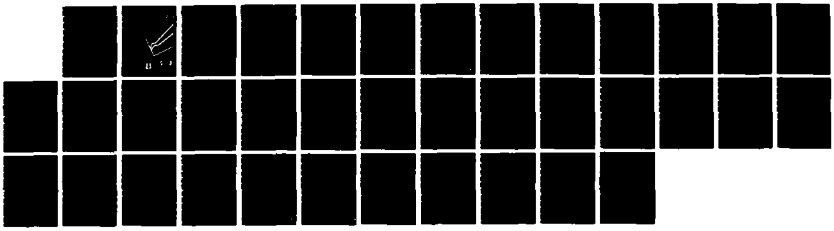
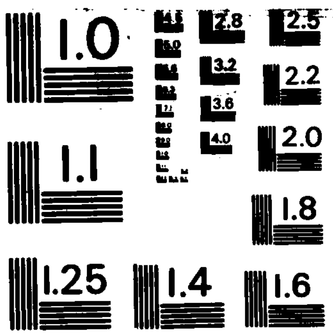
AD-A139 840   SOFTWARE COST ESTIMATION WORKSHOP REPORT(U) MITRE CORP   1/1
BEDFORD MA H SCHULTZ JAN 84 MTR-9165 ESD-TR-84-150
F19628-84-C-0001

UNCLASSIFIED   F/G 9/2   NL





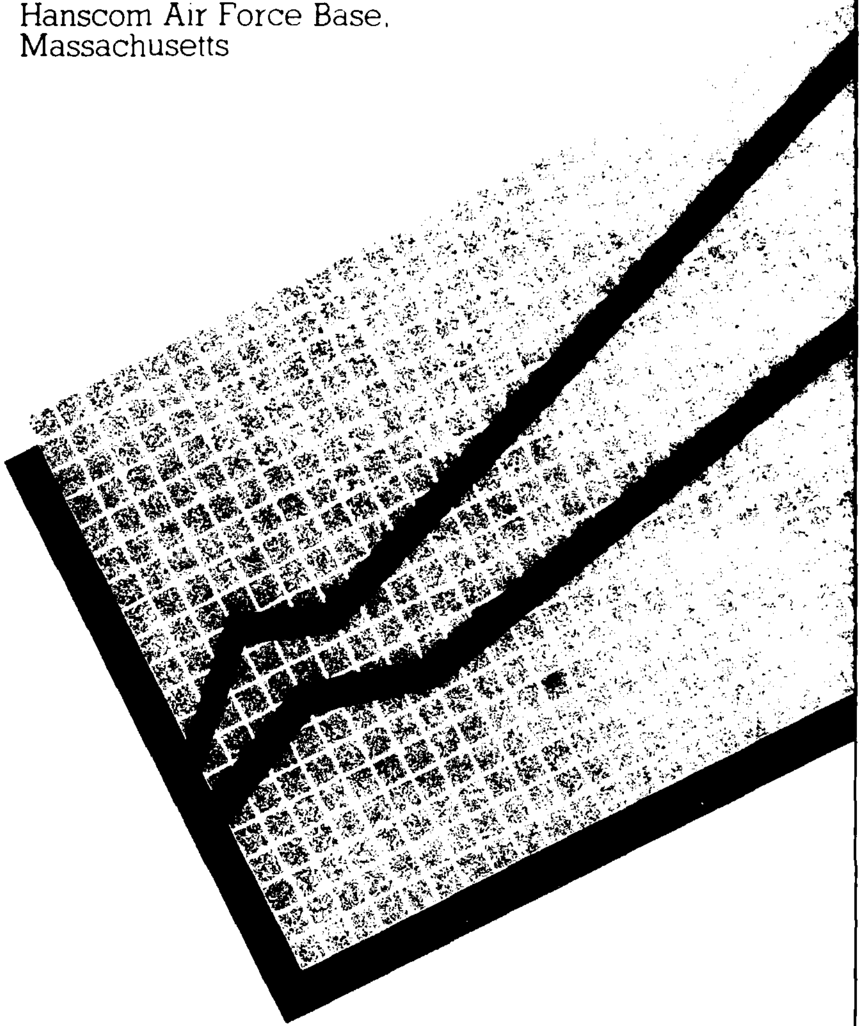
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS - 1963 - A

MTR-9165
ESD-TR-84-150

Software Cost Estimation Workshop Report

12

Prepared for
Comptroller, Cost Estimating
and Analysis Division,
Electronic Systems Division,
Air Force Systems Command,
United States Air Force,
Hanscom Air Force Base,
Massachusetts



January 1984

Approved for public release;
distribution unlimited

REVIEW AND APPROVAL

This technical report has been reviewed and is approved for publication.

[illegible]

JOSEPH P. DEAN, Capt, USAF
Project Officer

FOR THE COMMANDER

[illegible]

RONALD S. BOWEN, Lt Col, USAF
Director of Cost Analysis
Comptroller

AD-A139840

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) MTR-9165 ESD-TR-84-150	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION The MITRE Corporation 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code) Burlington Road Bedford, MA 01730	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Comptroller, Cost Estimating & Analysis Division 8b. OFFICE SYMBOL (If applicable) ACCE	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F19628-84-C-0001
8c. ADDRESS (City, State and ZIP Code) Electronic Systems Division, AFSC Hanscom AFB, MA 01731	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO.: PROJECT NO.: 6810 TASK NO.: WORK UNIT NO.:
11. TITLE (Include Security Classification) Software Cost Estimation Workshop Report	
12. PERSONAL AUTHOR(S) Herman Schultz	
13a. TYPE OF REPORT Final Report 13b. TIME COVERED FROM ______ TO ______	14. DATE OF REPORT (Yr., Mo., Day) 1984 January 15. PAGE COUNT 42
16. SUPPLEMENTARY NOTATION	
17. COSATI CODES FIELD / GROUP / SUB. GR.	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number) Cost Estimation Models Software Cost Estimation Software Data Collection

19. ABSTRACT (Continue on reverse if necessary and identify by block number)

The Software Cost Estimation (SCE) Workshop was held September 13-15, 1983 at The MITRE Corporation in Bedford, Mass., sponsored by the Electronic Systems Division of the U.S. Air Force and the Rome Air Development Center, Rome, N.Y. Government and industry experts addressed the topics: (I) Cost Effective Software Data Collection on Defense Programs; (II) Integrating SCE with Program Management; (III) Organization and Performance of SCE; and (IV) New Directions in SCE. This report contains a summary of each group's discussions and findings, together with a list of recommendations. The views expressed at the workshop and reported in this document are those of the participants and should not be interpreted as official positions of the government agencies, corporate entities, academic institutions or other organizations with which the individual participants are affiliated.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT. ☒ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Susan R. Gilbert	22b. TELEPHONE NUMBER (Include Area Code) (617) 271-8088 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

Acknowledgments

This document has been prepared by The MITRE Corporation under Project 6810, Contract No. F19628-84-C-0001. The contract is sponsored by the Electronic Systems Division, Air Force Systems Command, Hanscom Air Force Base, Massachusetts.

Table of Contents

List of Illustrations

List of Tables

PREVIOUS PAGE IS BLANK

1 Executive Summary

General

In his keynote address, Colonel David Kanter (ESD/AC) set the tone and direction for the workshop. He outlined the difficulties ESD has experienced trying to estimate software development costs. Colonel Kanter pointed out that current software cost estimation (SCE) models do not produce good estimates 3 to 5 years in advance, at the time the initial budgetary estimates are made in the Program Objectives Memorandum (POM). He also noted that cost overruns of four times the original estimates for software products with half the planned capability are not uncommon. He linked this deficiency to the inaccuracy of lines-of-code (LOC) estimates required by current models.

Colonel Kanter challenged the workshop participants to identify promising new tools and techniques to help novice and expert estimators do their jobs better. He cited, in particular, the need for a quality SCE database and asked the participants to concentrate on data collection issues. He concluded by emphasizing his intention to implement the recommendations of the workshop at ESD and carry forth the recommendations to help guide SCE research efforts of both AFSC and DOD.

The remaining opening day presentations provided a point of departure for the four working groups. The following is a list of the speakers and their subjects:

Dr. Barry W. Boehm (TRW)
"Research Issues in SCE"

Robert Thibodeau (GRC)
"SCE Techniques vs. Life Cycle Phases"

Al Kopania (Aerospace)
"Software Size Estimating Model"

Maj. Joseph Duquette (HQ AFSC)
"Problems, Experiences . . ."

John Gaffney (IBM)
"Software Function Estimates"

Merle McKenzie (JPL)
"Software Life Cycle Model"

Carolyn Wong (SDC)
"Technical Estimates of Cost Model"

Alan J. Roberts (MITRE)
"Making Costing Count"

The speakers corroborated Colonel Kanter's observations, shared their own experiences, and described the directions of SCE research within their organizations.

Findings and Recommendations of the Working Groups

Group I · Leader, Ronald B. Leask

The problem area assigned to Group I was "Cost Effective Data Collection on Defense Programs." The primary recommendation of Group I is that SCE data collection be standardized for all defense organizations/agencies in the form of a Military Standard (MIL-STD) for software Work Breakdown Structures (WBSs) and a data item description (DID) for reporting attributes of software projects that drive development cost. Furthermore, a common set of automated tools should be developed to help contractors collect and report the data and provide government agencies with data base management capabilities.

The benefits of standardizing data collection across DOD, according to Group I, include commonality of data definitions to support SCE research, reduced costs, compatible data base structures, and elimination of the need for contractors to tailor their management systems to fit individual contracts.

The group reviewed the Software Acquisition Resource Expenditure (SARE) Data Collection Methodology and found it to be a suitable starting point for the DOD standardization effort. The SARE methodology, developed by the MITRE Corporation for ESD, includes a draft MIL-STD for software WBSs and an associated DID.

Because the benefits of standard data collection will be realized across DOD, Group I recommends that funding be provided at the DOD level to disseminate the SARE documents for widespread industry/government review and finalization, and to develop automated tools to collect, report, and manage the data. A joint DOD/industry working group should be formed to coordinate these efforts.

PREVIOUS PAGE IS BLANK

Group II · Leader, Dr. Barry W. Boehm

Group II addressed the problems of "Integrating SCE with Program Management." Citing a serious lack of traceability between initial POM budgetary estimates, proposal estimates, development budgets, and "cost-to-complete" estimates, the group pointed out that this lack implies there is no accountability for poorly conceived estimates.

