 21 indicators follows.

1. **Cost/Schedule Deviations.** Includes actual cost of work performed; budgeted cost of work scheduled; budgeted cost of work performed; cost variance (budgeted minus actual work performed); and schedule variance (budgeted cost of work performed minus budgeted cost of work scheduled).

2. **Requirements Progress.** Includes expected total number of software requirements (functional, performance, and interface) at completion; cumulative number of planned requirements; and cumulative number of actual requirements.

3. **Requirements TBDs (unresolved requirements at SSR (Software Specification Review)).** Includes planned number of unresolved requirements; actual number of unresolved requirements; forecasted actual number of unresolved requirements.

4. **Development Progress.** Includes percent of planned and actual cumulative preliminary design completions; percent of planned and actual cumulative detailed design completions; percent of planned and actual cumulative code and unit test completions.

5. **Test Progress.** Includes cumulative number of planned and actual developed test cases; cumulative number of planned and actual developed test procedures; cumulative number of planned and actual passed test procedures; cumulative number of planned and actual test reports.

6. **CDRL Schedule.** Includes cumulative number of software CDRLs (Contract Data Requirements List) due to the program office; cumulative number of software CDRLs delivered to the program office; cumulative number of software CDRLs delivered to the program office still requiring approval; cumulative number of software CDRLs delivered to the program office and approved.

7. **Internal Review Progress.** Includes cumulative number of internal reviews planned; cumulative number of internal reviews conducted.

8. **Requirements Stability.** Includes cumulative number of requirements (usually the number of "shalls") specified, expressed as a percentage that is baselined as 100% at the SRR; cumulative number of requirements additions as a result of resolved TBDs; cumulative number of new requirements additions; cumulative number of requirements deletions; cumulative number of requirements modifications.

9. **Software Size.** Includes maximum budgeted size (may be adjusted for engineering changes); expected size in thousands of source lines of code (KSLOC) (developed plus reused); estimated developed code size; actual developed code size; reused code size. It is used to manage the size of the software work.

10. **Incremental Capability Delivered.** Includes original planned number of requirements delivered for each release; current planned number of requirements delivered for each release; planned point in time for each release; actual point in time for each release.

11. **Defect Resolution.** Includes predicted total number of opened Software Change Reports (SCRs); actual total number of opened SCRs; total number of opened SCR that have been dispositioned as approved, withdrawn, rejected, deferred; total number of SCRs whose change has been authorized; total number of SCRs whose change has been verified; total number of SCRs whose change has been completed.

12. **Code-Impact Defect Resolution.** Includes predicted total number of approved SCRs that affect source code; actual total number of approved SCRs that affect source code; total number of approved code-impact SCRs that have been verified; total number of approved code-impact SCRs that are completed.

13. **Defect Profile.** Includes number of new approved critical SCRs (prevent operation of the system); number of new approved significant SCRs (degrade operation of the system); number of other new approved SCRs; defect density (all approved SCRs/total code size); expected defect density at delivery.

14. **Defect Response Time.** Includes total number of SCRs that have been dispositioned, grouped by age of being opened (one week, two weeks,..., over six weeks); total number of SCRs that have not yet been dispositioned, grouped by age; total number of approved SCRs, grouped by length of time opened before changes were verified.

15. **Defect Characterization.** Includes defects are characterized as to TYPE (original defect, repeat defect, induced defect, enhancement), SEVERITY, ORIGIN (life cycle phase or "manuals" or "environmental support"), ACTIVITY (activity during which the problem was found: analysis, design, implementation, integration and test, operational use), CAUSE, and NATURE (missing, wrong, extra).

16. **Product Evaluation Findings Characterization.** Includes product evaluations are characterized similarly to defects.

17. **Review Action Latency.** Includes number of open action items distributed by number of weeks since origination.

18. **Staffing Profile.** Includes planned number of software personnel; actual number of software personnel; number of unplanned personnel losses.

19. **Staff Experience.** Includes average number of years embedded software development experience; desired average number of years embedded software development experience; average number of years experience in application domain (e.g., tactical missiles); desired average number of years experience in application domain (e.g., tactical missiles).

20. **Staff Training.** Includes total number of software personnel needing a particular type of training (e.g., Ada programming); total number of software personnel completing a particular type of training.

21. **Target Processor Utilization.** Includes amount of resident memory, mass storage, processor throughput, and input/output (I/O) bandwidth available for use; maximum amount of resource capacity allocated for current use; actual amount of resource use; estimated amount of resource use.

In practice, not all projects use all of the indicators. All of the metrics are recommended. Programs may tailor the metrics requirements based on their contracts and specific program needs, and document the tailoring in their program plans.

Broad, subjective status readings are also generated based on the 21 software indicators and other program circumstances. Status readings look at seven areas (cost, schedule, product, process, tools, personnel, and equipment), each of which is rated as critical, unsatisfactory, satisfactory, or excellent. The metrics standard gives guidance on which indicators to use in evaluating each status area, and how to assign a rating level to each area.

A single composite software status rating, composed of a weighted sum of the seven area status ratings, is also calculated.

### B.2.2 Levels of Reporting

Indicators are calculated monthly and used by software program managers for project management. Some indicators are collected and presented for each computer software configuration item (CSCI), while other indicators are collected and presented for a whole program. Indicators may also be collected for each CSCI but summarized across the program.

Status readings and composite status ratings are used by program management and may be reported to upper management.

The software indicator data is also provided monthly to the software process group and stored in their central repository. Analyses on this data, such as analysis of staffing needs, analysis of defect characterization to determine effectiveness of product evaluations, and analysis of the effectiveness of their metrics, are fed back to the programs.

### B.2.3 Tools and Repositories

File Maker Pro and Excel are two of the tools the group uses. The emphasis is on keeping data organized and using it on a daily basis. All of this is done using simple tools.

All project metrics information is reported back to the software process group and kept in a central repository. Projects are encouraged to report their data electronically, over the company network.

They are not moving in the direction of having a corporate-level repository.

### B.2.4 Best Practices and Lessons Learned

Company B has seen several benefits from their use of metrics.

- Metrics support the use of cost models, by helping in their calibration to the company's own work environment. It is beginning to find consistency of model performance within product lines. (There are still not enough samples to get consistency across product lines.)
- Metrics data have formed the basis for estimating new proposals.

- Metrics form a basis for process improvement. For example, defect characterizations were used to assess and improve the goodness of their technical product evaluations.

Of all the metrics in their set, the progress indicators, requirements stability, software size, defect metrics, product evaluation characteristics, and resource utilization metrics are perceived to be the most important.

One metrics problem encountered is quantifying staff experience.

One lesson learned is that it helps to have uniform practices across a company: it makes it easier to move people across groups by reducing the "learning curve", and makes group or company-level metrics analysis feasible.

Advice to DoD:

- Would not favor trying to collect atomic data on a national level—there is too much diversity. Concentrate instead on good metrics practices, letting individual companies define their own specific metrics.
- Would not hurt to standardize on a core set of indicators that should be used on every program (with high-level definitions).

### B.2.5 Future Plans and Directions

Company B is currently defining a corporate-level metrics standard and continuing to implement new methods of communicating best practices to programs.

## B.3 COMPANY C

Company C is a large computer company operating in both the Department of Defense (DoD) and commercial arenas. This report describes corporate-level metrics activities related to software quality metrics for use across the commercial sector of the company.

In 1990, a corporate instruction established aggressive software quality goals. A software measurement steering committee was formed, with one representative from each division producing software products, plus additional representatives from company-wide groups such as the corporate database group. The steering committee produced two documents describing corporate-level metrics to be used to track software quality, and is responsible for maintaining and further evolving the documents. The original required metrics set contained five metrics that were used routinely but not regularly for executive presentations. As of 1994, the set has been modified to include seven metrics that are required to be presented quarterly to executive-level company management. Most divisions have adhered to the definition and usage requirements set for these metrics, with slight deviations in usage by some divisions.

### B.3.1 Metrics and Reports

The goal of the software quality metrics is to achieve significant quality improvements by 1994. A further requirement is that each new product and release have better quality than its predecessor. Seven required corporate quality metrics will be reported quarterly, accompanied by other data and charts that further identify the causes of any trends identified.

1. **Product Defects.** This measure reports the total number of unique defects reported for a product, either by customers, by internal testing, or by development. (A "defect" is a "bug"; a "problem" is anything a customer perceives to be a problem in using the product, including defects. One defect may result in more than one problem report by more than one customer.) The number of defects are reported by month, for the current release, the previous release, and all older releases of a product. In addition to actuals, predicted numbers are plotted. Previously, the number of defects was normalized by dividing it by million shipped source LOC, but the divisor was later eliminated because of lines of code (LOC) differences in different languages and because it hid the true customer perception.

2. **Customer Problems.** This measure reports the total number of problem reports from customers for a given product. It covers all releases of the product and is reported monthly. It includes usability concerns as well as defective operations. This metric also was originally divided by LOC, but recently the divisor was eliminated. Current focus is on classifying customer problem reports more finely.

3. **Customer Satisfaction.** Customer satisfaction is measured by routinely administered surveys. Two related reports are required for each product. The first shows overall customer non-satisfaction, reported as the percentage of customers who were overall dissatisfied or neutral about the product. The percentage is reported for the current release and for all releases and should be reported quarterly. The second report shows customer non-satisfaction broken into eight categories (capability, usability, performance, reliability, installability, maintainability, documentation, and service), and calculated over the current release of a product.

4. **Service Defects.** This metric gives a count of the number of problem fixes that themselves contained an error. The total number of "service" defects are reported, as well as the number of such defects discovered by a customer. These defects have a higher level of visibility to the customer, since the customer originally identified them as errors and sees that they have not been fixed correctly. In the past, service defects were divided by the total number of defect fixes, but now Company C looks just at the raw number of service defects. The goal is to get this number to zero.

5. **Problem Resolution Time.** This measure shows the average age (from the date the problem was reported to the date the final solution was completed) for software problems of each severity level that were closed during the month.

6. **Cycle Time.** Cycle time is defined as the number of months to produce the product, starting at the point where 10% of development resources have been expended and ending at the first shipment of the product to a customer. The cycle time of all versions and releases of a product should be shown on the same chart. Both planned and actuals are given, with the size of the product release noted on the actuals line.

7. **Service Cost.** This measure is the total dollar cost expended in servicing all versions and releases of a product. Both actual and predicted costs are reported, on a monthly basis.

In addition to the seven required metrics, there are 10 optional metrics that can be used as supplemental measures by product and organizational management. These 10 metrics are categorized as customer view metrics (problem severity days, defect severity days), post-process view metrics ("where caused" defects, problems per user month, development cycle time histogram, fix quality, cycle time/productivity, reuse measurements), and in-process view metrics (internal process cycle time, defect backlog).

1. **Problem Severity Days.** All customer reported problems are weighted according to their severity. Severity days is the product of the weight and the number of days the problem is open. Severity days for each severity level are summed across all problems and graphed.

2. **Defect Severity Days.** A weighting and summing of customer reported defects similar to that done for problem severity days.

3. **"Where Caused" Defects.** This metric measures the quality of newly developed code. It is defined as the total number of unique defects found in the new code/size of the new code.

4. **Problems per User-Month.** This metric is defined as the total number of problems encountered by users with a given version of a product/total number of user-months the version has been in-use. Total user-months is the sum over all customers of the number of months the customer has had the product version.

5. **Development Cycle Time Histogram.** This metric shows the percentage of new code whose development cycle time falls within each of three time ranges (< 18 months, 18-30 months, > 30 months). The same information is shown for historical data and for newly planned development over the next two years.

6. **Fix Quality.** This metric depicts the number of errors made while fixing reported defects in a product. The goal is zero. Fix quality is defined as 100*(number of erroneous fixes/number of fixes shipped). The metric is reported in a histogram, by quarters. Fix Quality is similar to Service Defects. Service Defects are charged against the date a defect is corrected; Fix Quality is charged against the date a defect is caused.

7. **Cycle Time/Productivity.** Cycle time and productivity (code size/people years) of a product release are plotted as a percentage of historical cycle time and productivity rates. The N-1st release and N-2nd release data for the product are also plotted, as well as (optional) the plan for the next release.

8. **Reuse Measurements.** Three metrics are used: reuse percentage, reuse cost avoidance (estimated cost of developing and maintaining equivalent sized new code, based on actuals for new code), and reuse value added ((developed code size + reused code size + size of code contributed to other's reuse)/developed code size).

9. **Internal Process Cycle Time.** Baseline, current, and benchmark cycle times are given for chosen sub-process areas (e.g., system test, bug fix, integration and build).

10. **Defect Backlog.** This metric tracks the number of open defects reported by customers, per product.

A to-be published article addressing the quality initiatives taken by one product group lists the following metrics used by the group, in addition to the required metrics: phase-based defect removal patterns, phase effectiveness, in-process escape rate, percent of interface defects, integration and build defect arrivals, root cause analysis of defects, number of delinquent defect fixes, inspection effort and coverage, number of unit test defects found before and after code integration, testing defect arrivals by phase, testing defect rates by phase, defect severity distribution, late performance changes, and actual vs. plan for phase activities such as completion of design reviews, execution of test cases.