The group recommends that estimators at all stages of the system life cycle be required to provide a set of common inputs for an appropriate set of SCE models as part of the justification for their estimates. The assumptions and constraints embedded in earlier estimates (e.g., POM estimates) should be made available to subsequent estimators (e.g., bidders). A standard format for disclosing the SCE methodology used by bidders and the supporting rationale should be included in the Instructions for Bidders in the Request for Proposal (RFP) package. This would allow qualified personnel to evaluate the cost realism of alternative proposals.

Also noted were the potential benefits of integrating SCE with the risk management function. The group recommends that SCE models be developed with the capability of identifying and tracking areas of software development risk.

As a general observation, Group II recognizes the need for continued research to develop more accurate models. The group cited data collection as a key to improving SCE models. Government and industry must cooperate in developing a standard definition of the software life cycle, a standard software WBS, and an SCE DID. They concurred with Group I that the SARE methodology should be the starting point for the standardization effort and a joint task force of industry and government representatives must coordinate the documents to ensure that the data collected is non-threatening and useful to program managers.

Group III · Leader, Robert C. Thibodeau

Group III was concerned with "Organization and Performance of SCE." Noting that SCE for mission critical software is hampered by a lack of organizational focus, the group recommended that SCE be given greater status by recognizing it as a separate entity worthy of its own position in the organization, that separate facilities be provided, and training and career paths be developed to attract and maintain the quality of specialists needed.

The group noted the significant proportion of system costs attributable to software and that software costs tend to be incurred earlier in the acquisition cycle than hardware costs. Recommendations were made for achieving earlier and better software cost visibility by generating SCEs periodically during the concept phase, for every major change in software requirements, and for each formal review milestone. More frequent SCEs would also be helpful for early identification of software risk areas.

The concept of an independent SCE and the use of an overseer agency to direct, compare, and audit both the SCE prepared by the project and the one performed independently, is recommended as a structure to achieve better baseline SCEs. The key to the success of this structure is the overseer's ability to obtain truly independent cost estimates.

The group addressed the problem of obtaining accurate SCEs early in the system life cycle. The concept of a truly independent SCE, as just mentioned, was discussed, but the difficulty in developing sizing parameters was cited as a major shortcoming. However, the group noted that current research in equating functions to lines-of-code, and the use of new software development techniques such as reuseable code, prototyping, incremental development, high-level languages, and application generators, will greatly reduce the number of unknowns and result in more accurate early SCEs.

Group IV · Leader, Dr. Randall W. Jensen

The fourth working group discussed "New Directions for Software Cost Estimation (SCE)." Topics addressed included alternate approaches to SCE, the impact of emerging technologies, alternatives to Source Lines of Code (SLOC) as the fundamental measure of software development, and approaches to SCE for software upgrades.

The group noted that current SCE models are static, i.e., incorporate time-invariant relationships. A proposed alternate approach is to develope dynamic models of the software development process, using simulation models or the mathematics of control theory. Feedback can be dealt with directly in simulation models, but it has not yet found its way into parametric cost modeling. Strategies for incorporation of feedback were explored and are discussed in this report.

The group discussed new and emerging technologies for software developments. Some have a major impact on SCE, dictating new estimating methodologies, e.g., artificial intelligence and distributed development workstations. Other technologies may only require new parameter values for existing models. These include Ada programming and its support environment, programming via application generators and very high-level languages (VHLs), the use of program design languages (PDLs), the programming of distributed processing systems, and firmware technology. But the development of new parameter values requires the collection of additional data, and the group recommends that this begin as soon as possible.

There is a need to quantify software as early as possible in the acquisition cycle. Most often this is done by using Source Lines of Code (SLOC). The group addressed the need to find alternatives to SLOC and recommended that information available at the time, such as requirements or functionality, be examined. The use of quantitative requirements produced by such tools as PSL/PSA, SADT, and data flow diagrams, and quantifying the relationship between functionality and SLOC were two suggested areas of investigation.

SCE for software upgrades was recognized as a major problem area due to many unique factors. It was felt that many of these could become model parameters and be translated into SLOC if sufficient data were available. For example, little data is available regarding the effect on the system architecture of new system requirements and the use of off-the-shelf software packages. Also, little data is available on the extent of the testing required for development and integration of software upgrades. Recommendations were made to collect experiential data to identify and quantify key upgrade cost factors, to modify existing models by defining new parameters, and to develop new models specifically for software upgrade projects.

List of Recommendations

The following list was extracted from the reports of the working groups. For convenience in referencing the discussions of and supporting reasoning for the recommendations, they are numbered identically here and in the body of the report.

Group I

1.1 Establish a uniform method of SCE data collection to be used by DOD and industry on all DOD programs.

1.2 Develop or modify a MIL-STD for a software WBS and establish a standard DID for collecting uniform SCE data.

1.3 Develop automated tools to standardize the management of SCE data bases and the data collection process.

1.4 Have the procuring agency validate SCE data when it is reported and before it is put into the data base.

1.5 Use the Software Acquisition Resource Expenditure (SARE) Data Collection Methodology as a tentative framework to begin the SCE data standardization effort.

1.6 Establish DOD funding for evaluation and enhancement of the SARE MIL-STD and DID approach to SCE data collection.

1.7 Collect SCE data at the Computer Program Configuration Item (CPCI) level using the SARE proposed MIL-STD for software WBSs.

1.8 Establish a subset of SCE data, using the SARE methodology as a basis, within 90 days via a tri-service working group, and begin near-term data collection.

Group II

2.1 Establish a program to enhance traceability between successive SCEs.

2.2 Obtain SCEs from a variety of sources, e.g., independent costing office, line organization, different cost models, different estimation techniques. Pass any assumptions and constraints embedded in earlier estimates (e.g. POM) on to subsequent estimators (e.g. bidders).

2.3 To reduce the cost of low-level cost and progress reporting, develop an automated planning and reporting system that is flexible and matched to management needs. This automated system must provide traceability back to SCE model results.

2.4 Develop and require a standard format for disclosing the software costing methodology and rationale used by the contractor.

2.5 Develop software cost models with the capability of identifying and tracking software development risk.

2.6 Require more realistic budgetary estimates, along with their basis-of-estimate justification. Validate budgetary estimates by analogy and other accepted estimating methods before use in POMs.

2.7 Apply adequate R&D resources to determine the earliest time at which cost estimation techniques can replace seat-of-the-pants estimates and to identify and quantify the significant cost estimators that will be available at that time.

2.8 Continue to fund research for the development and validation of SCE models. Government and industry must cooperate in the requesting, gathering, analysis, and reporting of SCE data. This includes an agreement of what cost data should be gathered and how the information should be stored and used.

2.9 Establish a joint industry-government task force to propose a set of standardized definitions for the cost-estimation area and seek a public consensus on those definitions. Once agreement has been reached, the relationships between models whose parameters have been redefined to the consensus view should be examined.

2.10 Develop SCE models responsive to the need for project managers to make trade-off decisions.

Group III

3.1 Give responsibility for software cost estimation to a separate group of personnel.

3.2 Minimum staffing for a software cost estimation group is a software engineer, a cost-estimating specialist, and a third specialist for data collection, data-base management, and literature research.

3.3 Establish a career path in software management within DOD.

3.4 Have program managers treat software at the same level of detail as hardware.

3.5 Estimate software development costs periodically during the concept phase as well as at SRR, SDR, PDR, and CDR.

3.6 Establish a separate DOD regulation calling for review or audit of software progress at various milestones, thereby making the software "visible" to the program office.

3.7 Prepare a new cost estimate for any significant change in software requirements.

3.8 Provide a software Work Breakdown Structure (WBS) and a work schedule to the cost estimator.

3.9 Anticipate the activities required for CM, QA, and IV&V in detail and include their costs in the SCE.

3.10 Establish guidelines for the performance of Independent SCEs (ISCEs), paying particular attention to ensure that the ISCE is truly independent, impartial, and fair, and, at the same time, relevant to the project.

3.11 The Baseline SCE (BSCE) and ISCE shall each contain a basic estimate as well as alternative estimates based on variations in assumptions or design methodologies.

3.12 Establish guidelines for the Audit process to ensure as accurate a BSCE as possible.

3.13 Establish a BSCE for each definable phase in the life cycle of a project.

3.14 Validate SCE models and, if necessary, recalibrate them, as a project progresses and more data becomes available.

3.15 Establish guidelines for each SCE organization to ensure choosing SCE models that are accurate and appropriate.

3.16 Encourage and continue research to formulate methodology for SCE as early in the life cycle of software as possible.

Group IV

4.1 Investigate control-theory or simulation-modeling techniques for modeling the software-development process as a system in order to determine if a dynamic (time-varying) model can be applied to SCE.

4.2 Determine if the number of data items and the number of logical inference rules can be used as a replacement for source lines of code in SCE for artificial intelligence systems.

4.3 Collect further data for emerging technologies in order to develop new values for existing model parameters.

4.4 Use quantitative measures of requirements, derived from PSL/PSA, SADT, data flow diagrams, or other tools, as input to the SCE process.

4.5 Establish relationships between the functions identified during functional decomposition of systems and SLOC.

4.6 Gather and analyze applicable experience data and case studies for software upgrades to see what the key factors were that affected resources and schedules.

4.7 Modify existing models for use in upgrade projects by defining new model parameters (or recombining or rescaling existing model parameters) to account for key upgrade factors.

4.8 Develop new models specifically tailored to software upgrade projects.

2 Background and Organization of the Workshop

The Software Cost Estimation Workshop of 13-15 September 1983 was sponsored by the Electronic Systems Division of the U.S. Air Force and by the Rome Air Development Center, an affiliated laboratory. It was held at The MITRE Corporation in Bedford, Massachusetts. The schedule for the workshop is provided in appendix A. Appendix B contains a list of the Workshop Committee. The participants in the workshop are listed in appendix C, and the members of each of the working groups are listed in appendix D.

The workshop was a three-day program during which government and industry representatives exchanged ideas and experiences, and formulated near-term and long-term recommendations aimed at improvement of software cost estimations for mission-critical software.

Most of the first day was dedicated to presentations and discussions on the need for software cost estimation and the state of the art in its practice. After this, the participants were organized into four working groups, each addressed to one of the following topics:

I. Cost Effective Software Data Collection on Defense Programs

II. Integrating Software Cost Estimation with Program Management

III. Organization and Performance of Software Cost Estimation

IV. New Directions in Software Cost Estimation

A more detailed listing of the subjects suggested to the working groups is given in appendix E.

Each working group was asked to provide a written summary report of its findings, conclusions and recommendations. These separate reports, and the general findings, conclusions and recommendations of the workshop as a whole, form the major content of this document. An executive summary is provided in section 1. The individual group reports appear in sections 3 through 6.

3 Working Group I

Cost Effective Software Data Collection on Defense Programs

Approach

The objectives of Group I were to:

- Recommend a software work breakdown structure (WBS) for consistent cost data collection on defense programs.
- Determine appropriate data collection levels and frequency.
- Consider the implications of institutionalizing data collection in military standards (MIL-STDs) and associated data item descriptions (DIDs).
- Assess the cost/benefit ratio of creating and maintaining an SCE data base.

The group initially observed that the type and level of detail of data collected in the SCE data base must support the needs of a varied group of SCE users and that the amount of data detail needed by the various users may be in conflict. For example, a project manager may need high-level cost and schedule estimates to support conceptual phase activities 3 to 5 years *before* the start of system development, whereas a project manager evaluating proposals during a source selection may need detailed cost breakdowns of activities and schedules to determine the feasibility of alternative approaches.

A draft military standard for software WBSs and an associated DID to collect technical characteristics of software projects were made available and were used as a basis of discussion by the group. The group proposed to accomplish the following tasks:

- Identify the likely users of an SCE data base.
- Define and characterize the types of SCE data that should be collected.
- Identify how and when the SCE data will be utilized by each user during the life cycle of a software system.
- Assess the implications of establishing a MIL-STD for software WBSs and a standard DID for collecting characteristics of software projects.
- Assess the implications of protecting and disseminating proprietary and potentially classified information.
- Assess the suitability of the Software Acquisition Resource Expenditure (SARE) Data Collection Methodology as a tentative framework for common SCE data collection on defense programs.

Summary of General Discussions

It was found necessary to establish a common perspective of the various SCE users and their needs before focusing on the specific data collection issues. This perspective is summarized in table 1, which characterizes, functionally, the expected users of an SCE data base and the uses they would

Table 1

Use for the SCE data / Functional user of the SCE	Early high-level cost schedule estimating	Inputs to cost models Detail cost, schedule, risk	Monitor/ assess on-going dev. cost, schedule, risk	Evaluate design options	Provide life-cycle cost	Future planning	Evaluate SW dev. methods	Use for operation & maintenance
Program/Project Manager	•		Summary		•	•		•
Software Monitor			•	•				
Software Developer	•	•		•				
Maintainer								
End User						•		•
Research & Technical Evaluator		•				•	•	
Cost Estimator	•	•			•	•		•

have for the data. The functions listed are not necessarily mutually exclusive. (E.g., a cost estimator can reside within a project manager's office or in an independent organization.)

An "X" in the table indicates a primary use for the data. For example, a project manager may use the SCE data base for high-level cost and schedule estimates in the conceptual phase of the system life cycle. (The term "program manager" also includes any planning organization involved with the project before a program office has been formed.) In the development phase, the program manager would use data reported by contractors to measure cost/schedule performance and evaluate risk. If this data were reported against a standard software WBS, it would contribute to a consistent data base and allow the program manager to make comparisons against actual data from past programs. The project manager might also use summary data from the software monitors and cost estimators.

Table 2 shows who, when, and how often the data base would be used. For example, the software development organization could use it to prepare cost and schedule estimates that would be updated at program milestones. Similar information would be used by the DOD software development monitors to evaluate the realism of proposals and later to measure contractor's progress. The data base would also allow the monitor to independently calculate the schedule and cost of the project.

Table 2

SCE User	When SCE Is Used in Life Cycle	Frequency
Program Management	Concept (planning), proposal, development, operation, maintenance	Monthly
Software Monitor Software Developer	Proposal phase, development (SRR, PDR, CDR, PQT/FQT, IOC)	Major Milestones
Maintainer	IOC to termination (ECPs)	For Each ECP
Research/ Technical Evaluation	All data	As required

Although several software data bases exist today, they are useless to most SCE users and researchers because they lack homogeneity or are proprietary. Even though data needs vary among DOD organizations and among users within each organization, a single, uniform approach to data collection would allow data from all organizations to be combined for supporting research that will benefit all. In addition, a common set of tools to collect, report, manage, and analyze the data could be provided to contractors and DOD agencies. Uniform methodology would also reduce the cost of data collection. A prerequisite to this is the use of common software terminology on all DOD programs and uniform application of existing MIL-STDs, such as MIL-STD-483, MIL-STD-490, MIL-STD-1521A, and the Joint Logistics Command's new MIL-STD-SDS.

Recommendation 1.1 Establish a uniform method of SCE data collection to be used by DOD and industry on all DOD programs.

MIL-STD-881A, "Work Breakdown Structures for Defense Materiel Items," the governing authority for WBSs, is presently deficient with respect to software. MIL-STD-881A must be revised, or a new MIL-STD created to clearly define work elements and address aspects of the system acquisition process that are peculiar to software development. A DID is also needed to define technical characteristics of the software that drive development cost. The only alternative would be to have government monitors attempt to extract the needed information using existing DIDs. This would, in addition to being error prone and yielding second-hand, inconsistent information, waste considerable time. It would be more cost-effective to specify precisely the information needed in a DID and collect it directly from the software developers.

Recommendation 1.2 Develop or modify a MIL-STD for a software WBS and establish a standard DID for collecting uniform SCE data.

Each DOD organization/agency responsible for system acquisition needs to maintain an SCE data base for its own programs. However, the data bases should be compatible, so that they can be combined to support SCE research across DOD.

The concept of standard SCE data collection should be extended to include a common set of automated tools provided to DOD organizations to manage their data bases. To make SCE data collection and management as painless as possible for defense contractors, they should be provided with a similar set of tools. DOD should fund the development of such tools, and contractors should report the data through an interface with the cognizant DOD agency's data base management system. This will reduce cost and the potential for error.

Recommendation 1.3 Develop automated tools to standardize the management of SCE data bases and the collection process.

Recommendation 1.4 Have the procuring agency validate SCE data when it is reported and before it is put into the data base. Only the procuring agency has the knowledge necessary to verify the reported data.

Cursory review of the SARE documents indicates that they provide a sufficient basis to begin the DOD standardization effort. The SARE report should be circulated among DOD agencies, defense industry associations, and other interested organizations and individuals for review and comment.

Recommendation 1.5 Use the Software Acquisition Resource Expenditure (SARE) Data Collection Methodology as a tentative framework to begin the SCE data standardization effort. The methodology is documented in ESD-TR-83-214, AD A137 084, "Final Report: Software Acquisition Resource Expenditure (SARE) Data Collection Methodology," prepared by The MITRE Corporation under contract to AFSC Electronic Systems Division (ESD) and dated December 1983. The report includes a proposed MIL-STD for software WBSs and a DID that contains five forms for collecting technical characteristics of the software and software development environment. The DID includes the major parameters used by the currently popular SCE models.

Although the SARE methodology is still under development and needs exposure and trial use before it can approach its final form, it would be advantageous to have DOD provide funding to support its evaluation and refinement. This includes funds to consolidate comments from reviewers, revise the documents, and support trial applications. DOD should also provide funds to develop a common set of automated tools needed to collect, report, and manage the SCE data. Further, DOD should form a work group of comptroller and SCE representatives from all services and agencies to coordinate the standard data collection methodology and plan future enhancements.

Recommendation 1.6 Establish DOD funding for evaluation and enhancement of the SARE MIL-STD and DID approach to SCE data collection.

CPCIs are the primary software products delivered on defense programs. They form the basis for functional and performance allocations, interface control, detailed specification and design, development, testing, and configuration management. As such, they provide a solid definition of the software to be produced and are at an appropriate level of detail to support software cost estimation.

The SARE MIL-STD presently extends the software WBS below the CPCI level, down to the Computer Program Component (CPC) level. The SARE WBS elements below the CPCI level should be provided to contractors as models only. The contractor should be allowed to use any specific breakdown it wishes below the CPCIs, provided the activities in aggregate correspond to the combined SARE WBS elements. This will provide the contractor with a model of an acceptable WBS it can use to fashion its own low-level WBS elements, thereby assuring uniform cost reporting without restricting the contractor to a single, rigid structure.

When CPCIs are extremely large—in excess of 50K lines of code—reporting at the CPC level, rather than the CPCI level, may be appropriate. In addition, collection of a limited number of CPC characteristics, primarily size and function, would be useful to support software size estimation. The SARE DID presently contains such a data collection form.

Recommendation 1.7 Collect SCE data at the CPCI level using the SARE proposed MIL-STD for software WBSs.

Creation of a MIL-STD and DID will require a long time. In order to meet near-term SCE data needs, a tri-service working group should be convened within 90 days. Specifically, the group should review the proposed SARE DID and determine a useful subset of data items that can be uniformly collected on on-going and recently completed programs at little or no additional contract cost. It is recommended that data collection begin by the second half of FY84.

Recommendation 1.8 Establish a subset of SCE data, using the SARE methodology as a basis, within 90 days via a tri-service working group, and begin near-term data collection.

Implications and Considerations

Group I recognizes that implementation of its recommendations will not be easy or convenient. As a first hurdle, clear and unambiguous definitions of every item in the WBS are needed to insure consistent interpretation of collected SCE data. Second, since the relevant data bases will contain classified and/or proprietary information, steps must be taken to restrict access to qualified personnel. Such restrictions can, in and of themselves, make the program unwieldy, perhaps requiring that classified material be handled in a separate data base. Additionally, where contractor proprietary information is involved, it may be necessary to release only summary level information until a sufficient number of projects are in the data base that no individual project can be identified by its characteristics. Another potential problem is that once a MIL-STD and DID are instituted, it may not be possible to update them rapidly enough to keep pace with technological developments. Nonetheless, it is the consensus of the group, that the above recommendations will sufficiently enhance SCE data collection and management to justify their implementation.

4 Working Group II

Integrating SCE With Program Management

Introduction

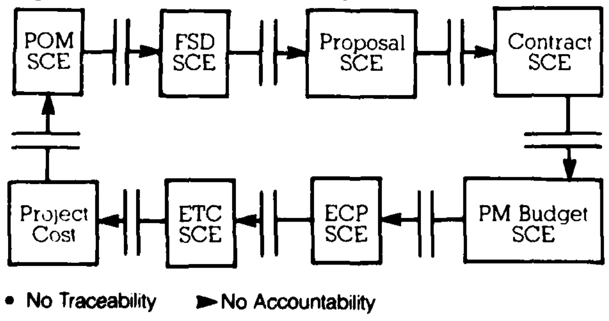
Group II's primary conclusion was that SCE is generally not well integrated with program management (PM) in USAF software acquisitions. This is a major source of USAF's critical problems with software project predictability and control. In reaching this conclusion, the group noted three principal areas of concern: traceability, risk management, and SCE technology. For many acquisitions there is a serious lack of traceability between initial SCEs, proposal SCEs, program development budgets, and cost-to-complete estimates. Moreover, both SCE and PM are compromised in effectiveness by insufficient attention to software risk management. The current state-of-the-art in SCE technology can help PM appreciably, but further research and development is needed to make SCE models and techniques more responsive to PM needs.