### B.3.2 Levels of Reporting

The metrics defined in this summary are reported to executive-level management and are used to track quality improvements over product lines over time. In-process measurements are defined within product organizations (see examples given in preceding section) and used for project management and product quality management.

### B.3.3 Tools and Repositories

The interviewee's division keeps predictions for such data as the number of defects, the number of problems, and customer satisfaction in a database. Later, it can do predicted vs. actual analyses using the data. Other divisions are believed likely to do similar things.

The seven required corporate-level measures are kept in a corporate database. The internal system that is used to log and/or fix problems feeds into this database as well. There is also an internal tool that calculates problem resolution time and other measures across the corporation.

### B.3.4 Best Practices and Lessons Learned

At the corporate level, the most beneficial metric is customer satisfaction. It provides the most direct measure of how it is doing as a company. For two of the past three years, 50% of an employee's pay bonus has been based on the customer satisfaction level reported for his or her division.

Advice to the DoD:

- Keep measures as simple as possible. For example, division by lines of code (LOC) or calculation of a productivity rate does not work too well across groups. Keep measurements to essential, basic elements.
- The most important thing is to compare self against self over time, not to do comparisons among different groups.

The product group mentioned above experienced a 30% reduction in the defect rate reported overall by formal testing and a 60% reduction in the defect rate reported by system testing. The reductions are attributable to front-end process improvements (including defect prevention process, improved inspection process targeted at the coding phase and at interface errors) identified by the analysis of metrics data. The group has learned from its experience that in-process metrics cannot be used in a piecemeal fashion, but must be integrated, must be interpreted in the context of any models used, and must be compared with the group's history.

### B.3.5 Future Plans and Directions

In addition to maintaining and evolving the corporate metrics documents, the steering committee is currently working on the following:

- In-process measurements. Currently, each division/lab/product is using its own set. The goal is to define overall in-process measurements for all divisions.
- Object-oriented metrics. LOC is just one example of a common metric that does not work well with object-oriented development.

- Productivity measure. It is currently based on LOC. Function points might be more suitable to object-oriented software. The group is addressing issues surrounding function points, such as the difficulty of developing/purchasing tools to count them.

## B.4 COMPANY D

Company D, a commercial avionics software developer, uses the metrics and tools developed by Quantitative Software Management, Inc. (QSM). Company D is actually a division of a larger company, so they will be called a "division" in this summary. The division's process lead, who is responsible for defining their software development process, attended a free training class by QSM in the mid-1980's. He became convinced that QSM's approach provided a better way to manage his products. He started with a low-level program that trained people in QSM's methods. This program provided immediate benefits, and led to pressure being place on upper management to establish a metrics program.

Due to their efforts, there is now a division-wide requirement on all projects to report a metric set based on QSM's metric set. The division uses the book *Measure for Excellence* by Larry Putnam (QSM's founder) as its guidelines for metrics definitions. Most of the division management is QSM trained and owns a copy of this book. This approach fits with the corporate culture, which strongly prefers the use of off-the-shelf, documented standards to expending scarce resources to develop company-specific standards.

### B.4.1 Metrics and Reports

The division metrics are primarily project management metrics, based on QSM metrics. Four are key and are measured dynamically:

- Size: Effective SLOC (adjusted for reuse).
- Cycle time: Time elapsed between key milestones.
- Effort: Staff hours.
- Reliability: Key is mean time to defect, after release. Tracked are occurrences of defects, relative to calendar time, taking exposure time into account.

Other metrics include schedule performance index, cost performance index, and a productivity index (using QSM definitions). The productivity index (PI) for the division is a 10-year rolling, running average, and is used to estimate effort and time for new projects. Sometimes they adjust the PI for a new product based on comparisons to existing product lines (for example, if it is a new type of product for the division). Divisional level MBOs (management by objectives) are based on the productivity index. All of these metrics are reported to the division process lead and tracked across the division.

### B.4.2 Levels of Reporting

All the metrics listed in the previous section are reported to the division process lead and tracked across the division.

### B.4.3 Tools and Repositories

The division has used the entire QSM tool set for three to four years, and some of the tools have been used for much longer. These include:

- Software Lifecycle Management (SLIM): Develops estimations of project schedule, staffing, costs, and reliability.
- SLIM Control: Performs dynamic project tracking.
- Size Planner: Provides software size estimations.
- Productivity Analysis Database System (PADS): Provides a historical database.

In the process lead's opinion, the biggest difference between QSM's approach and a COCOMO-based approach to project management is availability of tools to support QSM's approach (the estimation equations are similar). One attractive and unique aspect of the QSM estimation tools is that they take a statistical approach and show risk bands around each number.

There are two types of repositories: dynamic (information from an ongoing project) and static (information from completed projects). For dynamic information, the "SLIM Control" tool provides a dynamic project tracking database for each individual project, supporting comparisons of actual project measures to estimations. For static information, PADS is used to record end-of-project summary data across the entire division. Data is collected via a manual form and then entered into the PADS database. PADS data collection began in the mid-1980s, when the database was initially populated with project data.

### B.4.4 Best Practices and Lessons Learned

Company D's experience with QSM's approach has been very positive. Lessons learned include the following:

- People hate and fear to be measured, so trust must be established first. The organization got over this hump because (1) there was a high-level requirement for measurement and (2) groups got benefits quickly. Requests for more money from management or more time from their customer could then be justified, based on their metrics data.

- An outside consultant should be used to teach training courses, preferably an outstanding trainer. People put more faith in outside expertise.
- Adoption of a documented, standard metrics set developed elsewhere may save time and money.

### B.4.5 Future Plans and Directions

The division is considering expanding the application of QSM to hardware development and systems engineering.

## B.5 COMPANY E

E is a large company that produces a wide range of electronics products for commercial as well as Department of Defense (DoD) customers. The information in this report was gathered from a published article written by one member of a corporate champion group for software process improvement, as well as from a phone interview with the same individual, now working in a commercial products group. Opinions and advice expressed come from this one individual and do not necessarily represent company-endorsed ideas.

A company-wide software metrics initiative started several years ago, driven partly by published studies showing the usefulness of software metrics in improving the technical and management practices of software engineering. Senior management established policy requiring the use of metrics in eight areas (delivered defects and delivered defects per size, total effectiveness throughout the process, adherence to schedule, estimation accuracy, number of open customer problems, time that problems remain open, cost of nonconformance, and software reliability) while subsequent discussions among representatives of several business units added a ninth (software productivity).

A metrics working group, with participation across business units, was established in the 1980s. Using Basili's Goal-Question-Metric approach, the group established metrics goals addressing the nine measurement areas and came up with questions and metrics to support the goals. The result, after 3 years, was a set of 10 common software metrics to be used across the company, to track improvement over time across projects. The metrics working group also defined or adapted metrics to be used by projects for in-process control, including specialized metrics for formal software reviews and for software test. It helped deploy metrics to software development groups, partly by developing and teaching a two-day training workshop on metrics. Requirements for data collection, analysis, and feedback were developed and provided to tools groups involved in automating software metrics. In addition, criteria were established to use in evaluating commercially available metrics tools.

Adherence to established company-level metrics requirements varies across business groups and is affected partly by their ability to comply. One survey of managers and software engineers indicated that 67% of them are using the software review metrics developed by the metrics working group.

### B.5.1 Metrics and Reports

There are several metrics sets being used within Company E, each contributing to the company's broad goals of achieving a better understanding of its software processes and identifying changes to improve productivity, quality, and cycle time. In addition to the 10 process improvement metrics and the in-process control metrics mentioned in the introductory paragraphs, it has recently developed a set of five common metrics to be reported to senior managers and executives. A description of each of these three sets follows.

The first set, which is reported to the most senior levels of management within E, contains five metrics, whose descriptions follow.

1. **Software development process and product quality.** This metric reports the number of in-process faults per delta KAELOC, the number of defects per delta KAELOC, and the number of defects per total released KAELOC. The data is reported for a whole division by quarters. (KAELOC is thousands of assembler-equivalent physical source lines of code. Delta source size is the size of the source code added, deleted, or modified from the previous software release. An error is a problem found during the review of the phase in which it was introduced; a defect is a problem found later than the phase review; a fault is either an error or a defect.)
2. **Customer satisfaction.** Based on customer satisfaction surveys, the metric reports on customer perception of product quality, timeliness, technology, communication, responsiveness, value to the customer, and performance. A scale of 0 to 10 is used.
3. **Cycle time and productivity.** This metric reports by quarters the calendar months per released delta KAELOC, the calendar months per released total KAELOC, and the delta KAELOC per staff-hour.
4. **Software engineering technology roadmap.** This metric reports on plans for different software technologies, as well as implementation to date of the plan.
5. **Software Engineering Institute (SEI) Key Process Area (KPA) profile.** This metric reports previous and current scores, from 0 to 100, in the 18 key process areas defined for maturity levels 2 through 5.

The second set contains 10 process-improvement metrics charts and is primarily reported to software quality assurance (SQA) and to division-level managers. The charts were devised to address the goals of improving project planning, increasing defect containment, increasing software reliability, decreasing software defect density, improving customer service, reducing the cost of nonconformance, and increasing software productivity. All metrics reported in the charts are calculated over all projects within a division and are typically plotted against time. The descriptions of the ten charts follow:

1. **Software Development Process Quality (in sigma).** This chart contains two metrics, In-Process Faults (IPF) and In-Process Defects (IPD). IPF is defined as in-process faults caused by delta software development /delta KAELOC; IPD is defined as in-process defects caused by delta software development/delta KAELOC.

2. **Released Software Quality (in sigma).** This chart contains two metrics, Total Released Defects total and Total Released Defects delta. Total Released Defects total is defined as the number of released defects/total KAELOC; Total Released Defects delta is defined as the number of released defects caused by delta software development/delta KAELOC.

3. **Customer Found Defects (in sigma).** This chart contains two metrics, Customer-Found Defects total and Customer-Found Defects delta. Customer-Found Defects total is defined as the number of customer-found defects/total KAELOC; Customer-Found Defects delta is defined as the number of customer-found defects caused by delta software development/delta KAELOC.

4. **Post-Release Problem Report Activity.** This chart contains two metrics, New Open Problems and Total Open Problems. New Open Problems is defined as the total new post-release problems opened during the month; Total Open Problems is defined as the total number of post-release problems that remain open at the end of the month.

5. **Post-Release Problem Report Aging.** This chart contains two metrics, Age of Open Problems and Age of Closed Problems. Age of Open Problems is defined as (total time post-release problems remaining open at the end of the month have been open)/(number of open post-release problems remaining open at the end of the month); Age of Closed Problems is defined as (total time post-release problems closed within the month were open)/(number of post-release problems closed within the month).

6. **Cost to Fix Post-Release Problems.** This chart contains one metric, Cost of Fixing Problems. Cost of Fixing Problems is defined as the dollar cost of fixing post-release problems within the month. An alternative definition for this metric is the percentage of organizational effort spent for software maintenance or fixes.

7. **Total Defect Containment Effectiveness.** This chart contains one metric, Total Defect Containment Effectiveness. Total Defect Containment Effectiveness is defined as the number of pre-release defects / (number of pre-release defects + number of post-release defects).

8. **Phase Containment Effectiveness.** This chart contains one metric, Phase Containment Effectiveness, shown for four phases, requirement, high-level design, low-level design, and code. Phase Containment Effectiveness is defined as the number of phase $x$ errors/(number of phase $x$ errors + number of phase $x$ defects).

9. **Estimation Accuracy.** This chart contains two metrics, Schedule Estimation Accuracy and Effort Estimation Accuracy. Schedule Estimation Accuracy is defined as actual project duration/estimated product duration; Effort Estimation Accuracy is defined as actual project effort/estimated project effort.

10. Software Productivity. This chart contains two metrics, Software Productivity total and Software Productivity delta. Software Productivity total is defined as total KAELOC/software development effort in person months; Software Productivity delta is defined as delta KAELOC/delta software development effort in person months.

Usage of the third type of metrics, in-process metrics, varies among business groups, and may typically be addressed in group-level metrics handbooks or may be more informally propagated through activities such as project status meetings. Examples (not exhaustive) of some in-process metrics used within E include the following:

1. **Life cycle and schedule tracking metrics**, typically shown in Gantt chart form.

2. **Cost and earned value tracking metrics**, including estimated, budgeted and actual costs of the project and earned value of the project (the sum of the budgeted cost for activities already completed by the project).

3. **Requirements tracking metrics**, including actual and projected number of requirements changes as well as sources and/or causes of requirements changes.
4. **Design tracking metrics**, including the number of requirements traceable into design over time as well as design complexity measures.
5. **Fault-type tracking metrics**, which analyze the types and numbers of faults being introduced during coding.
6. **Remaining defects metric**, which looks at numbers of faults found to date and predicts the numbers to be found within the next *n* months using a Rayleigh curve.
7. **Inspection metrics**, such as inspection rate, error density per review, and distribution of faults into categories.
8. **Problem severity/priority-tracking metrics**, including the number of open problems of each severity level over time, and the average age of open problems at each severity level over time.

### B.5.2 Levels of Reporting

In-process metrics are used by project management to predict, track, and improve project-level software development. The 10 division-level software metrics are reported to SQA and to division management, while the set of 5 high-level metrics are reported to senior managers and executives.