Cost Traceability and Cost Baseline Management

As explained below and illustrated in figure 1, the major difficulties in USAF software project predictability and control arise from the lack of traceability between successive estimates. Many software cost estimates are developed throughout the life of a project. They support such activities as:

- Program Office Memorandum (POM) Budgeting
- Full-Scale Development (FSD) Budgeting
- Proposal Cost Estimation
- Contract Cost Negotiations
- Project Management Budgeting
- Engineering Change Proposal (ECP) Budgeting
- Estimate-to-Complete (ETC) Assessments

Figure 1 The SCE Traceability Problem



Without traceability, there is no accountability. Because nobody will be able to relate future results back to the current software cost estimate, it becomes too easy to arbitrarily adjust software estimates to meet available budgets or competitive pressures.

The government makes very early estimates for the Program Objective Memorandum (POM) in order to get funding approved by Congress. The personnel submitting the POM are under pressure to provide an estimate that will ensure project approval. Experience to date has shown POM estimates to be consistently low. The assumptions and constraints within which the estimates are made are not made available either to the personnel developing the system specification or to the contractor personnel proposing against the system specification. This provides the opportunity for wide and varied interpretations of the original scope of the project.

Recommendation 2.1 Establish a program to enhance traceability between successive SCEs. Elements of this program would:

- Require estimators at all stages to provide a set of common estimation inputs that can be used to drive an appropriate set of software cost estimation models.
- Require revised inputs and estimates to be prepared at major project milestones or any time there is a change in budget, schedule, or product scope.
- Establish and use a software Work Breakdown Structure (WBS) to map estimates into program management budgets.
- Track expenditures and progress vs. budgets and plans via an earned value system or the equivalent.
- Minimize the instabilities of the traceability process via incremental development. (There should be a high threshold on accepting proposed changes that complicate cost/progress traceability within a development increment. Most changes should be deferred to a subsequent increment.)

This program should first be tested on a relatively small and short acquisition. After refinement, it could be applied to a sample of larger projects. Finally, refined yet further, it could be applied to all USAF software aquisitions.

Recommendation 2.2 Obtain SCEs from a variety of sources, e.g., independent costing office, line organization, different cost models, different estimation techniques. Pass on any assumptions and constraints embedded in earlier estimates (e.g. POM) to subsequent estimators (e.g. bidders).

Recommendation 2.3 To reduce the cost of low-level cost and progress reporting, develop an automated planning and reporting system that is flexible and matched to management needs. This automated system must provide traceability back to SCE model results.

Risk Management

Risk management uses planning, analysis, engineering, and program management to identify and eliminate sources of cost, schedule, or performance risk in the development and support of software. Because of the importance of software risk management, a single focal point should be established in each Program Management Office (PMO) to be responsible for this activity. A risk management plan should begin with the identification of high risk elements. The plan should provide for the tracking of these risks and for the development of alternate plans for responding to cost overruns (actual or anticipated), schedule slips, and shortfalls in performance. A key data input to the PMO software risk manager is the SCE. The risk management plan should contain the baseline SCE with risk assignment, i.e., most probable cost and maximum cost. SCEs for the risk management alternatives should be developed as an aid to program management in making redirection decisions. An updated SCE should be prepared at all key program decision points. The SCE model should be used to determine how sensitive the software development and support costs are to program changes. This information can be used as an aid in selecting alternatives should redirection be required.

The SCE staff should participate in the risk identification by coordinating with system and software design engineering organizations. The SCE staff should participate in design reviews and ad hoc risk reviews at the prime and subcontractor's facilities. It should be the task of the SCE staff to answer all questions concerning the effects of changes in manpower, approach, skill levels, capital resources, schedule, subcontractors, use of management reserves, etc., that might be dictated by alternate strategies.

The SCE staff should closely support the PMO in tracking the implementation of corrective actions and, as data becomes available, maintain a current estimate of cost at completion.

The current source selection process often forces the government to select from unrealistic cost proposals. The normal process is for two teams, a technical evaluation team (TET) and a cost evaluation team (CET), to accept or reject their respective parts of a proposal. Since the TET has no access to the cost data and the CET is not qualified to evaluate software development cost from a technical point of view, the CET is forced to rate proposals on a lowest-bid basis. With individual companies all trying to "low-ball" a price for a program, an artificial downward pressure is placed on all competitive price estimates.

Recommendation 2.4 Develop and require a standard format for disclosing the software costing methodology and rationale used by the contractor. This standard format would become part of the "Instructions to Bidders" in the RFP and the software costing methodology and rationale would become part of the technical information disclosed. Such disclosure would allow qualified personnel to evaluate the cost realism of the proposal. Better techniques for balancing costs and benefits in source selection should be developed and used.

Current software cost models are not responsive to the needs of management in executing risk analysis.

Recommendation 2.5 Develop software cost models with the capability of identifying and tracking software development risk. Ranges in estimated software size and other cost drivers, hardware/software trade-offs, and varied software development approaches such as structured design, walk-throughs, prototyping, independent verification and validation, etc., should be accommodated in the SCE model.

Cost overruns result when software cost commitments are established without a good understanding of system requirements. Competitive concept definition contracts and software prototyping are means of defining the requirements and deferring full development cost commitments until the nature of the desired end product is well understood.
Recommendation 2.6 Require more realistic budgetary estimates along with their basis-of-estimate justification. Validate budgetary estimates by analogy and other accepted estimating methods before use in POMs.

Summary

Integrating the SCE function into the risk management process through the identification, quantification, and tracking of risk and of risk resolution effectiveness should lead to an improved software product on schedule and at minimum cost. R&D investment to improve the capability of the SCE models to track and quantify risk is encouraged.

The risk management policy recommended by the recent USAF/SAB Monterey Summer Study on Software Acquisition provides a sound framework for implementing these recommendations. It is further endorsed by this Working Group.

SCE Technology

Actual use of an SCE model by program managers depends on the degree to which they trust the model and the degree to which it satisfies their information requirements. To this end, a number of significant improvements in SCE technology are needed. Greater accuracy must be achieved, and earlier applicability in the life cycle is needed. In addition, better communication must be established, both among the different models and between DOD and industry. Finally, the models should support analyses aimed at projecting the impact of contemplated changes in program goals or the means used to implement them.

Initial cost estimates may err by as high a factor as 16. From so shaky a beginning, proper program planning and control is difficult to establish. Software cost estimation models should be applicable far earlier in the life cycle than they are at present and should limit the range of error to perhaps 3. One useful method might be to reprogram existing models to accept the cost of a project and produce a set of parameters that could be optimized in different ways for the same cost. Examination of the parameters might lead to a determination of whether or not the target cost can be realized.
Recommendation 2.7 Apply adequate R&D resources to determine the earliest time at which cost estimation techniques can replace seat-of-the-pants estimates and to identify and quantify the significant cost estimators that will be available at that time. Since some of the most effective cost drivers are very difficult to estimate early (e.g. source lines of code), new estimators, such as application type, software quality requirement, or function points, that can be used in an early phase need to be considered.

Even when SCE models are used after sufficient data is obtained, the manager's confidence in the accuracy of the results is often low. Unfortunately, there is some justification for this attitude. Current models show different effects for factors that should remain constant across contractor or application boundaries. One extreme example is the different effect on project effort that several models predict for a schedule expansion. While the Deep Space Network (DSN), Jensen, and SLIM models predict that project effort will decrease indefinitely with a schedule expansion, the COCOMO and Price-S models predict an eventual increase in project effort. Until models reach a consensus concerning gross project behavior, their results will continue to be treated with skepticism.
Recommendation 2.8 Continue to fund research for the development and validation of SCE models. Government and industry must cooperate in the requesting, gathering, analysis, and reporting of SCE data. This includes an agreement of what cost data should be gathered and how the information should be stored and used. For example, collected data must not divulge a con-

tractor's proprietary information, yet it must provide enough information to let government agencies justify the contractor's cost. Collected data must also provide information to a central repository — for example, the Data Analysis Center for Software (DACS) at RADC — for aggregation into a software cost data base for nationwide use.

The availability of several different SCE models is generally perceived as both necessary and beneficial since it permits the selection of a model on the basis of application domain or project phase. However, to realize these advantages, common and consistent definitions of model parameters, terms, and outputs must be achieved. That is, there must be common understanding of lines of code, man-months, development phase, complexity, and the like.

Recommendation 2.9 Establish a joint industry-government task force to propose a set of standardized definitions for the cost-estimation area and seek a public consensus on those definitions. Once agreement has been reached, the relationships between models whose parameters have been redefined to the consensus view should be examined. Definitions of SCE model inputs and outputs, life cycle phases, cost drivers, and Work Breakdown Structure elements should be covered.

Recommendation 2.10 Develop SCE models responsive to the need for project managers to make trade-off decisions. Models should be able to trade off factors such as:

- Incremental vs. one-time development
- Prototyping vs. MIL-STD specification
- Redevelopment vs. reuse of previously developed software
- Nominal schedule vs. compressed schedule vs. extended schedule
- Capability (reliability, function, maintainability) vs. cost
- Methodology (unit test vs. code inspection, DeMarco vs. Yourdon design techniques, SREM vs. MIL-STD B5 specifications, etc.)

Further, SCE models should provide such program management needs as estimates of effort distribution by phase, activity, and labor grade. They need to be accurate within the program's domain of applicability, and easy to calibrate or tailor to particular application domains.

5 Working Group III

Organization and Performance of Software Cost Estimation (SCE)

Summary

Recommendations related to the organization and performance of the SCE function are presented in three subsections:

- Organizational Structure and Personnel Requirements
- The Acquisition Process and Project Data Requirements for SCE
- Tools and Techniques

Software cost estimation for mission-critical systems is hampered by a lack of organizational focus. Software costing is often delegated to a group that does many kinds of cost estimating. The beginning point for better SCE is to give it more status by recognizing it as a separate technical discipline worthy of investment in facilities and training, and of a permanent place in the organization. This will help to attract the talent that SCE requires. It also creates an identifiable resource for other elements in the organization and establishes a repository for accumulated knowledge in the field, thus providing a foundation for future improvement.

Current practice has the system developer make either routine or special requests for SCE without specifying precisely what is needed, often ignoring possible incompatibility of the request with the information needed to provide it. Possibly, additional information could be given to the estimator by the developer if the need were known. It is, therefore, incumbent on the SCE organization to obtain from the system developer a precise statement of needed support and, in turn, to specify to the system developer the data that must be provided to accomplish this support.

Consideration of tools and techniques for SCE must be preceded by an understanding of the application. Three basic functions that impose requirements on the tools are: Independent Cost Estimates, Audits, and Baseline Cost Estimates. These functions have different requirements for cost information and are usually performed by different groups operating in different environments with different estimating data at their disposal.

Since a large selection of adequate tools exists for making SCEs, the problem is generally that of choosing one appropriate for a given situation. Tool selection depends on a number of considerations and requires that the estimator know precisely what information is needed and to what order of accuracy. The dependence of SCEs on application and environmental factors requires that a model be calibrated to a given estimating situation and that its performance be validated for its intended use.

Organizational Structure and Personnel Requirements

Creating a Separate Identity

Software costs, once treated as incidental, are becoming a very significant, if not overriding, concern in the cost estimating community. Within Air Force Systems Command (AFSC), for example, software comprises 55-70% of the aquisition costs of many weapon systems. To gain control over software costs, management must be willing to dedicate resources to software cost estimating.

Recommendation 3.1 Give responsibility for software cost estimation to a separate group of personnel.

This group can be a part of the existing cost estimation organization or exist within the program management organization itself; but it must be dedicated exclusively to software cost estimating. Only specialists, dedicated to the discipline and allowed time to pursue it can hope to keep up with the great mass of software research currently being conducted. Data must be collected and maintained in order to calibrate, validate, and update existing estimating tools (Price-S, Jensen, COCOMO, etc.), as well as to develop new ones.

To function effectively, a group of this type must be able to analyze, evaluate, and develop sizing of software efforts; must have familarity with existing estimation tools and the results they deliver; must be able to collect and maintain a software data base and perform literature research. Necessarily then, it will draw on multiple disciplines.

A multi-disciplinary software cost estimating group will also be particularly valuable in providing management recommendations. Having detailed knowledge of a software effort and the proper exercise of estimating tools, the group will be able to recommend trade-offs and cost control measures.

In addition, it could supply expertise in the areas of contracting (i.e. contract types based on risks involved) and program management.

Recommendation 3.2 Minimum staffing for a software cost estimation group is a software engineer, a cost-estimating specialist, and a third specialist for data collection, data-base management, and literature research.

Career Path

One of the challenges facing the DOD cost-analysis community is development of skilled personnel in software engineering. Current personnel assignment policies are not responsive to this end. Program managers who successfully acquire SCE personnel do so on the basis of personal familiarity with well-trained, exceptional individuals.

Recommendation 3.3 Establish a career path in software management within DOD.

The Acquisition Process and Project Data Requirements for SCE

The Acquisition Process and Its Impact on SCE

The existing procurement process for major systems treats software at a lower level of detail than hardware. The inherent difference in the time phasing of hardware and software expenditures along with the emergence of software as a dominant cost item require that this practice be changed. For example, software go/no-go decisions are made at various times during or after PDR. Up to 50% of the software project dollars may have been spent by this time, compared with 10-20% for hardware.

Recommendation 3.4 Have program managers treat software at the same level of detail as hardware.

Recommendation 3.5 Estimate software development costs periodically during the concept phase as well as at SRR, SDR, PDR, and CDR.

Recommendation 3.6 Establish a separate DOD regulation calling for review or audit of software progress at various milestones, thereby making the software "visible" to the program office.

Recommendation 3.7 Prepare a new cost estimate for any significant change in software requirements.

Data Requirements

The software cost estimating process requires certain programmatic data. This data should allow the estimator to determine the tools and techniques required to capture the cost of the software development. The estimating procedure should utilize prior project information for arriving at a given estimate — if reestimation is done after SRR, then project expenditure until SRR shall be considered by the model for cost reestimation.

Recommendation 3.8 Provide a software Work Breakdown Structure (WBS) and a work schedule to the cost estimator.

Configuration Management, Quality Assurance, and Independent Verification and Validation

Configuration Management (CM), itself essential for successful system development, is also part of the project cost burden. For example, a change in a software baseline configuration would demand a new cost estimate, which, in turn, imposes its own costs on the project. Additional costs also result from Quality Assurance (QA), which is obviously necessary, and Independent Verification and Validation (IV&V), which leads to early error identification and may in the long run reduce test, integration, and overall life cycle costs. Thus, the price of CM, QA, and IV&V can legitimately be regarded as an investment in improved software quality and reduced life cycle costs.

Recommendation 3.9 Anticipate the activities required for CM, QA, and IV&V in detail and include their costs in the SCE.

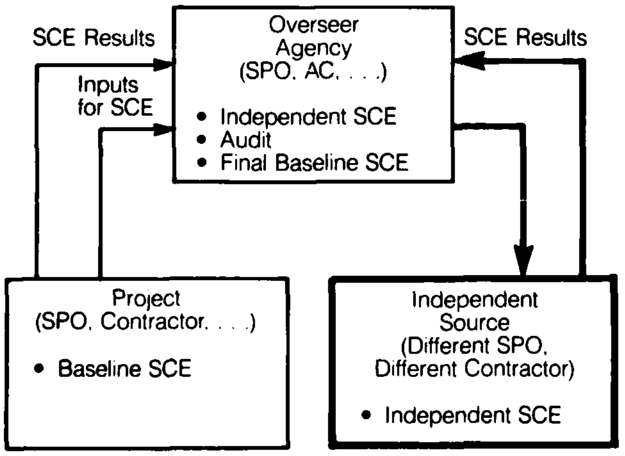
We note, in passing, that since company management style often has a signficant impact on a project, some means should be found to allow for this in a software cost estimate. It is assumed that a competent cost-management system would allow for cost estimation as well as cost tracking and provide an audit trail, no matter how large the project.

Tools and Techniques

SCE Environment

The application of SCE for embedded systems should occur in an environment represented by figure 2. The overseer agency is the customer or an agency authorized by the customer to represent the customer's interests. The project is the contractor or the government program office (depending on the contract phase) that develops the initial SCE. The independent source may be another program office or an independent contractor. The project is responsible for the initial BSCE and reports the results to the overseer agency. The overseer agency is responsible for the generation of Independent Software Cost Estimates (ISCEs), performing audits, and resolving differences in estimates to arrive at a refined BSCE. The overseer agency can contract for this work internally or externally. If the work is done internally, figure 2 becomes simplified, as the parts shown as dashed lines are eliminated.

Figure 2
Organizational Relationships for SCE


SCE Results
Overseer Agency (SPO, AC, . . .)
• Independent SCE
• Audit
• Final Baseline SCE
SCE Results
Inputs for SCE
Project (SPO, Contractor, . . .)
• Baseline SCE
Independent Source (Different SPO, Different Contractor)
• Independent SCE

Independent Software Cost Estimates

There are currently two major problems with Independent Software Cost Estimates (ISCEs): they are usually not truly independent of the project; and no guidelines exist for doing them. The ISCE group should operate in a framework that includes the overseer agency (e.g. ESD) and the project; but, to ensure an independent estimate, it should be as separate and self-contained as possible. For instance, the ISCE group might be an outside contractor hired by the overseer agency, or another group — perhaps a different project, using personnel with similar expertise — within the same organization. Greater independence, of course tends to reduce bias, but there are limits to the amount of independence an ISCE group can have and still fully understand the technical aspects of the project.

The ISCE group should receive the inputs it needs directly from the overseer agency, not from the project, although the inputs might conceivably originate there. To ensure comparability of the ISCE and BSCE, the same inputs, e.g., WBS, should be used for both. These inputs must be sufficiently detailed, otherwise the ISCE group may be forced to do design work to establish the criteria it will use for estimation. (Having inputs reviewed by another project with similar software experience will help to improve their quality.) The ISCE group should analyze the inputs for validity and possible cost implications. The overseer agency should serve as the vehicle by which the ISCE group makes its input needs known to the project. Neither the overseer agency nor the project should provide the ISCE group with any direction concerning methodology.

To perform its own ISCE, the overseer agency needs to impose on itself the same requirements and procedures that would be used on an independent organization. This means that the overseer agency must have the expertise to perform the ISCE and that the data and information needed for the ISCE must not be changed by the overseer's ability to find out in detail what the project is doing.

Recommendation 3.10 Establish guidelines for the performance of Independent SCEs (ISCEs), paying particular attention to ensure that the ISCE is truly independent, impartial, and fair, and, at the same time, relevant to the project.

Recommendation 3.11 The Baseline SCE (BSCE) and ISCE shall contain a basic estimate as well as alternative estimates based on variations in assumptions or design methodologies.

Audit

Audit takes on two broad aspects in software cost estimation. First, it supports management's role in comparing an ISCE with the BSCE. Second, it evaluates and validates the BSCE. Either aspect of the audit requires expert personnel and independence comparable to that required for the ISCE itself. ISCE and audit are distinguished by the fact that the latter uses less in the way of resources. Further, audit evaluates existing estimates without generating new data. To audit properly, the overseer agency must be able to understand the relationships between the estimates and either resolve the differences, accept the differences, or develop a position on separate factors. Finally, by combining the various factors into a bottom-line cost, the overseer should be able to generate an acceptable and supportable BSCE. The overseer agency may also have to decide if the project estimate and ISCE are so divergent that one or both have to be redone.

An audit of the project's BSCE can involve any or all of: review of assumptions, evaluation of the design, reviews of historical calibration data, review of model selection, validation of parametrics, etc. Primary benefit can be received by the review of assumptions, design considerations, and alternatives. Assuming that the project organization has performed an honest and expert analysis, made good use of historical data, and selected a proper model, etc., additional alternatives, missed assumptions, and additional design considerations can be identified by the overseer. Further changes to the BSCE may result from looking at how historical data and considerations of analogous projects were applied.

Recommendation 3.12 Establish guidelines for the Audit process to ensure as accurate a BSCE as possible.