### B.5.3 Tools and Repositories

The Excel Spreadsheet is used company-wide to produce the five senior level reports that are used. Company C has developed a tool for generating the 10 metrics used at the division management level, although it is not used much. Some groups had already developed their own tool (typically, spreadsheets), others had the wrong platform, and still others were using project-specific metrics rather than the 10 metrics defined at the corporate level.

Repository usage is typically decentralized, local, and not consistent across the company. However, there is a database within a corporate group where subsets of the first set of metrics are provided by business units, and some tracking of where the business units are relative to these metrics has started. Such data is typically not visible or used at the project level.

### B.5.4 Best Practices and Lessons Learned

Several best practices and lessons learned emerged from the corporate metrics program:

- Metrics are not the goal. Metrics can only show problems and give ideas for solutions. The benefits come from the improvement actions that are taken in response to metrics.
- It is better to start with a set of metrics that address your metrics goals and improve these over time than to debate indefinitely trying to define the "perfect" metrics set.
- It is better to collect a smaller amount of metrics data rather than a burdensome amount. Identify what is most important (e.g., cost, schedule) and concentrate on that. The best software metrics are simple to understand, precisely defined, and objective.
- An organization must establish an atmosphere where the problems spotlighted by metrics are discussed in terms of how to improve the process to correct them. Metrics must not be used to evaluate individual software engineers.
- A metrics program implementation should involve its user community in defining and accepting the metrics set and should support users through training and consulting services. As a result of doing this, metrics definitions within E are close, if not identical, across the company.
- A metrics program should provide help in automating data collection, analysis, and feedback. Implementation of a metrics program is easier if basic (preferably automated) systems such as cost accounting, configuration management, and software problem reporting/tracking are in place.
- Company E encouraged localized data storage, analysis, and feedback, so that data can be used in the context of a single product group or business unit with projects of similar size and complexity. It felt that this also made their metrics initiative more manageable. To satisfy requests for benchmarking data across projects, the company may later connect localized databases across a network. Some projects have prepared for the benchmarking usage of metrics by collecting and storing project-description data along with their metrics data.

The following benefits have come from the metrics program:

- Company E has seen quality, productivity, and cycle-time improvements that are attributable to process improvement actions taken as a result of metrics data. For example, one division has achieved a 50-times reduction in the defect density of its released software over a three and a half year period. Benefits well outweigh the cost of the metrics program, which, as an example, has been reported in two divisions as 1% or less of personnel resources.
- Company E has achieved significant cost reductions due to improved quality.
- Metrics have helped improve ship-acceptance criteria.
- Estimation accuracy metrics have helped many projects improve their techniques for estimating schedule, effort, and quality.
- Software problem-related metrics help in allocating personnel resources to problem fixes versus new development.
- In general, metrics focus attention on process improvement and help motivate improvement. They also help people focus on actions with quantifiable results.

Advice to the DoD:

- Establishment of a common DoD industry-wide repository is not recommended. There would be too many problems to overcome at this time. As organizations become more sophisticated in the use of metrics and move to higher SEI maturity levels, some sharing of common metrics data may be possible. However, the problem of data inconsistency across organizations will still need to be addressed.
- It may be easier to get agreement on common metrics definitions (versus a common repository). The SEI core metrics would be one good way to start.
- Encouraging contractors to use their own metrics data internally to improve the quality, productivity, and cycle time of their own projects would benefit the DoD greatly. The side benefit of internal company improvement, cost reduction, is also a direct benefit to the DoD. For the same reasons, encouraging contractors to use the SEI CMM to achieve higher CMM levels would benefit the DoD.
- DoD's overall objective should be to encourage contractor improvements through internal process and metrics activities and tracking, rather than to establish the reporting of a common set of metrics to a repository. A contractor's

track record and the dollars charged for a successful project are the types of metrics that can be used to distinguish among contractors.

### B.5.5 Future Plans and Directions

For DoD funding purposes, metrics buy-in, training, and process improvement and tool support are the areas that need attention and produce the greatest benefit, rather than new metrics definitions or other research and development efforts.

## B.6 COMPANY F

Company F is a large, diversified company that includes contract software development for commercial, Department of Defense (DoD), and government non-DoD customers among its profit-making activities. The information in this report gives insight into the software metrics activities within one division of the company.

Divisional-level metrics usage, less mature than project-level usage, began with the push to increase the Software Engineering Institute (SEI) maturity level. Total quality management (TQM) and integrated product teams were additional positive forces on management. Project-level metrics usage has been around for a longer time. However, acceptance of metrics at this level differs depending on the customer and the customer-induced culture. Personnel in DoD Subsystem Program Offices (SPOs), having two-year tours of duty, tend to require metrics showing good performance on their shifts. They require projects to manage to cost and try to gain some visibility into contractor software process. Commercial customers also are typically interested in both cost and time-to-market performance of their contractors. Some non-DoD government customers, however, have long-term associations with their projects and focus on delivery of the right product rather than on schedule or cost. Within this subculture, there is a lack of incentive to manage to cost and an accompanying lackadaisical attitude to metrics collection.

Company F has a corporate quality and integrity policy. The division, in addition, has its own software process policies and specific metrics specifications.

### B.6.1 Metrics and Reports

At the divisional level, metrics goals are to improve cost, schedule, and quality performance within an application domain. More specifically, the company looks at how to improve delivery time, how to capture the cost of delivering its product, and how to improve product quality, where quality is measured in terms of the cost of the rework applied to the product. It identifies two aspects to quality improvement: improvement in the ability to elicit requirements that will satisfy the end user, and improvement in the implementation of the requirements.

Similar metrics goals are found at the project level. In addition, there may be customer-required metrics information, although most of the customers, with the exception of recent DoD customers, have not placed much emphasis on metrics information.

Metrics that are commonly used within the company include the following:

- Size, which may be measured in terms of lines of code or function points. Function points tend to be used early in the life cycle, but are not universally accepted.
- Complexity, which is identified in terms of the type of application (e.g., inhouse tool, business application, ground support, flight software).
- Defect Counts, which focus on defects that are observable by the end user, as defined by the Institute of Electrical and Electronics Engineers. Therefore, requirements, design, and code defects are emphasized.
- Cost and Schedule.

Defect counts and cost and schedule are predicted and analyzed relative to the size and complexity of the project.

The division specifies 11 process and product metrics, although not all are equally valued. They include extent of requirements changes, requirements stability, training, process assessment, peer review results, computer resource utilization, manpower loading, and development and test progress.

### B.6.2 Levels of Reporting

At the divisional level, the divisions look at how projects are managed to cost and schedule, and how quality costs affect the bottom line. They analyze different processes (including training) on different projects to see how they affect cost, schedule, and defect counts.

The Software Engineering Process Group (SEPG) collects information from the projects' inspections and problem change reports databases and does various analyses including review and test effectiveness (no. of errors found/time spent), cycle time to bring defects to closure, and error severity.

### B.6.3 Tools and Repositories

Spreadsheet tools and databases are commonly used in metrics activities. The SEPG maintains a central repository, which contains information collected from project inspections databases and problem change report databases.

### B.6.4 Best Practices and Lessons Learned

Metrics problems and lessons learned include the following:

- Company F has experienced problems predicting schedule, cost, and expected defects based on the lines of code (LOC) of a product as well as problems re-estimating the LOC-based cost in the face of requirements additions or changes.
- You have to be able to control the software process by controlling the evolution of the product. There should be one database to handle change control of the product, including change authorization forms. If direction is not given at the divisional level, each project will develop its own change control system and forms.

Advice to the DoD:

- Every government agency seems to have its own criteria and requirements for metrics, which is disturbing. There should be one focal point and coordinator for the DoD. It would be beneficial if the equivalent of the National Institute of Standards and Technology (NIST) existed to set common criteria for all of software development.
- There should be a closer tie-in between requirements volatility and estimates. Practices such as demonstration/validation (DEM/VAL) are helpful in this respect.
- Current Ballistic Missile Defense Organization (BMDO) focus on process improvement as part of contracts is beneficial and should be extended to the other services.

### B.6.5 Future Plans and Directions

There is an SEPG initiative to collect non-recurring development costs and recurring maintenance costs, cycle times, and defect counts across projects and put them into a central repository for analyses.

## B.7 COMPANY G

Company G represents the government sector of a corporation that also has a large presence in the commercial computer and software industry.

Company G has a well-developed software engineering process. Its process documentation includes a software engineering manual (first version, 1977), software metrics standards for corporate-level metrics (1990), and metrics data collection procedures. Similar process documentation exists for systems engineering, hardware engineering, and product integration and test. Several pages in the software engineering manual are devoted to describing project-level metrics, but these are not as crisply defined as the corporate-level metrics.

Although software metrics standards did not appear until 1990, the current software collection structure was begun in 1989. Company G has five years of metrics data in the current form. Prior to that, data collection in a different form began in 1973. All told, 20 years of metrics data exist.

Company G's goals for its corporate metrics program are to become the best in breed in metrics data collection and analysis, to provide an integrated infrastructure that supports low-cost metrics operations, thereby encouraging program-level metrics usage, and to align with government metrics policy to the extent feasible. Metrics must be applicable to multiple lifecycle models, methodologies, and programming languages.

Current efforts are focused on creating integrated standards across disciplines (systems engineering, hardware engineering, software engineering, integration and test) and corresponding integrated measurements. Company G is working on "rules of thumb" for interpreting and reacting to metrics, for inclusion in the new standards. It is also planning for training on the new standards. Currently, it has no formal measurements training program. Information is dispersed through Software Engineering Process Groups (SEPGs) at each site.

### B.7.1 Metrics and Reports

Corporate-level software metrics goals are to assess the health of the software engineering discipline and to assess the health of the business. Metrics at the corporate level are used to provide comparables in estimating new proposals and to track trends (e.g., productivity, quality, where defects are caught and where they are inserted).

Each project is required to report six types of software metrics data yearly to the corporate level: lines of source code, labor months, defects, effort distribution, practice compliance, and elapsed time. Source code is measured in logical lines and is broken down by language used, by code type (application, executive, diagnostic, horizontal microcode, support, other), and by whether the code is new, modified, retained, reused, or ported. Defects are broken down by the phase discovered and by the original source of the error.

Project-level metrics goals are project management and process improvement. Individual projects augment the corporate metrics set to support day-to-day project management.

### B.7.2 Levels of Reporting

Measurement data is reported up the management chain, but this does not usually benefit projects directly because of the reporting time lag and difficulties with comparing data across projects. Sites tend to collect metrics data by project. Software Quality Assurance (SQA) gathers all project data at the site to deliver to the corporate level. A project that cannot produce the required metrics is perceived to be in trouble.

### B.7.3 Tools and Repositories

Tools are selected on a project or site basis. One company-wide tool under development is a source code line counter that will count logical statements. The company is finding it difficult to count changed logical lines of code. Automation supporting its data collection process needs to be improved. Company G is looking at commercial off-the-shelf (COTS) tools on personal computers to help in this area. It is also looking at workstation-based tools to help increase automation of metrics reporting. Several metrics repositories exist. At one site, SQA gathers data from projects and analyzes it for trends. Historical metrics data is also used in proposal preparation and project planning.

### B.7.4 Best Practices and Lessons Learned

Best practices and lessons learned include the following:

- Although involved in a corporate-level metrics program, our contact's opinion is that the most valuable usage of metrics is at the local level, for day-to-day project management.
- Therefore, edicted, company-level metrics must be valuable at the project level as well.

- Comparisons of software metrics across projects are highly sensitive and are invalid without proper characterization.
- Logical source statements are a better code measure than physical lines.
- Data that is hard to collect is less likely to be valid.
- It is not productive to collect data that you do not know how to analyze or use.
- Metrics are not static—they change with technology and process maturity, and must adapt to close loopholes, correct miscommunications, and account for outliers.
- The focus of software metrics should be on the clarity and completeness of primitive measures.

Problems or issues identified in Company G's metrics program include the following:

- Building an infrastructure to support low-cost operations of a metrics program.
- Responding to different metrics requirements from different government organizations.
- Getting good metrics data from subcontractors who are competitors.

### B.7.5 Future Plans and Directions

Company G identifies the following challenges for the future: refinement of metrics definitions; fitting metrics to projects to provide necessary, sufficient, effective, and efficient measurement; matching up metrics data collection categories with project work breakdown structures; analyzing measurements and proactively responding to their messages; identifying metrics and data collection standards for managing subcontracted work; and identifying metrics that can be compared across projects for managing the business.

Current focus is on creating integrated measurements that span the company's technical disciplines and include project management as well.

## B.8 COMPANY H

Company H, a large, leading developer of commercial personal computer software, has a company-wide standard set of metrics. The effort to develop the standard set began three years ago. Each project may choose additional metrics and decide on what to report.

Many of the company's internal management courses emphasize software metrics; the concept of metrics is embedded as part of the company process.

### B.8.1 Metrics and Reports

Monthly status reports are sent from each project to top management which include key dates for projects, in particular release to manufacturing dates as published, currently planned, and previously planned.

Currently, projects select their own metrics (other than key dates). Common project metric goals and specific metrics include the following:

- Quality (number of system crashes, outstanding problem reports)
- Progress (schedule, number of units designed, status)
- Performance (system throughput)

While there are no standard corporate-level metrics, some metrics are commonly provided by projects to higher levels. These include the following:

- Fagan S-curve (number of bugs found in development life cycle to date)
- Bug clustering (which modules have the most bugs)
- Churning metric (which modules have been changed the most)
- Slip charts (estimated release-to-manufacturing date versus calendar day)

A very large list of metrics is provided by the company's upper management to its projects to provide ideas of what metrics might be collected. These metrics can be broken into the following metric categories, with some sample metrics for each category included. Note that many are lists and many are subjective.

- **Project Completion**: Size in source lines of code (SLOC), executable size, number of bugs (total and by severity), mean time between failure (MTBF), effort (in person-years), sorted list of files most checked in, sorted list of functions with the most reported bugs, list of tools used.

- **Customer Satisfaction**: Units sold, phone calls per unit sold, ease of learning/ installing/customization/access/used.
- **Specification**: List of competitor features not in product, customer suggestions.
- **Design**: List of procedures that read/write global data, defect removal cost.
- **Code**: Percent of code used unchanged from another project, percent of adapted code, complexity (McCabe and Halstead), average module size.
- **Cost of Development and Marketing**: Cost of development, testing, user education, marketing, program management.
- **Supplier**: Cost to determine vendor product's quality.
- **Team**: Experience (of developers, testers, program management, marketing, user education), average number of days of training per year, list of major risks, staff turnover, workspace.
- **Testing**: Find/fix ratio, code coverage, bebugging score (percent of planned bugs found by testing).
- **Corporate Customers**: MTBF, mean time to repair, probability of availability.
- **Employee (Subjective) Satisfaction**: Feeling of achievement, recognition, work satisfaction, sufficient responsibility, opportunity for advancement, learning and growing, satisfactory quality of product.
- **Defect**: Severity, symptoms, where and how found, where and when and how fixed.

The company is currently considering developing documentation metrics.

### B.8.2 Levels of Reporting

A standard organizational model for metrics reports is generally followed on every project.

### B.8.3 Tools and Repositories

A standard defect tracking database is used by all projects. Standard metrics are gathered and compared across projects using this database.

### B.8.4 Best Practices and Lessons Learned

Company H's informal approach to metrics collection worked well when the company was small, but as the company has grown, there is now a perceived need to formalize and record metrics information.

The four metrics commonly given at the corporate level (Fagan S-curve, bug clustering, churning metric, and slip charts) appeared to be some of the more useful metrics.

### B.8.5 Future Plans and Directions

This company is continuing to refine its core set of metrics. There is ongoing effort to improve its development methodology using metrics as a basis.

## B.9 COMPANY I

This company is a major defense contractor in a wide range of military areas. It is a division of a much larger company that has substantial commercial business. Most of its software development is done through a group devoted to one military area, which has its software staff matrixed out to the groups specializing in other business areas.

The company has a 20-year history of metrics usage. The chronology is roughly as follows:

- 1972–1980: It evolved their current set of metrics.
- 1980–1985: It defined the set of reports they wanted to generate from the metrics;
- 1985 to the present: It has been refining the contents of these reports.

The company goal in collecting project metrics is to ensure healthy projects (i.e., to support project management and risk management). Software project managers use metrics data to manage their projects. If a project is proceeding without apparent difficulty, senior management does not look at much of the data. If based on divisional norms that a project seems to be in trouble, management makes use of all of the metrics data that is collected or produced to help diagnose why the project is ailing and how to help cure it.

The company has corporate-wide policies in place for monitoring process maturity level and continuing process improvement, as well as a wide range of management and financial measures such as earnings, return on sales, and others. It currently is addressing additions to the corporate policy to expand the required project and product quality metrics.

### B.9.1 Metrics and Reports

The metric sets of Schultz or Rozum are similar to what this company has been using successfully for 20 years. It considers this set the minimum essential set for successfully managing projects.

Company I's most beneficial metric, "without a doubt," is the rate chart, a profile of planned versus actual production of units through all life cycle phases.

The most recent addition to the company's set of reports is defect density across the life cycle (requirements, design, code, and test defects, normalized by lines of code (LOC)). It is a refinement of previous defect-type reports, one that enables them to reduce defects during development rather than after testing begins. It also provides a way of implementing an Software Engineering Institute (SEI) Level 4 requirement according to SEI TR-87-23 and possibly according to the more recent SEI Capability Maturity Model (CMM).

### B.9.2 Levels of Reporting

The company routinely collects a large number of metrics at the project level and reports them monthly to division level and program management.

### B.9.3 Tools and Repositories

The company recently completed a two-year internal development of a metrics tool that makes it easier to enter metrics data (requiring each data item to be entered only once), and that calculates and combines data to produce the charts, plots, and reports it uses. The company could not find a suitable commercial tool for its purposes.

### B.9.4 Best Practices and Lessons Learned

Prior to the above tool development, automation of metrics collection and analysis was a major challenge that each project leader had to solve. Now the company feels that benefiting from lessons learned on projects is a new challenge requiring further automation of project data files and the ability to maintain historical records across many projects.

For DoD, the company recommends that mandatory data collection should be limited to what is needed to meet concrete management goals.

### B.9.5 Future Directions

Technology transfer and "buy in" at the working project level continue to have high importance.

## B.10 COMPANY J

J, a well-known company in the aerospace industry, develops products commercially and for the Department of Defense (DoD). This report addresses metrics practices used on the commercial side of company J. It reflects the perspective of an individual in an internal organization that is chartered to run the metrics program for all commercial software product groups.

Company J uses a metrics framework that is based on the Capability Maturity Model of the Software Engineering Institute (SEI). Different metrics are used at different level of process maturity. Although it agrees somewhat with the SEI metrics set, the company's set is better adapted to its purposes. It feels that trying to measure individual Key Process areas (KPAs) is not practical.

### B.10.1 Metrics and Reports

The primary goal in using metrics is to improve the company's software development and maintenance process ("metrics is the heart of all process improvement"). It collects measurements in four basic areas:

- Size (function point is standard, lines of code (LOC) also used. It is transitioning to fully function points).
- Cost (effort).
- Quality (defects, trouble reports, change requests, etc.).
- Time (e.g., calendar time/phase, time/trouble report, cycle time to fix problems, time/requirement change).

Company J tailors the use of the basic measurements and additional metrics activities based on a project's maturity level. Projects in a Level-1 organization collect just the four basic metrics. They are collected only at the end of the project, and they stay at the project level. Their experience is that level 1 projects do not have the resources to do more, and can better expend their efforts in other areas.

At Level 2, the same four basic metrics are collected, but with more attributes. A total of 8 to 12 metrics are used. Metrics are collected per phase rather than just once per project. Data is more meaningful, but is still kept at the project level.

At Level 3, the same metrics are collected, but with again more attributes. At this level, the company does not look at projects but at the functional organization. By now, standards exist, and there are consistent data and metrics definitions across the organization. Metrics data is put into a repository. The company uses metrics to look at the process to see how effective it is and where to improve it.

At Level 4, the same metrics again, with more attributes, are used, looking at the whole division. The division manager can make decisions based on metrics. At this level, correlations are done between the company process (which is standard across the division) and the characteristics of its products (e.g., defects, cycle time, productivity).

### B.10.2 Levels of Reporting

The metrics group collects and analyzes metrics data and produces reports covering all commercial groups in the company.

### B.10.3 Tools and Repositories

Repositories are used for organizations at Levels 3 and above.

### B.10.4 Best Practices and Lessons Learned

Company J offered five pieces of advice to the DoD that reflect its lessons learned and best practices:

- Common metrics definitions are important.
- Start simple at the lower levels of process maturity. Projects at these levels are not mature enough for complex measures.
- Do not act on metrics data for at least 6 months (12 to 24 months is better)—let it stabilize. Before six months, there is not enough data for it to be meaningful and it has not been collected over a broad-enough spectrum of development activities. Look at trends over a year to see if the data has stabilized.
- At Levels 1 and 2, only use data for project management. These organizations are not consistent across projects, so data only has meaning at the project level.
- Standardization occurs when you go from Level 3 to 4, not before.

### B.10.5 Future Plans and Directions

Not discussed.

## LIST OF REFERENCES


[AFSC 1986] Air Force Systems Command. Jan 1986. *Software Management Indicators*. AFSCP 800-43. Washington DC: Andrews Air Force Base.

[AFSC 1987] Air Force Systems Command, Jan 1987. *Software Quality Indicators*. AFSCP 800-14. Washington DC: Andrews Air Force Base.

[Andres 1990] Andres, Don. November 1989. *Software Project Management Using Effective Process Metrics: the CCPDS-R Experience*. TRW Technology Series TRW-TS-89-01. Redondo Beach, CA: TRW.

[Army 1992] Department of the Army. 30 September 1992. *Software T&E Metrics: Recommended Metric Set*. Department of Army Pamphlet 73-1, Vol. 6, Part VII, Chapter 17, Section I (Draft).Washington, DC.

[Arthur 1991] Arthur, James D. and Richard E. Nance. August 1991. The Formalization of Software Quality Indicators Within the Objective/Principles/Attributes Framework. *Third Annual Software Quality Workshop*, Alexandria Bay, NY.

[Basili 1984] Basili, Victor R. and David M. Weiss. 1984. A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering* (November): 728-738.

[Carleton 1992] Carleton, Anita D et. al. 1992. *Software Measurement for DoD Systems: Recommendations for Initial Core Measures*. SEI Technical Report SEI-92-TR-19. Pittsburgh, PA: Software Engineering Institute.

[Carleton 1993a] Carleton, Anita D. July 1993. *Software Measurement: Embedding Quantitative Principles into Software Engineering*. SEI

Special Report SEI-93-SR-17. Pittsburgh, PA: Software Engineering Institute.

[Carleton 1993b] Carleton, Anita D., Robert E. Park, and Wolfhart B. Goethert. 1993. *Measurement Definitions for DoD Systems: Recommendations for an Initial Core Set*. Pittsburgh, PA: Software Engineering Institiute.

[CECOM 1993] U. S. Army Communications - Electronics Command Software Engineering Directorate. July 12, 1993. *Streamlined Integrated Software Metrics Approach (SISMA) Guidebook: Application of STEP Metrics*. Ft. Monmouth, NJ.

[Chubin 1993] Chubin, Sherrie et al. August 1993. *Software Reuse Metrics Plan. Version 4.1.* Washington, D.C.: Defense Information Systems Agency, Joint Interoperability Engineering Organization, Center for Information Management.

[Daskalantonakis 1992] Daskalantonakis, Michael K. 1992. A Practical View of Software Measurement and Implementation Experiences Within Motorola. *IEEE Transactions on Software Engineering* (November): 998-1010.

[Dion 1992] Dion, Raymond. 1992. Elements of a Process-Improvement Program. *IEEE Software* (July): 83-85.

[DISA 1993a] Defense Information Systems Agency. June 1993. *Software Metrics Standardization: Action Plan*. Washington, D.C.: Defense Information Systems Agency, Joint Interoperability and Engineering Organization Information Processing Directorate.

[DISA 1993b] Defense Information Systems Agency. July 1993. *Software Metrics Standards Implementation Strategy*. Washington, D.C.: Defense Information Systems Agency, Joint Interoperability and Engineering Organization Information Processing Directorate.

[Dyson 1991] Dyson, Peter and James McGhan. 1991. *CECOM Executive Management Software Metrics (CEMSM) CEMSM Set Defini-*

*tion*. Indialantic, FL.: Software Productivity Solutions, Inc. (SPS-EMSM-00191)

[Dubin 1992] Dubin, Henry C. and Raymond A. Paul. March 1992. Management of Software Operational Testing. *ITEA Journal.*

[Fife 1993] Fife, Dennis W., Bill Brykczynski, and Beth E. Springsteen. September 1993. *Software Risk Assessment for DoD Acquisition Programs*. Alexandria, VA: Institute for Defense Analyses. IDA Paper P-2636.

[Florac 1992] Florac, William A. 1992. *Software Quality Measurement: A Framework for Counting Problems and Defects*. Pittsburgh, PA: Software Engineering Institute. SEI-92-TR-22.

[Goel 1993] Goel, Amrit L. et.al. March 1993. *Software Metrics: Analysis and Interpretation*. Draft STEP Technical Report.

[Goethert 1992] Goethert, Wolfhart B. 1992. *Software Efforts Measurement: A Framework for Counting Staff Hours*. Pittsburgh, PA: Software Engineering Institute. SEI-92-TR-21.

[Humphrey 1987] Humphrey, W.S. and W.L. Sweet. 1987. *A Method for Assessing the Software Engineering Capability of Contractors*. Pittsburgh, PA: Software Engineering Institute. SEI-87-TR-23.

[IEEE 1992] IEEE Computer Society. 1992. *IEEE Std 1061-1992: IEEE Standard for a Software Quality Metrics Methodology*. New York, NY: IEEE.

[ISO 1990] International Organization for Standardization. 1990. ISO/IEC Information Technology—Software Product Evaluation—Quality Characteristics and Guidelines for Their Use. ISO/IEC DIS 9126-90.

[Kan] Kan, Stephen H. et al. [n.d.]. AS/400 Software Quality Management. Accepted by *IBM Systems Journal* special issue on software quality.