Baseline Software Cost Estimate

The purpose of the BSCE is to provide a current description of a software project in terms of cost, schedule, and technical performance. Maintaining the BSCE is a continuous activity within the overseer agency. It begins with the initial BSCE and continues with updates throughout the various stages of development and deployment. This management tool is associated with DODI (Department of Defense Instruction) 7000.2, "Performance Measurement for Selected Acquisitions." A BSCE component should be established for each phase of the life cycle. This will allow the effects of trade-offs made at conceptual design to be related to their impacts on costs "downstream." The BSCE should be compatible with incremental development activities.

The method by which the BSCE is established is dependent on the phase of development. The type and amount of data available during the conceptual phase will determine which software cost model will be used, or even if a model will be used at all. As the software development proceeds from PDR to CDR to coding, parameters such as lines of code, number of modules and their interfaces, and the complexity are better defined and quantified. This may lead to the selection of new models. As the development process goes from one stage to the next, the precision of estimating the future cost improves. In many instances the initial BSCE is the negotiated value (contract target cost) between SPO and contractor. This value should then be traceable, from an auditing viewpoint, to all the identified WBS tasks. This assumes the WBS is sufficiently detailed, not just in levels, but to account for the various stages of software development (A Spec to B5 to C5, etc.).

Upon the establishment of the BSCE, the deviations in terms of cost and schedule performance can then be expressed in terms of variance analyses, allowing the customer's management to make decisions regarding performance trade-offs, schedule changes, etc. Provided that the BSCE is sufficiently detailed, changes in requirements in terms of added or deleted scope can be more efficiently processed and incorporated to yield a new BSCE. The short-term gain is a more accurate estimate of the project cost, since more data is available. The long-term gain is that the "downstream" data base for the total project now includes this new information in sufficient detail for management interpretation.

As the name implies, the BSCE is a baseline for management to measure performance. How it is derived and to what precision is contingent on two points: (1) when estimates are made in the life cycle, and (2) what data are available. The availability of detailed data, provided it is not cost prohibitive, can serve to make the BSCE a dynamic entity with an ever-improving accuracy.

Recommendation 3.13 Establish a BSCE for each definable phase in the life cycle of a project. To ensure traceability through later phases of the project, a description of the methodology and assumptions used should be incorporated into each BSCE.

Validation and Calibration

At specified milestones in the life cycle (or whenever significant changes in the project occur, e.g., changes in software requirements or the development environment), revised software cost estimates should be developed. These should be based on the most current information available for model input variables. The revised input data can be used in the same model or in a new model that is more appropriate to the current point in the life cycle. If actual costs-to-date are in reasonable agreement with the prior estimate for this point in the life cycle, this is a validation of the model and its parameters.

If unreasonable discrepancies occur between the actual cost and the estimate of cost, one must seriously consider the need for calibrating or fine tuning the SCE model. Calibration can be based on objective statistical methodologies (e.g., regression analysis) or can be more subjective in nature (e.g., slight adjustments to model parameters based on experience and judgement). If for the same software system, calibration is significant or must be employed frequently during the life cycle, one must seriously question the appropriateness of the current model and search for a better one.

Recommendation 3.14 Validate SCE models and, if necessary, recalibrate them as a project progresses and more data becomes available. Frequent need for recalibration should be considered grounds for questioning the accuracy or appropriateness of the model.

Criteria for Model Selection

One of the more critical tasks facing an organization responsible for SCE is that of model selection. All too often, an organization relies on a single model. The consequence of such practice may be the generation of an estimate that is inaccurate not because the model is flawed, but because it is inappropriate. Good criteria for model selection are needed to avoid such problems.

The information need of the organizational element requesting the software cost estimate represents the first criterion for considering which model to select. What is the level of detail required by the sponsor of the cost estimate? The estimator needs to know whether the estimate required is for the complete life cycle cost or simply for the cost of maintaining an operational system for some period of time. It may be that the required information must be broken down to, say, the CPCI or component level. Whatever the case, the selected model must be capable of providing the requested information to the required level of detail.

The second criterion is that the model must be capable of the accuracy required by the requesting organization. An estimate for a POM submission will differ greatly in its need for accuracy from an estimate performed following the conceptual stages of the development process. Models are often used in environments that differ considerably from those in which they were calibrated. As the need for accuracy increases, so does the need to calibrate the model to the environment for which it will be used. Attention must also be paid to the data used in the original calibration to determine whether it is possible to account for cost drivers unique to a project. A need for great accuracy also increases the amount and detail of information required by the estimator. It may also be necessary to have multiple groups participate, each generating an independent estimate. As the required accuracy increases, the evaluation of the associated risks must also be considered.

The final criterion for model selection is the inputs available to the estimators. These will invariably affect the value of the information the sponsoring organization gets. The inputs available over the life cycle will also vary considerably between projects. Sizeable differences in available inputs will most likely occur in the conceptual stages of the life cycle. For first-of-a-kind systems, the number of inputs available will undoubtedly be smaller than for a system undergoing an upgrade.

Recommendation 3.15 Establish guidelines for each SCE organization to ensure choosing SCE models that are accurate and appropriate. Where possible, models should be rated for their precision and level of detail. In particular, since models tend to be highly sensitive to the number of inputs available, this parameter should be stressed.

Shortcomings

There are many shortcomings in current software costing methodology. Notable examples are, first, the inability of most models to handle system integration, and, second, the inability to determine sizing parameters very early in the life cycle. The costs associated with the integration and testing of software components when they are combined into software systems, and of integrating software with hardware are not given visibility in most models.

Some of these costs may be included in the WBS for a "standard" development, but they cannot be varied to take into account the peculiarities of the particular system being costed. For example, the cost of integrating two subsystems developed at two separate sites cannot be distinguished from the cost of developing the entire system at one site, although, presumably, the integration costs for the two approaches would be considerably different.

Although it is advantageous to do software costing very early in the software life cycle, the basis for doing so is limited. The problem is that important design and implementation decisions have not yet been made. In the absence of a system design, it is hard to predict the size of code to be developed. Research is in progress on calculating software size on the basis of functions to be performed. This work has value as a cross-check on other predictors of size, and its outcome should be interesting. It would seem, nonetheless, that accurate prediction of software size must depend on the existence of a software design. Many of the new software development techniques will alleviate this situation to a degree, in that they reduce the number of unknowns earlier in the life cycle. Techniques such as reusable code, prototyping, incremental development, high-level languages, and application generators all reduce the amount of new design necessary at each development step. As the number of unknowns decreases, the error in the cost estimate of a program should as well.

Recommendation 3.16 Encourage and continue research to formulate methodology for SCE as early in the life cycle of software as possible. As new techniques for software development come into use, they should be examined to determine how much their characteristics can promote early, accurate SCE.

6 Working Group IV

New Directions in Software Cost Estimation

Alternate Approaches

"Dynamic-Process" Models

Current software cost estimation models are static, meaning they attempt to predict project schedule and cost as a function of estimated product size, available technology, etc., through time-invariant relationships. It may be useful to model the software life cycle process as a system itself, using a simulation tool that captures dynamic (time-varying) relationships among the life cycle processes. By representing the software life cycle system in a manner that includes component processes, dynamic causal relationships between processes, and feedback, the user can:

- predict the behavior of the software production system as a function of time;
- predict the effects of possible changes in such factors as requirements, environment, or system delays; and
- predict the effects of proposed policy changes.

One methodology well suited to implementation of the software life cycle simulation tool is the mathematics of control theory. This technique is applicable to analysis of behavior patterns. Hopefully, it will prove applicable to point prediction as well, if the software life cycle system is analyzed at a sufficient level of detail.

Alternatives to Data-Based Analyses

Currently, most software cost analyses use recorded historical data. It is also possible to use conceptual models and understanding held in the minds of experts who can deal with the system under analysis intuitively. This information includes not only the system definition and its past behavior, but also feedback and some of the relationships between pairs of system variables. Techniques exist to elicit, quantify, and validate this information, which can be applied to both static and dynamic-process models.

Although feedback can be dealt with directly in simulation models, it is not yet used in parametric cost modeling. Currently, parametric cost estimation revolves around linear regression on historic data (parameters such as number of inputs, number of files, number of lines of code, or logarithms of these) that can be altered by curve fitting. This, unfortunately, yields no better than a first-order model and steady-state solutions. Transients, such as changes in requirements after project inception, are not allowed; each stage looks for a local minimum cost to each parameter rather than a global minimum to the whole software development, and no cost feedback is returned during the project to improve these estimates.

Current parametric models fit data by minimizing the cost estimate, using only first order terms. These models can be improved by adding "moving average" forms that smooth the data. The parametrics can also be organized differently.

Assume the following work breakdown structure (WBS):

00	Software Cost
01	R&D Cost
011	Requirements Definition
012	IV&V Expenditure
02	Acquisition Cost
021	Tool Development (compilers, etc.)
022	S/W Test
.	
.	
.	

A current strategy uses linear regression on parameters (such as lines of code in the compiler for 021) to determine a "minimal" or actual cost, then modifies that value by a curve — the learning curve for the new language. For each element of the WBS, cost estimation is accomplished by an appropriate linear regression. These WBS costs can then be aggregated and modified if necessary, taking into account their interactions, by curve fitting techniques.

An alternate strategy is to organize the WBS major elements into state vectors. Each vector has an associated statistical weight that can vary in time. The cost equation then roughly resembles the Ricatti equation. The individual state vectors can be estimated via linear regression as before, or other techniques can be used. The gains now are not constants, but statistical weights that can vary in time, or over application type. This approach looks at the entire vector space as an irregular nth-order surface.

The analysis problem now is to find a region of global minimization. A typical control theory approach is to optimize on one of the state vectors while holding the others constant. For example, in the optimization of a missile trajectory, one can first minimize rate of fuel consumption, then path length, then weight distribution, and finally control. Such an approach will find the most sensitive state vector in the model. Feedback gains are then initially calculated to achieve minimum cost or optimum performance.

In a sampled system, such as cost estimates with periodic feedback via costs from a WBS, costs can be accumulated on a scheduled basis and used to adjust the gain figures. This method assumes linearity and statistical independence among the model state vectors. It may not be possible to measure values that are indeed independent. In this case simulation modeling, where the context of the model changes as more becomes known about the inputs, is a more feasible model-development strategy. Both optimal control and dynamic-process modeling provide for adaptation to new inputs (changes in requirements) and allow for feedback into the model such that costs may be controlled as well as estimated.

Recommendation 4.1 Investigate control-theory or simulation-modeling techniques for modeling the software-development process as a system in order to determine if a dynamic (time-varying) model can be applied to SCE. Such a model might satisfy the following requirements:

- mapping to historic data;
- allowing changing and behavioral inputs;
- accomodating a WBS;
- improving prediction estimates of a standard model;
- improving accuracy in dynamic environments; and
- providing engineering trade-offs such as development vs. maintenance without noticeably increasing the complexity of the model's input data base.

The Impact of Emerging Technologies

Introduction

New technologies that impact software development cost can be divided into two categories according to the way they affect SCE. Some have such sweeping effects that they dictate the use of new methodology. Others are more easily handled, in that they merely give rise to new values in the parameters of existing SCE models.

Methodology Drivers

There are two major technologies that will have a significant impact on the methods used in software costing. These are Artifical Intelligence (AI) and distributed development workstations.

The AI (or knowledge-based systems) development environment is radically different from the traditional one. AI development groups are expected to implement software in a less formal, more evolutionary manner than current cost-estimation models and techniques can represent. The interactions between developer and user and between developer and source(s) to define the knowledge base are expected to be more dominant than in present software development projects. Also, the degree of testing required to guarantee proper system performance is not directly related to functional requirements. These differences will require a reformulation of the estimation models. If, since AI model input parameters are still undefined, we use source lines of code (SLOC) from standard software development as a guide, we might project that factors such as the number of data items and the number of logical inference rules will become the AI development model inputs.

Recommendation 4.2 Determine if the number of data items and the number of logical inference rules can be used as a replacement for source lines of code in SCE for artificial intelligence systems.

The isolated workstation (lone programmer) concept that has recently resurfaced as part of a distributed workstation concept meant to improve software productivity creates a different set of estimation problems. The major problem is that the lone programmer, a statistical sample of one, limits the ability to estimate the development schedule and cost within an acceptable variance.

Parametric Drivers

Some of the emerging technologies can be represented in current parametric software cost estimation models by simply using new values for existing input parameters such as size, complexity, application, and environment. These technologies include Ada programming and its support environment, programming via application generators and very high-level languages (VHLs), the use of a program design language (PDL), the programming of distributed processing systems, and firmware technology.

In the case of Ada, several factors will complicate the revaluation of model parameters; it is recommended that these be considered, validated, and calibrated. First, because Ada is so radically different from FORTRAN, COBOL, et. al, the learning curve for it will rise more slowly. Next, allowance must be made for the availability, maturity, and commonality of the Ada Programming Support Environment (APSE).

Newly emerging application generators and VHLs should primarily affect the productivity variables in standard models. For size-driven parametric models, VHLs can be characterized by scaling the size variable (e.g., by letting one line in VHL equal five lines in FORTRAN). We would expect such a re-scaling also to represent the lower costs due to reduced errors in the implementation and improvements in locating and removing errors.

Since PDL usage implies earlier detection of errors and better definition of functional requirements, testing requirements should decrease for a given level of reliability. PDL can be considered a particular "tool." Tools are commonly accounted for by changes in the environment variable, in terms of the software development environment, or the environment in which the software will be utilized. Since the operating environment will demand a certain level of reliability and testing, the effect of PDL usage can be represented by adjusting the variable that describes the specification environment, and/or modifying the extent of the testing phase to reflect the greater efficiency of analysis and design.

Distributed processing software may be represented in the models by a larger complexity variable.

Firmware technology (existing as well as emerging) requires adjustment of parametric variables to account for increased costs of testing, and verification and validation, when standard software cost estimation models are used. Hardware costs can be estimated by using standard techniques. The wide variation of impact on firmware maintenance due to development styles must be noted. Some developers test and document firmware to a greater degree than they would straight software deliverables, thereby improving reliability and ease of maintenance. But others document less (providing, sometimes, as little as a bit map), anticipating maintenance to be a throw-away-and-re-do activity.

Mapping these technologies into new values for existing models' parameters cannot always be done immediately. In many cases, further data must be collected. This is true for the impact of the Ada programming language on productivity — the first significant entries for a new data base are not yet available. For VHLs, the data to be collected must reveal not only productivity ratios (e.g., will five lines of a new VHL be written at the rate of one line of FORTRAN?), but the degree of functionality carried (i.e., how much of the requirements specification is now programmed by these five lines of VHL?).

New definitional issues and data collection must also be carried out in order to relegate the use of PDL to parametric cost modeling. Earlier studies provided values for adjusting the output of parametric models according to the environmental variable, "use of modern programming practices" either as yes, no, or degree of use. But that is no longer satisfactory for structured analysis and design. New data collection is needed to isolate PDL utilization from the "use of modern programming practices" aggregate, especially to reveal whether the cost savings is accounted for by less testing, fewer errors, and/or more efficient re-work of design activities.

Non-Impacting Technologies

The group consensus is that VLSI will have no apparent impact on software cost estimation methodology. Constraints of speed, function, and complexity will be handled on the chip and not with the software.

Fault-tolerant software development is expected to impact cost estimation only minimally. Some consideration must be given to the additional coding needed to achieve such protection, the more extensive testing performed, and the V&V activity.

Recommendation 4.3 Collect further data for emerging technologies in order to develop new values for existing model parameters. The following must be resolved:

- The impact of the Ada programming language and the Ada Programming Support Environment (APSE) on productivity;
- The productivity improvement and functionality per line of code for very-high-level languages;
- The impact on cost savings when using a PDL;
- Whether increased complexity variables can be used to represent distributed processing systems;
- The impact on test, V&V, and maintenance costs for firmware developments.

Alternatives to Source Lines of Code

To facilitate accurate and timely SCE, it is necessary to identify the requirements to be satisfied and the software product to be developed as early as possible and in a more quantitative, concrete fashion. Most often, cost estimation methodologies use SLOC as a measure of the product size. However, we should focus on the information available at the time we are making the estimate. In the early conceptual stages, this known information relates more strongly to requirements or function. One problem is to identify these known entities. Another is to identify a measure of user needs that is understandable by both user and developer for use in software cost estimation. A third problem is using the estimate for early cost trade-offs of system capabilities.

Such cost analyses can employ measures such as function counts, interface counts, number of requirements, test counts, etc. Every software system has several levels of function elaboration. Hence, one should be able to produce an estimate based on the number of "boxes" at a certain level of abstraction. The objective is to link a quantitative measure of requirements to the cost of effecting the requirements.

Recommendation 4.4 Use quantitative measures of requirements, derived from PSL/PSA, SADT, data flow diagrams, or other tools, as input to the SCE process.

Recommendation 4.5 Establish relationships between the functions identified during functional decomposition of systems and SLOC.

Software Upgrades

Software upgrades go by different names and differ radically in character and extent. Table 3 indicates various types of upgrades and the major elements that are usually changed during the course of the upgrade. Certain factors or sources of change can become dominant cost drivers in a given upgrade, e.g.,

- Quality of existing documentation
- Degree of structure within the existing software design
- Knowledge of the functionality of existing potentially reusable software.
- Knowledge of the new language
- Degree of test/retest required
- Knowledge of the new hardware/operating system/utilities/tools, etc.
- Understanding of the new functionality and how it relates to the existing functionality.
- Capabilities of, maturity, documentation, and ease of use of the selected off-the-shelf (OTS) software packages, e.g., data-base management systems (DBMSs).
- Scale of the overall upgrade.

Some of these factors and sources of change can be translated into SLOC or other standard model factors; however, they tend to be highly subjective, based on little experience and no calibration. For example, effects of the use of software packages have not been extensively reported or analyzed, and are hard to translate into SLOC-oriented models.

Table 3
Software Upgrade Categories

Upgrade Level	Added Function	Language Change	Hardware Change	Use of OTS S/W Packages
Enhancement	No (1)	No	No	No
Conversion	No	Maybe (2)	Maybe (2)	Maybe
Adaptation	No	No	Yes	Yes
Modification	Yes	No	No	No
Modernization	Yes	Maybe	Yes	Yes
Reuse	Yes	No	No	Yes

Notes (1) Enhancement is herein taken to imply revision to a program for the purpose of its running more efficiently.
(2) Conversion may affect either a change of language or a change of hardware, or both.

The architecture of the new system is greatly affected by the extent of new requirements (added SLOC) and by the reusability of existing code and by the use of OTS packages (reduced effective SLOC). Likewise, the required level and extent of development and integration test varies widely across upgraded systems. These factors cause great divergence in the standard cost breakouts built into existing models (e.g., 40% design, 20% coding, 40% test), and in the schedules derived from the estimates of man-month requirements. Models notwithstanding, it has been suggested that upgrades be treated as new developments whenever they exhibit "enough" sheer scale, degree of added requirements, and/or degree of use of reusable software. "Enough" has yet to be quantified.

Recommendation 4.6 **Gather and analyze applicable experience data and case studies for software upgrades to see what the key factors were that affected costs and schedules.**

Recommendation 4.7 **Modify existing models for use in upgrade projects by defining new model parameters (or recombining or rescaling existing model parameters) to account for key upgrade factors.**

Recommendation 4.8 **Develop new models specifically tailored to software upgrade projects.**

Appendix A

Workshop Schedule

Tuesday, September 13

8:00 - 9:00
Registration

9:00 - 9:30
Colonel Kanter (ESD/AC) - Keynote Address

9:30 - 10:15
Barry Boehm (TRW)
"Research Issues in SCE"

10:15 - 10:30
Coffee Break

10:30 - 11:00
Bob Thibodeau (GRC)
"SCE Techniques vs. Life Cycle Phases"

11:00 - 11:30
Al Kopania (Aerospace)
"Software Size Estimating Model"

11:30 - 12:00
Major Duquette (HQ AFSC)
"Problems, Experiences, ..."