[Koch 1994] Koch, C.F. May 1994. *NAWCADWAR Software Measurement Guide (Vers. 2)*. Warminster, PA: Naval Air Warfare Center.

[McGarryF 1993] McGarry, Frank. 1993. Experimental Software Engineering: 17 Years of Lessons in the SEL. Briefing. Greenbelt, MD. NASA Goddard Space Flight Center.

[McGarryJ 1992a] McGarry, John. May 1992. *Software Development Metrics Department of the Navy*. Newport, RI: Naval Undersea Warfare Center (NUWC).

[McGarryJ 1992b] McGarry, John. November 1992. *Navy Submarine Combat Systems Software Metrics Program*. Newport, RI: Naval Undersea Warfare Center (NUWC).

[McGarryJ 1993] McGarry, John. 1993. *Software Development Metrics Qualitative Assessment Structure*. Newport, RI: Naval Undersea Warfare Center (NUWC).

[McGarryJ 1994] McGarry, John H. and Cherl L. Jones. 1994. *Application of a Quantitative Software Metrics Assessment Process to Military Software Development Programs*. Boston, MA: Electro International 1994.

[McWhinney 1992] McWhinney, Mark S. and John H. Baumert. 1992. *Software Measures and the Capability Maturity Model*. Pittsburgh, PA: Software Engineering Institute. SEI-92-TR-25.

[MICOM 1991] Army Missile Command. February 1991. *Software Management Indicators: User's Manual*. Red Stone Arsenal, AL: US Army Missile Command, Software Engineering Directorate.

[Mills 1988] Mills, Everald E. 1988. *Software Metrics*. Pittsburgh, PA: Software Engineering Institute. SEI-CM-12-1.1.

[Mitre 1985] Mitre Corporation. 1985. *Software Reporting Metrics*. Bedford, MA: Mitre Corporation. ESD-TR-85-145. Prepared for Electronics Systems Division of the Air Force.

[Nance 1993] Nance, Richard E. and James D. Arthur. 1993. *Navigating the Tar Pits or An Holistic Approach to Software Quality Assessment*. Gaithersburg, MD: National Institute for Standards and Technology. NIST Lecture Series on High Integrity Systems.

[NAVAIR 1992] NAVAIR. 18 June 1992. *Avionics Software Metrics*. Washington DC: Naval Air Systems Command. AIR-546, AVION Instruction 5235.1.

[NUSC 1991] Naval Underwater System Center. July 1991. *Software Development Metrics*. Newport, RI: Naval Underwater System Center.

[Park 1992] Park, Robert E. 1992. *Software Size Measurement: A Framework for Counting Source Statements*. Pittsburgh, PA: Software Engineering Institute. SEI-92-TR-20.

[Paul a] Paul, Raymond A. [n.d.] Metrics to Improve the US Army Software Development Process.

[Paul b] Paul, Raymond A. [n.d.] *US Army Software T&E Panel (STEP) Initiatives for Software Risk Management.*

[Paul 1992] Paul, Raymond A. November 1992. Metric-based Neural Network Classification Tool for Analyzing Large-Scale Software. *Proceedings of the 1992 IEEE International Conference on Tools with AI*, Arlington, VA.

[Putnam 1992] Putnam, Lawrence H. and Ware Myers. 1992. *Measures for Excellence*. New York: Yourdon Press.

[Rifkin 1991] Rifkin, Stan and Charles Cox. 1991. *Measurement in Practice*. Pittsburgh, PA: Software Engineering Institute. SEI-91-TR-16.

[RL 1985] Rome Laboratory. 1985. *Rome Laboratory Software Quality Framework (RLSQF)*. Rome Laboratory. RADC-TR-85-37. (Three volumes; also available from NTIS as AD-A153-988, AD-A153-989, and AD-A153-990.)

[RL 1991] Rome Laboratory. 1991. *QUES Final Technical Report*. Rome Laboratory. TR-91-407. (Two volumes; also available from NTIS as AD-A252-679 and AD-A252-976.)

[RL 1992] Rome Laboratory. 1992. *A Guide to Software Quality Control*. Rome Laboratory. TR-92-316. (Two volumes.)

[RL 1993] Rome Laboratory. 1993. *Cooperstown I Workshop: Creating a National Vision and Force in Software Through Software Mea-*

*surement, August 30-September 1, 1993*. Cooperstown, NY: Rome Laboratory.

[Royce 1990] Royce, Walker. 1990. *Pragmatic Quality Metrics for Evolutionary Software Development Models*. Carson, CA: TRW. TRW-TS-91-01.

[Rozum 1992a] Rozum, James A. 1992. *Software Measurement Concepts for Acquisition Program Managers*. Pittsburgh, PA: Software Engineering Institute. SEI-92-TR-11.

[Rozum 1992b] Rozum, James. October 1992. *NAWCADWAR Software Measurement Guide*. Pittsburgh, PA: Software Engineering Institute. SEI/NAWC-92-SR-1.

[Sackman 1967] Sackman, Harold. 1967. *Computers, System Science, and Evolving Society*. New York: Wiley.

[SAF/AQ 1994] Department of the Air Force. 16 February 1994. Software Metrics Policy-Action Memorandum. Acquisition Policy 93M-017. Washington, DC.

[Schultz 1988] Schultz, Herman P. May 1988. *Software Management Metrics*. Bedford, MA: MITRE Corporation. NTIS Accession Number AD-A196 916. Sponsored by the AF Systems Command.

[SEI 1993] Software Engineering Institute. 1993. *SEI Quarterly Update, 2Q93*. Pittsburgh, PA: Software Engineering Institute.

[SPRC 1992] Software Practices Research Group. 1993. *Software Measurement Practices in Industry*.

[U.S. Army 1992] U.S. Army STEP Panel. 1992. *US Army Software T&E Panel Metrics Training*. Washington, D.C.

[Valett 1989] Valett, Jon D. and Frank E. McGarry. 1989. A Summary of Software Measurement Experiences in the Software Engineering Laboratory. *Journal of Systems and Software* 9: 137-148.

[Van Verth 1992] Van Verth, Patricia B. 1992. *A Concept Study for a National Software Engineering Database*. Pittsburgh, PA: Software Engineering Institute. SEI-92-TR-23.

[Wohlwend 1993] Wohlwend, Harvey and Susan Rosenbaum. 1993. Software Improvements in an International Company. *Proceedings of the 15th International Conference of Software Engineering*. New York: IEEE Computer Society Press.

# BIBLIOGRAPHY[1]


Adamov, Rade and Lutz Richter. 1990. A Proposal for Measuring the Structural Complexity of Programs. *Journal of Systems and Software* 12: 55-70.

Air Force Systems Command, Jan 1986. *Software Management Indicators*. AFSCP 800-43. Washington D.C.: Andrews Air Force Base.

Air Force Systems Command, Jan 1987. *Software Quality Indicators*. AFSCP 800-14. Washington D.C.: Andrews Air Force Base.

Ami Consortium. *A Quantitative Approach To Software Management*. London, UK: Ami Consortium.

Andersen O. October 1992. Industrial Applications of Software Measurements. *Information and Software Technology* 34/10: 681-693.

Andres, Don. November 1989. *Software Project Management Using Effective Process Metrics: the CCPDS-R Experience*. Redondo Beach, CA: TRW. TRW Technology Series TRW-TS-89-01.

Applications of Software Measurement. *See* ASM.

ASM93. 1993. *Proceedings of the 4th International Conference on Applications of Software Measurement,* November 7-11, 1993. Orlando, FL.

ASM92. 1992. *Proceedings of the 3rd International Conference on Applications of Software Measurement*. November 15-19, 1993. La Jolla, CA.

Baker, Albert L. et al. 1990. A Philosophy for Software Measurement. *Journal of Systems and Software* 12: 277-281.

Baker, Mark D. 1991. Implementing an Initial Software Metrics Program. National Aerospace Electronics Conference. May 20-24, 1991.


1 [n.d.] - no date identified; [n.p] - no publisher identified.

Basili, Victor R. 1990. Recent Advances in Software Measurement (Abstract for Talk). *12th International Conference on Software Engineering, March 26-30, 1990, Nice, France.*

Basili, Victor R. and David H. Hutchens. 1983. An Empirical Study of a Syntactic Complexity Family. *IEEE Transactions on Software Engineering* Vol. SE-9/No. 6 (November): 664-672.

Basili, Victor R. and Barry T. Perricone. 1984. Software Errors and Complexity: An Empirical Investigation. *Communications of the ACM* 27/1 (January): 42-52.

Basili, Victor R. and H. Dieter Rombach. 1988. The TAME Project: Towards Improvement-Oriented Software Environments. *IEEE Transactions on Software Engineering* (June): 758-773.

Basili, Victor R. and David M. Weiss. 1984. A Methodology for Collecting Valid Software Engineering Data. *IEEE Transactions on Software Engineering* (November): 728-738.

Berry, Rolland H. and George H. Wedberg. 1991. Metrics for Competitiveness. In *Proceedings of the Washington Ada Symposium, June 1991*, 119-140. New York: ACM.

Bhide, Sandhiprakash. 1990. Generalized Software Process-Integrated Metrics Framework. *Journal of Systems and Software* 12: 249-254.

Boeing Defense and Space Group. October 1993. *Engineering Operating Instruction on Software Metrics*. Seattle, WA: Boeing Corporation. Engineering Operating Instruction EOI 7-001 Rev A.

Boeing MIS Division. 1991. Software Metrics Reference Card. Seattle, WA: Boeing Computer Services.

Bourque, Paul and Vianney Cote. 1991. An Experiment in Software Sizing with Structured Analysis Metrics. *Journal of Systems and Software* 15: 159-172.

Bowman, Brent J. and William A. Newman. Software Metrics as a Programming Training Tool. *Journal of Systems and Software* 13: 139-147.

Brisneham, Linda. 1993. *Martin Marietta Astronautics Company Software Metrics Plan*. Panel Session Briefing. 1993 SEI [Software Engineering Institute] Symposium. Pittsburgh, PA.

Browne, J.C. and Mary Shaw. 1991. Toward a Scientific Basis for Software Evaluation. In *Software Metrics: An Analysis and Evaluation*, Alan Perlis, Frederick Sayward, and Mary Shaw, editors, 19-41.

Buchan, Fran. [n.d.] Weapon System Software Assessment: A Tool for Development and Oversight. Briefing. Alexandria, VA. ODASD(PR)IEQ

Card, David N. 1993. What Makes for Effective Measurement? *IEEE Software* (November): 94-95

Carleton, Anita D. July 1993. *Software Measurement: Embedding Quantitative Principles into Software Engineering*. Pittsburgh, PA: Software Engineering Institute. SEI Special Report SEI-93-SR-17.

Carleton, Anita D et. al. 1992. *Software Measurement for DoD Systems: Recommendations for Initial Core Measures*. Pittsburgh, PA: Software Engineering Institute. SEI Technical Report SEI-92-TR-19.

Carleton, Anita D., Robert E. Park, and Wolfhart B. Goethert. 1993. *Measurement Definitions for DoD Systems: Recommendations for an Initial Core Set.* Pittsburgh, PA: Software Engineering Institute.

Cartwright, Jeff. 1988. Book review of *Software Metrics: Establishing a Company-Wide Program* by Robert B.Grady and Deborah L. Caswell. *IEEE Computer* (April): 141.

Castellano, David et al. October 1993. *The Readiness Growth Model: A Quantitative Analysis of Software Risk.* Picatiny Arsenal, N.J.: U.S. Army Armament Research, Development and Engineering Center. Technical Report AR PAD-TR-93003.

CECOM. June 1993. *Quick Sight—A Program Manager's Metric Key Question Preference.* Fort Monmouth, N.J.: U.S. Army Communications Electronics Command (CECOM).

CECOM. July 1993. *Streamlined Integrated Software Metrics Approach (SISMA) Guidebook: Application of Step Metrics.* Fort Monmouth, N.J.: U.S. Army Communications Electronics Command (CECOM).

Cheadle, William G. November 25, 1991. *Software Estimating Process Description and Procedure (Preliminary Draft Review Copy).* Denver, CO: Martin Marietta Astronautics Group.

Christenson, D.A. et al. [n.d.] *Statistical Methods Applied to Software.* [n.p.]

Chruscicki, Andrew J. [n.d.] *Experiences with Software Quality Measurement in the US Defense Industry*. Griffiss AFB, NY: Rome Laboratory.

Chruscicki, Andrew J. [n.d.] *Models and Metrics for Measuring Software Scrap and Rework: Air Force Policy Recommendations*. Prepared for Lloyd K. Mosemann II, Deputy Undersecretary of Air Force for Communications, Computers and Logistics. Griffiss AFB, NY: Rome Laboratory, Software Engineering Branch.

Chubin, Sherrie et al. August 1993. *Software Reuse Metrics Plan. Version 4.1*. Washington, D.C.: Defense Information Systems Agency, Joint Interoperability Engineering Organization, Center for Information Management.

Clapp, Judith. 1993. Getting Started on Software Metrics. *IEEE Software* (January): 108 & ff.

Clapp, Judith and Saul Stanten. December 1992. *A Guide to Total Software Quality Control*. Bedford, MA: Mitre Corporation. RL-TR-92-316, Volume I and II.

Clay, A.W. et al. 1991. Quality Metrics at AG Communication Systems. *American Programmer* (September): 25-34.

Collins, James C. and William C. Lazier. 1993. Vision: The Greatest Companies Started with the Founders' Core Vision. *EMR* (Spring): 61-75.

Communications - Electronics Command. *See* CECOM.

Computer Software Modeling and Analysis (CSMA), Inc. June 1992. *Prototype Software Metrics Analysis, Project Management, and Risk Analysis Tools Investigation: Phase 2. Final Report*. Dewitt, NY: CSMA, Inc. Prepared for U.S. Army Operational Test & Evaluation Command, Alexandria, VA, and Rome Laboratory, RL/C3C, Griffiss AFB, NY.

Compton, Terry and Carol Withrow. 1990. Prediction and Control of Ada Software Defects. *Journal of Systems and Software* 12: 199-207.

Coupal, Daniel and Pierre N. Robillard. 1990. Factor Analysis of Source Code Metrics. *Journal of Systems and Software* 12: 263-269.

Cox, Guy M. [n.d.] *Sustaining a Software Metrics Programme in Industry*. Palo Alto, CA: HP Company, Corporate Engineering.

Cruickshank, Robert et al. December 1992. *Software Measurement Guidebook.* Software Productivity Consortium Report SPC-91060-CMC. Herndon, VA: Software Productivity Consortium.

Curtis, Bill. 1991. The Measurement of Software Quality and Complexity. In *Software Metrics: An Analysis and Evaluation*, Alan Perlis, Frederick Sayward, and Mary Shaw, editors, 203-224.

Curtis, Bill et al. March 1979. Measuring the Psychological Complexity of Software Maintenance Tasks with the Halstead and McCabe Metrics. *IEEE Transactions on Software Engineering* Vol. SE-5/No. 2: 96-104.

DACS. *See* Data and Analysis Center for Software.

Daskalantonakis, Michael K. 1992. A Practical View of Software Measurement and Implementation Experiences Within Motorola. *IEEE Transactions on Software Engineering* (November): 998-1010.

Data and Analysis Center for Software (DACS). 1991. Handouts. Griffiss AFB, NY: Rome Laboratory.

Data and Analysis Center for Software (DACS). 1992. *Proceedings of the Fourth Annual Software Quality Workshop: Experiences in Software Measurement and Implications for the Future, Bonnie Castle Resort, Alexandria Bay, New York, August 2-6, 1992.* Griffiss AFB, NY: Rome Laboratory.

Data and Analysis Center for Software (DACS). 1993. *DACS Guide: A User's Guide to DACS Products and Services*. Utica, NY: Data Analysis Center for Software.

Data and Analysis Center for Software (DACS). November 2, 1993. *Data and Analysis Center for Software (DACS) Steering Committee Charter*. Rome, NY: IIT Research Institute.

Defense Information Systems Agency. *See* DISA.

Dehnad, Khosrow. 1990. Software Metrics from a User's Perspective. *Journal of Systems and Software* 13: 111-115.

DeMarco, Tom. 1982. *Controlling Software Projects*. Englewood Cliffs, NJ: Prentice-Hall, Inc.

Denicoff, Marvin and Robert Grafton. Software Metrics: A Research Initiative. In *Software Metrics: An Analysis and Evaluation*, Alan Perlis, Frederick Sayward, and Mary Shaw, editors, 13-18.

Dion, Raymond. 1992. Elements of a Process-Improvement Program. *IEEE Software* (July): 83-85.

DISA. September 1993. *Military Handbook: Software Measurement Selection and Use. (MIL-HDBK-SWM)*. Draft. Washington, D.C.: Defense Information Systems Agency, Joint Interoperability Engineering Organization, Center for Standards.

DISA. June 1993. *Software Metrics Preliminary Assessment*. Washington, D.C.: Defense Information Systems Agency. Joint Interoperability and Engineering Organization, Center for Standards, Information Processing Directorate.

DISA. June 1993. *Software Metrics Standardization: Action Plan*. Washington, D.C.: Defense Information Systems Agency, Joint Interoperability and Engineering Organization Information Processing Directorate.

DISA. July 1993. *Software Metrics Standards Implementation Strategy*. Washington, D.C.: Defense Information Systems Agency, Joint Interoperability and Engineering Organization, Center for Standards, Information Processing Directorate.

Dubin, Henry C. and Raymond A. Paul. March 1992. Management of Software Operational Testing. *ITEA Journal*.

Dyson, Peter and James McGhan. March 1991. *CECOM Executive Management Software Metrics (CEMSM): CEMSM Set Definition (SPS-EMSM-0091)*. Indialantic, FL: Software Productivity Solutions, Inc.

Ejiogu, Lem O. 1991. TM: A Systematic Methodology of Software Metrics. *ACM SIGPLAN Notices* (January): 124-132.

Ellis, Walter. [n.d.] *Shepherding Software Reliability Growth Through Complexity Filters*. Software Process & Metrics, Inc.

European Strategic Programme for Research and Development in Information Technology (ESPRIT). April 1991. *METKIT: Training in Software Engineering Measurement: IT Developers, Best Practices of Leading Companies*. London, UK: Centre for Systems & Software Engineering, South Bank Polytechnic.

Evangelist, Michael. 1988. Complete Solution to the Measurement Problem. *IEEE Software* (January): 83-84.

Fenick, Stewart. 1990. Implementing Management Metrics: An Army Program. *IEEE Software* (March).

Fenick, Stewart. July 1993. Software Process Metrics Program Review to Software Engineering Directorate Division Chiefs and Supervisors' Management Metrics (Briefing). Fort Monmouth, NJ. U.S. Army CECOM Research, Development, and Engineering Center.

Fenick, Stewart and Harry F. Joner. 1992. So Much to Measure, So Little Time To Measure It: The Need for Resource Constrained Management Metrics Programs. *10th Annual Natural Conference on Ada Technology 1992.*

Fenton, Norman E. 1991. *Software Metrics: A Rigorous Approach.* London, UK: Chapman & Hall.

Fenton, Norman E. 1993. How Effective Are Software Engineering Methods? *Journal of Systems and Software* 22: 141-146.

Fenton, Norman and Austin Melton. 1990. Deriving Structurally Based Software Measures. *Journal of Systems and Software* 12: 177-187.

Fife, Dennis W., Bill Brykczynski, and Beth E. Springsteen. September 1993. *Software Risk Assessment for DoD Acquisition Programs.* Institute for Defense Analyses. Alexandria, VA. IDA Paper P-2636.

Florac, William A. 1992. *Software Quality Measurement: A Framework for Counting Problems and Defects.* Pittsburgh, PA: Software Engineering Institute. SEI-92-TR-22.

Gibson, Rose and Frank Arkell. October 1993. *Motorola Government & Systems Technology Group Software Maturity.* Briefing to U.S. Army CECOM.

Gilb, Tom. 1977. *Software Metrics.* Cambridge, MA: Winthrop Publishers, Inc.

Goel, Amrit L. et al. March 1993. *Software Metrics: Analysis and Interpretation.* Draft STEP Technical Report.

Goethert, Wolfhart B. 1992. *Software Efforts Measurement: A Framework for Counting Staff Hours.* Pittsburgh, PA: Software Engineering Institute. SEI-92-TR-21.

Grable, Ross. March 1993. *Object & Metrics: A Metrics Implementation Plan.* Redstone Arsenal, AL: U.S. Army Missile Command.

Grable, Ross. [n.d.] *Quality Metrics for Object-Oriented Ada.* Redstone Arsenal, AL: U.S. Army Missile Command.

Grable, Ross. [n.d.] Software Architecture Metrics. Briefing. Redstone Arsenal, AL: U.S. Army Missile Command.

Grable, Ross. 1991. Software Metrics Design. ACM Southeastern Regional Conference.

Grady, Robert B. and Deborah L. Caswell. 1987. *Software Metrics: Establishing a Company-Wide Program.* Englewood Cliffs, NJ: Prentice-Hall, Inc.

GTE. 1992. *Software Metrics Handbook, Version 6.0.*

Harrison, Warren. 1990. A Foreword to the Special Issue on Using Software Metrics. *Journal of Systems and Software* 13: 87-88.

Hausler, Philip A. August 5, 1992. Software Quality Using Cleanroom Software Engineering. Briefing. Gaithersburg, MD: IBM Corporation, General Sector Division, Cleanroom Software Technology Center.

Henry, Sallie and Roger Goff. 1991. Comparison of a Graphical and a Textual Design Language Using Software Quality Metrics. *Journal of Systems and Software* 14: 133-146.

Henry, Sallie and John Lewis. 1990. Integrating Metrics into a Large-Scale Software Development. *Journal of Systems and Software* 13: 89-95.

Hetzel, Bill. 1993. *Making Software Measurement Work.* Boston, MA: QED Publishing Group.

Hollis, Walker W. and Peter A. Kind. January 4, 1993. Memorandum for SEE Distribution: Preparation for Implementing Army Software Test and Evaluation Panel (STEP) Metrics Recommendations. Washington, D.C.: U.S. Army, Director of Information Systems for Command, Control, Communications, & Computers.

Hon, Samuel E., III. 1990. Assuring Software Quality Through Measurements: A Buyer's Perspective. *Journal of Systems and Software* 13: 117-130.

Huensch, G.D. et al. 1990. Managing the Final Phases of Software Development Using Software Reliability Modeling. In *Proceedings of the 13th International Switching Symposium, Stockholm, Sweden.*

Humphrey, W.S. and W.L. Sweet. 1987. *A Method for Assessing the Software Engineering Capability of Contractors.* Pittsburgh, PA: Software Engineering Institute. SEI-87-TR-23.

IBM Federal Systems. February 1993. *FSC Measurements*. Briefing.

IBM. July 1993. *Corporate Programming Measurements Definitions & Reference*. Document Number CPM-50.

IBM. December 1993. *Software Quality Corporate Measurements*. Draft.

IEEE Computer Society. 1993. *IEEE Std 1045-1992: IEEE Standard for Software Productivity Metrics*. New York, NY: IEEE.

IEEE Computer Society. 1993. *IEEE Std 1061-1992: IEEE Standard for a Software Quality Metrics Methodology*. New York, NY: IEEE.

International Organization for Standardization. ISO/IEC Information Technology—Software Product Evaluation—Quality Characteristics and Guidelines for Their Use. ISO/IEC DIS 9126-90.

Iuorno, Rocco and Robert Vienneau. July 1987. *Software Measurement Models: A DACS State of the Art Report*. Rome, NY: IIT Research Institute. Prepared for Rome Air Development Center, COEE, Griffiss AFB, NY.

Janusz, Paul E. August 1992. *Readiness Growth Model: A Quantitative Analysis of Software Risk*. Briefing. Picatinny Arsenal, NJ: U.S. Army, Armament Research, Development & Engineering Center (ARDEC).

Jeffery, D.R. [n.d.] *Software Process Quality: Requirements for Improvement from Industry*. Sydney, Australia: University of New South Wales, School of Information Systems.

Jenson, Richard L. and Jon W. Bartley. 1991. Parametric Estimation of Programming Effort: An Object-Oriented Model. *Journal of Systems and Software* 15: 107-114.

*The Journal of Systems and Software*. March 1993. Special issue on the Oregon Metric Workshop.

Kaman Sciences Corporation. September 22, 1992. *DACS Data Analysis Papers*. Utica, NY: Kaman Sciences Corporation. Prepared for Rome Laboratory, RL/C3C, Griffiss AFB, NY.

Kaman Sciences Corporation. May 7, 1991. *Linear Software Reliability Models Under Imperfect Debugging*. Utica, NY: Kaman Sciences Corporation. Prepared for the Rome Air Development Center (RADC/COEE). Griffiss AFB, NY.

Kan, Stephen H. 1991. Modeling and Software Development Quality. *IBM Systems Journal* 30/3: 351-362.

Kan, Stephen H. et al. [n.d.]. AS/400 Software Quality Management. Accepted by *IBM Systems Journal* special issue on software quality.

Kanko, Mark A. 1993. Of Gas Gauges and Software Metrics. *Crosstalk* Special Edition 1993: 8-18. Hill AFB, UT: Software Technology Support Center.

Keyes, Jessica (editor). 1993. *Software Engineering Productivity Handbook*. New York, NY: McGraw-Hill.

Koch, C.F. May 1994. *NAWCADWAR Software Measurement Guide (Vers. 2)*. Warminster, PA: NAWC-AD.

Lanphar, Robert. 1990. Quantitative Process Management in Software Engineering: A Reconciliation Between Process and Product Views. *Journal of Systems and Software* 12: 243-248.

Lasky, Jeffrey and Kevin Donaghy. May 1993. *Conflict Resolution (CORE) for Software Quality Factors*. Rochester, NY: Rochester Institute of Technology. RL-TR-93-80.

MacCabe, Thomas J. and Charles W. Butler. December 1989. Design Complexity Measurement and Testing. *Communications of the ACM* 32/12: 1,415-1,425.

McCall, James et al. April 1992. *Software Reliability, Measurement, and Testing Guidebook for Software Reliability Measurement and Testing*. San Diego, CA. Science Application International Corporation (SAIC). RL-TR-92-52. Two volumes.

McGarry, Frank. 1993. Experimental Software Engineering: 17 Years of Lessons in the SEL. Briefing. Greenbelt, MD: NASA Goddard Space Flight Center.

McGarry, John H. and Cherl L. Jones. 1994. *Application of a Quantitative Software Metrics Assessment Process to Military Software Development Programs*. To be presented at Electro International 1994, Boston, MA, May 1994.

McGarry, John H. August 4, 1992. *The Impact of Software Development Process Constraints on Software Product Quality*. Briefing. Newport, RI: Naval Undersea Warfare Center.

McGarry, John. November 1992. *Navy Submarine Combat Systems Software Metrics Program*. Newport, RI: Naval Undersea Warfare Center (NUWC).

McGarry, John H. May 1992. *Software Development Metrics*. Newport, RI: Naval Undersea WarFare Center (NUWC).

McGhan, James and Peter Dyson. February 1992. *CECOM Executive Management Software Metrics CEMSM Benefit Analysis (SPS-EMSM-00491-REV-A)*. Indialantic, FL.: Software Productivity Solutions, Inc.

McWhinney, Mark S. and John H. Baumert. 1992. *Software Measures and the Capability Maturity Model*. Pittsburgh, PA: Software Engineering Institute. SEI-92-TR-25.

Maness, Richard. February 17-19, 1992. *Software Architecture Sizing and Estimating Tool: SASET Training Course*. Martin Marietta.

Martin, Johnny and W.T. Tsai. February 1990. N-Fold Inspection: A Requirements Analysis Technique. *Communications of the ACM* 33/2 (February): 225-232.

Martin Marietta. February 1992. *Technical Report for the Software Metrics Tutorial (NTB-137-25-02-01)*. Falcon AFB, CO. Martin Marietta IS. Prepared for Strategic Defense Initiative Organization.

Martin Marietta Astronautic Group. May 1993. *Software Metrics Plan/Procedure*. Denver, CO: Martin Marietta.

Martin Marietta Information Systems. February 11, 1992. *Technical Report for the Software Metrics Tutorial*. NTB Program, Falcon AFB, CO. Prepared for the Strategic Defense Initiative Organization.

MICOM. February 1991. *Software Management Indicators: User's Manual*. Red Stone Arsenal, AL: U.S. Army Missile Command, Software Engineering Directorate.

Mills, Everald E. 1988. *Software Metrics*. Pittsburgh, PA: Software Engineering Institute. SEI-CM-12-1.1.

Missile Command. *See* MICOM.

Mitre Corporation. 1985. *Software Reporting Metrics*. Bedford, MA: Mitre Corporation. ESD-TR-85-145. Prepared for Electronics Systems Division of the Air Force.

Moller, K.H. and D.J. Paulish. 1993. *Software Metrics*. Piscataway, NJ: IEEE Press.

Moore, David. February 1993. Metrics. Internal memo. Redmond, WA: Microsoft Corporation.

Motorola, Inc. November 1991. *Common Motorola Software Metrics.* A Motorola Executive Summary.

Munson, John C. and Taghi M. Khoshgoftaar. 1990. Applications of a Relative Complexity Metric for Software Project Management. *Journal of Systems and Software* 12: 283-291.

Murine, Gerald E. 1985. On Validating Software Quality Metrics. Presented at the 4th Annual Phoenix Conference, Phoenix, AZ, March 1985.

Myrvold, Alan. 1990. Data Analysis for Software Metrics. *Journal of Systems and Software* 12: 271-275.

Nance, Richard E. and James D. Arthur. October 1993. *Navigating the Tar Pits or An Holistic Approach to Software Quality Assessment.* Gaithersburg, MD: National Institute of Standards and Technology. The NIST Lecture Series on High Integrity Systems.

NASA. March 1992. *Data Collection Procedures for the Software Engineering Laboratory (SEL) Database.* Greenbelt, MD: NASA Goddard Space Flight Center.

NASA. November 1990. *Manager's Handbook for Software Development.* Greenbelt, MD: NASA Goddard Space Flight Center.

NASA. October 1992. *Software Engineering Laboratory (SEL) Database Organization and User's Guide.* Greenbelt, MD: NASA Goddard Space Flight Center.

National Aeronautics and Space Administration. *See* NASA.

Naval Underseas Warfare Center. September 1993. *Fixed Distributed System Share Signal and Information Processing Segment: Software Development Metrics Analysis Summary.* Newport, RI: Naval Underseas Warfare Center.

Naval Underwater System Center. July 1991. *Software Development Metrics.* Newport, RI: Naval Underwater System Center.

Nusenoff, Ronald E. and Dennis C. Bunde. 1993. A Guidebook and a Spreadsheet Tool for a Corporate Metrics Program. *Journal of Systems and Software* 23: 245-255.

Office of the Inspector General. April 1989. *Management of Software for Mission-Critical Computer Resources.* Arlington, VA: DoD Office of the Inspector General. Audit Report No. 89-068.

Oman, Paul W. and Curtis R. Cook. 1990. Design and Code Traceability Using a PDL Metrics Tool. *Journal of Systems and Software* 12: 189-198.

Park, Robert E. 1992. *Software Size Measurement: A Framework for Counting Source Statements*. Pittsburgh, PA: Software Engineering Institute. SEI-92-TR-20.

Paul, Raymond A. November 1992. Metric-Based Neural Network Classification Tool for Analyzing Large-Scale Software. In *Proceedings of the 1992 IEEE International Conference on Tools with AI, Arlington, VA.*

Paul, Raymond A. [n.d.] Metrics to Improve the US Army Software Development Process.

Paul, Raymond A. September 1993. Software T&E Panel (STEP) Initiatives Status Report. STEP Briefing.

Paul, Raymond A. June 1992. Software T&E Panel (STEP) Metrics Overview. ARPA Briefing.

Paul, Raymond A. [n.d.] *US Army Software T&E Panel (STEP) Initiatives for Software Risk Management.*

Paulk, Mark A. et al. 1993. *Capability Maturity Model for Software, Version 1.1*. SEI Technical Report SEI-93-TR-24. Pittsburgh, PA: Software Engineering Institute.

Perlis, Alan, Frederick Sayward, and Mary Shaw. 1981. Table of contents, preface, and overview to *Software Metrics: An Analysis and Evaluation.* Cambridge, MA: The MIT Press.

Pfleeger, Shari and Clement McGowan. 1990. Software Metrics in the Process Maturity Framework. *Journal of Systems and Software* 12: 255-261.

Porter, Adam and Richard Selby. 1990. Evaluating Techniques for Generating Metric-Based Classification Trees. *Journal of Systems and Software* 12: 209-218.

*Proceedings of the TTCP Workshop on Software Metrics, May 21 - 24, 1990.* Rochester NY: Rochester Institute of Technology.

Putnam, Lawrence H. [various dates] Promotional material. McLean, VA: Quantitative Software Management (QSM).

Putnam, Lawrence H. and Ware Myers.1992. *Measures for Excellence*. New York: Yourdon Press.

Raytheon, Equipment Division, Software Systems Laboratory. [n.d.] *Quantifying the Benefit of Software Process Improvement.* Briefing.

Redmond, James A. and Reynold Ah-Chuen. 1990. Software Metrics—A User's Perspective. *Journal of Systems and Software* 13: 97-110.

Reifer, Donald J. July 23, 1990. *Joint Integrated Avionics Working Group Reuse Metrics and Measurement Concept Paper (Draft).* Torrance, CA: Reifer Consultants, Inc. Prepared for CTA, Inc., Ridgecrest, CA.

Rifkin, Stan and Charles Cox. 1991. *Measurement in Practice.* Pittsburgh, PA: Software Engineering Institute. SEI-91-TR-16.

Rome Laboratory. 1993. *Cooperstown I Workshop: Creating a National Vision and Force in Software Through Software Measurement, August 30–September 1, 1993.* Cooperstown, NY: Rome Laboratory.

Rome Laboratory. 1991. *Proceedings of the 3rd Annual Software Quality Workshop, Bonnie Castle Resort, Alexandria, Bay, New York, August 11-15, 1991.* Griffiss AFB, NY: Rome Laboratory.

Rome Laboratory. August 1993. *Software Quality Technology Transfer Consortium.* Griffiss AFB, NY: Rome Laboratory. Handouts on the consortium, membership benefits, membership requirements, member profiles. the Quality Evaluation System (QES) Tool, and the RL Software Quality Framework (RLSQF).

Royce, Walker. 1990. Pragmatic Quality Metrics for Evolutionary Software Development Models. Carson, CA: TRW. TRW-TS-91-01.

Royce, Walker. January 1990. *TRW's Ada Process Model for Incremental Development of Large Software Systems.* Carson, CA: TRW. TRW-TS-90-01.

Royce, Walker. October 1993. *UNAS Briefing.* Carson, CA: TRW.

Royce, Walker and Rhonda Mustard. [n.d.] *An Evolutionary Process Model Adapted to Software Product Development and Maintenance.* TRW System Engineering and Development Division.

Royce, Walker and Winston Royce. December 1991. *Software Architecture: Integrating Process and Technology.* Carson, CA: TRW. TRW-TS-91-04.

Rozum, James. October 1992. *NAWCADWAR Software Measurement Guide.* Pittsburgh, PA: Software Engineering Institute. SEI/NAWC-92-SR-1.

Rozum, James A. 1992. *Software Measurement Concepts for Acquisition Program Managers.* Pittsburgh, PA: Software Engineering Institute. SEI-92-TR-11.

Russell, Glen W. 1991. Experience with Inspection in Ultralarge-Scale Developments. *IEEE Software* (January): 25-31.

Russell, Meg. 1990. International Survey of Software Measurement Education and Training. *Journal of Systems and Software* 12: 233-241.

Sackman, Harold. 1967. *Computers, System Science, and Evolving Society*. New York: Wiley.

SAIC Metrics Working Group. 1993. *Handbook for Software Measurement Reporting Formats*.

SAIC Metrics Working Group. 1993. *Software Metrics Handbook and Collection Guide*.

Samadzadeh, M.H. and K. Nandakumar. 1991. A Study of Software Metrics. *Journal of Systems and Software* 16: 229-234.

Schultz, Herman P. May 1988. *Software Management Metrics*. Bedford, MA: MITRE Corporation. NTIS Accession Number AD-A196 916. Sponsored by the AF Systems Command.

Schwartz, Karen. September 1992. Army Gets Serious About Defining Software Metrics. *Government Computer News* 11/19: 70.

Selby, Richard W. 1990. Extensible Integration Frameworks for Measurement. *IEEE Software* (November): 83, ff.

Shaw, Mary. When is "Good" Enough? Evaluating and Selecting Software Metrics. In *Software Metrics: An Analysis and Evaluation*, Alan Perlis, Frederick Sayward, and Mary Shaw, editors, 251-262.

Smith, O.T. October 1993. Air Force Material Command Embedded Software Management Plan Software Development and Sustainment Metrics Task. Briefing. Robbins Air Force Base, GA: Air Force Material Command.

Software Business Management. 1993. Product Literature—DecisionVision 1. Westford, MA: Software Business Management.

Software Engineering Institute. 1993. *SEI Quarterly Update, 2Q93*. Pittsburgh, PA: Software Engineering Institute.

Software Practices Research Group. 1993. *Software Measurement Practices in Industry*.

Software Test and Evaluation Panel. *See* STEP.

SPARTA. May 1989. *Software Measurement Process: Evaluation of Software Measurement Support.* Arlington, VA: The Analytic Sciences Corporation. TR-9033-2. Prepared for the Strategic Defense Initiative Organization.

SPARTA, Inc., Teledyne Brown Engineering, and The Analytic Sciences Corporation. May 15, 1989. *SDS Software Measurement Plan Technical Report (Draft).* Arlington, VA: The Analytic Sciences Corporation. Prepared for the Strategic Defense Initiative Organization.

STEP. [n.d.] *Definition of the (STEP) Metrics Data Elements: AMC Guidance for Implementation of STEP Metrics.* Picatiny Arsenal, NJ: U.S. Army Armament Research, Development and Engineering Center.

STEP Panel. [circa 1990] *Summary Report: Findings and Recommendations from the Army Software Test and Evaluation Panel (STEP).*

STEP Panel. April 1990. T&E Roles.

Sulack, R.A. et al. 1989. A New Development Rhythm for AS/400 Software. *IBM Systems Journal* 28/3: 386-406.

Sunderhaft, Nancy L. and Robert Vienneau. May 1986. *STARS Measurement Survey Summary.* Rome, NY: IIT Research Institute, Data & Analysis Center for Software (DACS). Prepared for the STARS Joint Program Office, Arlington, VA, and the Rome Air Development Center, Griffiss AFB, NY.

Teledyne Brown Engineering, SPARTA, Inc., and The Analytic Sciences Corporation. September 28, 1990. *Verification of Metrics Models via Tools Application Technical Report.* Arlington, VA: The Analytic Sciences Corporation. Prepared for the Deputy for Engineering, Strategic Defense Initiative Organization.

Texas Instruments. June 1992. Software Metrics Tables.

Tolochko, Sharyn. October 1993. The Readiness Growth Model: A Quantitative Analysis of Software Risk. *EuroSTAR '93, London, U.K.*

Tolochko, Sharyn and David Castellano. [n.d.] *The Readiness Growth Model: A Quantitative Analysis of Software Risk.* Picatiny Arsenal, NJ: U.S. Army Armament Research, Development and Engineering Center.

Tso, Kam Sing. October 1991. *Complexity Metrics for Avionics Software.* Beverly Hills, CA: SoHaR Incorporated. Prepared for USAF Wright Laboratory.

U.S Air Force. 16 February 1994. Software Metrics Policy-Action Memorandum. Acquisition Policy 93M-017. Washington, D.C..

U.S. Air Force. 1993. *Draft Acquisition Policy 93M-017*. Washington, D.C.: Office of the Assistant Secretary of Air Force, Acquisition.

U.S. Air Force. 1993. *Guidelines for Successful Acquisition and Management of Software Intensive Systems: Weapon Systems and Management Information Systems*. Washington, D.C.: Headquarters, U.S. Air Force.

U.S. Army. June 1992. *Software T&E Metrics: Recommended Metrics Set*. Department of Army Pam 73-1, Vol. 6, Part IV, Chapter 17, section I.

U.S. Army CECOM Research, Development & Engineering Center, Software Engineering Directorate. August 2-6, 1992. Panel: Experiences with Management Metrics and Indicators. From *Rome Labs Fourth Annual Software Quality Conference, Alexandria, NY, August 2-6, 1992*.

U.S. Army Computer Systems Command. August 1984. *Software Quality Engineering Handbook*. Ft. Belvoir, VA: U.S. Army, Quality Assurance Directorate.

U.S. Army STEP Panel. 1992. *US Army Software T&E Panel Metrics Training*. Washington, D.C.

Valett, Jon D. and Frank E. McGarry. 1989. A Summary of Software Measurement Experiences in the Software Engineering Laboratory. *Journal of Systems and Software* 9: 137-148.

Van Verth, Patricia B. 1992. *A Concept Study for a National Software Engineering Database*. Pittsburgh, PA: Software Engineering Institute. SEI-92-TR-23.

Verner, June. [n.d.] *Software Process Maturity: Some Issues in Medium Sized Business Organizations*. Sydney, Australia: University of New South Wales, School of Information Systems.

Vienneau, Robert and James D. DeLude. October 22, 1993. *Rome Laboratory Research in Software Measurement*. Utica, NY: Kaman Sciences Corporation. Prepared for Rome Laboratory, RL/C3CB. Griffiss AFB, NY.

Wallace, Dolores R. and John C. Cherniavsky. April 1990. *Guide to Software Acceptance*. Gaithersburg, MD: U.S. Department of Commerce. NIST Special Publication 500-180.

Westinghouse Electronic Systems Group. 1993. *Software Metrics Handbook*.

White, Geoff. 1990. Software Metrics and Plagiarism Detection. *Journal of Systems and Software* 13: 131-138.

Willis, Ronald. October 1993. Software Metrics That We Use. Personal communications with the authors. Hughes Aircraft Company.

Whitty, Robin, Martin Bush, and Meg Russell. 1990. METKIT and the ESPRIT Program. *Journal of Systems and Software* 12: 219-221.

Wohlwend, Harvey and Susan Rosenbaum. 1993. Software Improvements in an International Company. *Proceedings of the 15th International Conference of Software Engineering*. New York: IEEE Computer Society Press.

Yu, Weider D., D. Paul Smith, and Steel T. Huang. 1990. Software Productivity Measurements. *AT&T Technical Journal* (May/June).

# LIST OF ACRONYMS

| | |
|---|---|
| AF | Air Force |
| AFAS | Advanced Field Artillery System |
| AFATDS | Advanced Field Artillery Tactical Data System |
| AFIT | Air Force Institute of Technology |
| AFMC | Air Force Materiel Command |
| AFSC | Air Force Systems Command |
| AFSCP | Air Force Systems Command Pamphlet |
| AIS | Automated Information System |
| AMC | Army Materiel Command |
| ARDEC | Army Armament Research, Development, and Engineering Center |
| ARPA | Advanced Research Projects Agency |
| ASARC | Army System Acquisition Review Council |
| AVION | AVIOnics division of NAVAIR |
| BMDO | Ballistic Missile Defense Organization |
| CASE | Computer-Aided Software Engineering |
| CCPDS-R | Command Center Processing and Display System Replacement |
| CDA | Central Design Agency |
| CDR | Critical Design Review |
| CDRL | Contract Data Requirements List |
| CECOM | Army Communications - Electronics Command |
| CEMSM | Army Communications - Electronics Command (CECOM) Executive Management Software Metrics |
| CFS | Center for Standards |
| CIM | Center for Information Management |
| CM | Configuration Management |
| CMM | Capability Maturity Model |
| COCOMO | COnstructive COst MOdel |
| COTS | Commercial off-the-Shelf |
| CPU | Central Processing Unit |
| CRM | Computer Resource Management |
| CSC | Computer Software Component |
| CSCI | Computer Software Configuration Item |
| CSU | Computer Software Unit |

| | |
|---|---|
| DA | Department of the Army |
| DACS | Data Analysis Center for Software |
| DAS | Decision Analysis System |
| DDR&E | Director, Defense Research and Engineering |
| DEM/VAL | Demonstration/Validation |
| DID | Data Item Description |
| DISA | Defense Information Systems Agency |
| DISC4 | Director of Information Systems for Command, Control, Communications, and Computers |
| DLA | Defense Logistics Agency |
| DoD | Department of Defense |
| DSI | Delivered Source Instructions |
| DSLOC | Delivered Source Lines of Code |
| DSMC | Defense Systems Management College |
| DUSA(OR) | Deputy Under Secretary of the Army for Operations Research |
| ECP | Engineering Change Proposal |
| ESC | Air Force Electronics System Center |
| ESD | Air Force Electronic Systems Division |
| FARV | Future Armored Resupply Vehicle |
| FATDS | Field Artillery Tactical Data Systems |
| FCA | Functional Configuration Audit |
| FFRDC | Federally Funded Research and Development Center |
| FQT | Formal Qualification Testing |
| FTE | Full-Time Equivalents |
| FY | Fiscal Year |
| GAO | Government Accounting Office |
| GQM | Goal-Question-Metric |
| HLL | High-Level Language |
| I/O | Input/Output |
| IDA | Institute for Defense Analyses |
| IDD | Interface Design Document |
| IEC | International Electrotechnical Commission |
| IEEE | Institute of Electrical and Electronics Engineers |
| IM | Information Management |
| IPD | In-Process Defects |
| IPF | In-Process Faults |
| IRAD | Internal Research and Development |
| IRS | Interface Requirements Specification |
| ISO | International Standards Organization |

| | |
|---|---|
| IT | Information Technology |
| JIEO | Joint Interoperability and Engineering Organization |
| JLC | Joint Logistics Commanders |
| JSTARS | Joint Surveillance Target Attack Radar System |
| KPA | Key Process Area |
| KAELOC | Thousands of Assembler-Equivalent Physical Lines of Code |
| KSLOC | Thousand Source Lines of Code |
| LOC | Lines of Code |
| MAISRC | Major Automated Information System Review Committee |
| MBO | Management By Objective |
| MICOM | Army Missile Command |
| MIS | Management Information System |
| MSLOC | Maintained Source Lines of Code |
| MTBF | Mean Time Between Failures |
| NASA | National Aeronautics and Space Administration |
| NAVAIR | Naval Air Systems Command |
| NAWC | Naval Air Warfare Center |
| NAWC-AD | Naval Air Warfare Center - Aircraft Division |
| NBCRS | Nuclear, Biological, and Chemical Reconnaissance System |
| NDI | Non-Developmental Item |
| NIST | U.S. National Institute of Standards and Technology |
| NUSC | Naval Underwater Systems Center |
| NUWC | Naval Undersea Warfare Center |
| O/T | Overtime |
| OO | Object Oriented |
| OPTEC | Army Operational Test and Evaluation Command |
| ORD | Operational Requirements Document |
| OT | Operational Test |
| OT&E | Operational Test and Evaluation |
| PADS | Productivity Analysis Database System |
| PC | Personal Computer |
| PCA | Physical Configuration Audit |
| PDL | Program Design Language |
| PDR | Preliminary Design Review |
| PDSS | Post-Deployment Software Support |
| PEO | Program Executive Officer |
| PI | Productivity Index |
| PM | Program Manager |
| PO | Program Office |

| | |
|---|---|
| PQT | Preliminary Qualification Test |
| QA | Quality Assurance |
| QUES | Quality Evaluation System |
| QSM | Quantitative Software Management |
| R&D | Research and Development |
| RAM | Random Access Memory |
| RFP | Request for Proposal |
| RGM | Readiness Growth Model |
| RL | Air Force Rome Laboratory |
| ROI | Return On Investment |
| SAGE | Semi-Automatic Ground Environment |
| SCM | Software Configuration Management |
| SCR | Software Change Report |
| SDD | Software Design Document |
| SDD | Software Development and Documentation, MIL-STD-498 |
| SDF | Software Development File |
| SDP | Software Development Plan |
| SDR | System Design Review |
| SED | Software Engineering Directorate |
| SED | Software Errors or Defects |
| SEE | Software Engineering Environment |
| SEES | Software Engineering Evaluation System |
| SEI | Software Engineering Institute |
| SEL | Software Engineering Laboratory |
| SEPG | Software Engineering Process Group |
| SEPO | Software Engineering Process Organization |
| SISMA | Streamlined Integrated Software Metrics Approach |
| SLIM | Software Lifecycle Management |
| SLOC | Source Lines of Code |
| SMMIS | Software Metrics Management Information System |
| SOW | Statement of Work |
| SPC | Software Productivity Consortium |
| SPO | Subsystem Project Office |
| SPR | Software Problem Report |
| SPS | Software Product Specification |
| SQA | Software Quality Assurance |
| SRP | Software Reuse Program |
| SRR | System Requirements Review |
| SRS | Software Requirements Specification |

| | |
|---|---|
| SSR | Software Specification Review |
| SSS | System/Segment Specification |
| STARS | Software Technology for Adaptable and Reliable Systems |
| STD | Software Test Description |
| STEP | Software Test and Evaluation Panel |
| STP | Software Test Plan |
| STR | Software Test Report |
| T&E | Test and Evaluation |
| TBD | To Be Defined |
| TECOM | Army Test and Evaluation Command |
| THAAD | THeater High-Altitude Defense |
| TQM | Total Quality Management |
| TR | Technical Report |
| TRR | Test Readiness Review |
| TT | Technical Test |
| U.S. | United States |
| USAF | United States Air Force |
| VCSA | Vice Chief of Staff of the Army |
| VDD | Version Description Document |
| WBS | Work Breakdown Structure |

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188



| 1. AGENCY USE ONLY (Leave blank) | 2. REPORT DATE | 3. REPORT TYPE AND DATES COVERED |
|---|---|---|
| | April 1994 | Final |

**4. TITLE AND SUBTITLE**
Survey of Software Metrics in the Department of Defense and Industry

**5. FUNDING NUMBERS**
MDA 903 89 C 0003

Task Order T-A15-742

**6. AUTHOR(S)**
Beth Springsteen, Dennis W. Fife, John F. Kramer, Reginald N. Meeson, Judy Popelas, David A. Wheeler

**7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)**
Institute for Defense Analyses (IDA)
1801 N. Beauregard St.
Alexandria, VA 22311-1772

**8. PERFORMING ORGANIZATION REPORT NUMBER**
IDA Paper P-2996

**9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)**
Office of the Director, Defense Research and Engineering (ODDRE)
Room 3D359, The Pentagon
Washington, D.C. 20301-3080

**10. SPONSORING/MONITORING AGENCY REPORT NUMBER**

**11. SUPPLEMENTARY NOTES**

**12a. DISTRIBUTION/AVAILABILITY STATEMENT**
Approved for public release, unlimited distribution: September 23, 1994.

**12b. DISTRIBUTION CODE**
2A

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

This report presents a survey of computer software measurement practices and technology in commercial and DoD organizations. Summaries are provided of 11 DoD and 14 industry programs in regard to metrics goals, metrics program implementation, metrics sets and reports, tools and repositories, and best practices. The premise of the survey is that commercial measurement practices are applicable within DoD's acquisition environment. The study concludes that industry has more mature and vigorous practices and technology than DoD organizations, largely because industry is applying metrics for clear market-driven reasons while DoD acquisition lacks comparable motivations. The results will be used to develop recommendations to improve both software measurement in weapon system acquisition and DoD software measurement research goals.

Appendix C contains the summaries of four commercial organizations requiring legal nondisclosure agreements. This appendix is published separately and is available only to government officials needing this information.

**14. SUBJECT TERMS**
Metrics, Software Measurement, DoD Software, Software Technology, Software Tools, Software Repositories,

**15. NUMBER OF PAGES**
214

**16. PRICE CODE**

| 17. SECURITY CLASSIFICATION OF REPORT | 18. SECURITY CLASSIFICATION OF THIS PAGE | 19. SECURITY CLASSIFICATION OF ABSTRACT | 20. LIMITATION OF ABSTRACT |
|---|---|---|---|
| Unclassified | Unclassified | Unclassified | SAR |