12:00 - 1:00
Lunch in Atrium

1:00 - 1:30
John Gaffney (IBM)
"Software Function Estimates"

1:30 - 2:00
Merle McKenzie (JPL)
"Software Life Cycle Model"

2:00 - 2:30
Carolyn Wong (SDC)
"Technical Estimates of Cost Model"

2:30 - 3:00
Al Roberts (MITRE)
"Making Costing Count"

3:00 - 3:15
Break

3:15 - 5:00
Working Groups

5:00 - 6:30
Cocktail Party in Atrium

Wednesday, September 14

8:00 - 10:00
Working Groups

10:00 - 10:15
Coffee Break

10:15 - 12:00
Working Groups

12:00 - 1:00
Lunch at Stouffer's Bedford Glen

1:00 - 2:00
General Session - Group Objectives

2:00 - 3:00
Working Groups

3:00 - 3:15
Coffee Break

3:15 - 5:00
Working Groups

Thursday, September 15

8:00 - 9:50
Reports from Groups I & II

9:50 - 10:10
Coffee Break

10:10 - 12:00
Reports from Groups III & IV

12:00 - 1:00
Lunch

1:00 - 3:00
Complete Reports

Workshop Committee

General Program Chairman

Col. D. G. Kanter (ESD/AC)

Technical Committee Chairman

Herman Schultz (MITRE)

Technical Committee

D. Bergstrom (RADC/COEE)
Lt. Col. Bowen (ESD/ACC)
J. Cavano (RADC/COEE)
A. J. Chruscicki (RADC/COEE)
1 Lt. J. Dean (ESD/ACCE)
Maj. J. Duquette (Hq USAF/ACMC)
Dr. J. L. Katz (MITRE)
W. Letendre (ESD/ALEE)

PREVIOUS PAGE IS BLANK

Appendix C

List of Participants

This appendix contains the names and addresses of all workshop participants, including speakers, committee, etc.

Mr. Tarek K. Adbel-Hamid
6 Whittier Place, Apt. 8E
Boston, MA 02114
(617) 367-9271
(617) 253-2781 (MIT)

Mr. Deane Bergstrom
RADC/COEE
Griffiss AFB, NY 13441
(315) 330-3827

Dr. Barry W. Boehm
TRW Defense Systems Group
One Space Park
Redondo Beach, CA 90278
(213) 535-2184

Lt. Col. Ronald Bowen
ESD/ACC
Hanscom AFB, MA 01731
(617) 861-5120

Mr. Elmer Branyon
General Electric Co.
Space Systems Division
P.O. Box 8048
Philadelphia, PA 19101

Mr. Joseph Cavano
RADC/COEE
Griffiss AFB, NY 13441
(315) 330-4875

Mr. Andrew Chruscicki
RADC/COEE
Griffiss AFB, NY 13441
(315) 330-4875

Ms. Judith A. Clapp
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-3755

Ms. Ellen Coakley
ESD/ACCE
Hanscom AFB, MA 01731
(617) 861-2788

Mr. James Couch
Ford Aerospace and Communications
P.O. Box 58487
Houston, TX 77058
(713) 280-6613

Mr. Robert Cruickshank
IBM Federal Systems Division
9500 Godwin drive
Manassas, VA 22110
(703) 367-3258

Mr. James Davos
DCASMA
495 Summer Street
Boston, MA 02210
(617)451-4064

Capt. Joe P. Dean
ESD/ACCE
Hanscom AFB, MA 01731
(617)861-5223

Mr. Samuel DiNitto
RADC/COE
Griffiss AFB, NY 13441

Mr. Dan Douglas
TASC
1 Jacob Way
Reading, MA 01867
(617)944-6850

Mr. Roger Dumas
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-3676

Maj. Joseph Duquette
HQ USAF/ACMC
Room 4D214
The Pentagon
Washington, DC 20330
A/V 227-0711
(202) 697-0211

Mr. Lorraine Duval
IITRI
199 Liberty Plaza
Rome, NY 13440
(315) 336-2359

Mr. Kurt F. Fischer
CSC
6565 Arlington Blvd.
Falls Church, VA 22046
(703) 237-2000

Mr. Frank Flett
TASC
1 Jacob Way
Reading, MA 01867
(617) 944-6850

Lt. Roy A. Flowers
ESD/ALEE
Hanscom AFB, MA 01731
(617) 861-2716

Mr. Samuel Frank
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-3535

Mr. John E. Gaffney, Jr.,
IBM Federal Systems Division
18100 Frederick Pike
Gaithersburg, MD 20879
(301) 840-6461

Mr. Carl Greenbaum
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 861-3634

Mr. Stephen Gross,
MAT01F4
NAVMAT Cost Analysis Division
Crystal Plaza 5, Room 902
Washington, DC 20360

Mr. David Grynberg
Grumman Aerospace
Steward AVenue
Bethpage, NY 11714
(516) 875-2767

Ms. Mary Jean Hayes
The MITRE Corporation
Burlington Road
Bedford, MA 01720
(617)271-8244

Mr. Eleanor Hetrick
ITT Programming Technology Center
1000 Oronoque Lane
Stratford, CT 06497
(203) 375-0200

Mr. Rocco Iuorno
IITRI
199 Liberty Plaza
Rome, NY 13440
(315) 336-0937

Mr. John H. James
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-2995

Dr. Randall W. Jensen,
618/B218
Hughes Aircraft Co.
P.O. Box 3310
Fullerton, CA 92634
(714) 732-8087

Col. David G. Kanter
ESD/AC
Hanscom, AFB, 01731
(617)861-5161

Dr. Joseph L. Katz
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-7328

Mr. Hall Katzenbach
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-7900

Mr. Al Kopania
Aerospace Corporation
P.O. Box 92957
Los Angeles, CA 90009
(213) 615-4447

Mr. Michael Koscielski
USAF/SD
P.O. Box 92960
Worldwide Postal Center
Los Angeles, CA 90009
(213) 643-1772

Mr. William E. Kuhn
RCA/Price Systems
Route 38, Bldg. 204-1
Cherry Hill, NJ 08358
(609) 338-5962

Mr. Gerard R. LaCroix
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-3758

Mr. Edward L. Lafferty
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-2773

Ms. Annette Lanigan
Calspan
P.O. Box 9X
Lexington, MA 02173
(617) 863-1705

Mr. Ronald B. Leask,
C3251
NUSC
New London Laboratory
New London, CT 06320
(203) 447-4366

Mr. W. Letendre
ESD/ALEE
Hanscom AFB, MA 01731
(617)861-5112

Mr. Joseph Levitan
Shell Oil Company
P.O. Box 20329
Houston, TX 77025(713)
795-2765

Ms. M. J. Leslie Lowry
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-7963

Ms. Joan Lovelace
The MITRE Corpora-
tion1820 Dolley Madison
Blvd.
McLean, VA 22102
(703) 827-6154

Col. Dennis Madl
HQ AFSC/ACC
Andrews AFB, MD 20334
(301) 981-5127

Mr. Ralph Maibor
Dynamics Research Corp.
60 Concord Street
Wilmington, MA 01887
(617)658-7685, Ext. 1229

Mr. Christopher Mankiewich
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102
(703) 827-7281

Mr. James McCall
Science Applications, Inc.
P.O. Box 2351
La Jolla, CA 92038
(619) 456-6220

Ms. Merle McKenzie
MS/238-540
JPL
4800 Oak Grove Drive
Pasadena, CA 91109
9213) 354-2577

Dr. Siba N. Mohanty
The MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22101
(703) 827-6017

Mr. Andrew Najberg
TASC
1 Jacob Way
Reading, MNA 01867
(617) 944-6850

Mr. Perry R. Nuhn
ITT Programming Technol-
ogy Center
1000 Oronoque Lane
Stratford, CT 06497
(203) 375-0200

Dr. Shashi Phoha
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-2353

Mr. Alan J. Roberts
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-3573

Mr. Kyle Rone
IBM Federal Systems Divi-
sion
1322 Space Park Drive
Houston, TX 77058
(713) 333-7378

Mr. Ralph San Antonio
Dynamics Research Corp.
60 Concord Street
Wilmington, MA 01887
(617) 658-7685, Ext. 1229

Mr. Herman P. Schultz
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-2274

Mr. J. Rod Sheffield
SofTech, Inc.
3100 Presidential Drive
Fairborn, OH 45324
(513) 42902771

Mr. John Shimer
NSA (N-32)
9800 Savage Road
Ft. Meade, MD 20755
(301) 688-7208

Mr. Charles Spear
ASD/ACCR
Wright Patterson AFB, OH
45433
(513) 255-5904

Mr. William M. Stein
The MITRE Corporation
Burlington Road
Bedford, MA 01730
(617) 271-8238

Mr. Richard Swanson
Aquidneck Data Corpora-
tion
P.O. Box 99
Middletown, RI 02840
(401) 847-7260

Mr. Robert Thibodeau
General Research Corpora-
tion
307 Wynn Drive, NW
Huntsville, AL 35805
(205) 837-7900

Mr. Frederick Titsworth
Raytheon Missile Systems
Division
Bedford, MA 01730
(617) 274-7100

Mr. Emil Vukasovich
AD/ACII
Eglin AFB, FL 32542
A/V-872-2126
(904) 882-2126

Mr. Norman Walenczyk
Norden Systems
P.O. Box 5300
Norwalk, CT 06856
(203) 852-4978

Mr. David Winningham
HQ AFCMD/EPER
Kirtland AFB, NM 87117
(505) 844-0859

Ms. Carolyn Wong
MD32-61
Systems Development
Corp.
2500 Colorado AVenue
Santa Monica, CA 90406
(213) 820-4111, X4302

Mr. Alice Youngblood
AD/ACCI
Eglin AFB, FL 32542
A/V 872-2126
(904) 882-2126

Appendix D

Working Group Members

This appendix contains a list of the participants in each working group.

Group I
LEASK (NUSC)

Branyan (GE)
Coakley (AF/ESD)
Couch (FORD)
Dumas (MITRE)
Grynberg (GRUMN)
Iuorno (IITRI)
Katzenbach (MITRE)
Lanigan (Calspan)
Madl (AFSC)
Maibor (DRC)
Najberg (TASC)
Nuhn (ITT)
Shimer (NSA)
Spear (ASD)
Titsworth (RAYTH)
Youngblood (AD)

Group II
BOEHM (TRW)

Abdel-Hamid (MIT)
Cavano (RADC)
Cruickshank (IBM)
Douglas (TASC)
Fischer (CSC)
Flowers (AF/ESD)
Hetrick (ITT)
Lafferty (MITRE)
Lowry (MITRE)
McCall (SAI)
San Antonio (DRC)
Walenczyk (NORDEN)

Group III
THIBODEAU (GRC)

Chruscicki (RADC)
Davos (DCASMA)
Earles (E-E)
Flett (TASC)
Greenbaum (MITRE)
Gross (NAVMAT)
Hayes (MITRE)
James (MITRE)
Koscielski (AF/SD)
Levitan (SHELL)
Lovelace (MITRE)
Mohanty (MITRE)
Rone (IBM)
Vukasovich (AD)
Winningham (AFCMD)

Group IV
JENSEN (HUGHES)

Dean (ESD)
DiNitto (RADC)
Duvall (IITRI)
Frank (MITRE)
Gaffney (IBM)
Kopania (AERO)
Kuhn (PRICE-S)
Mankiewich (M-W)
McKenzie (JPL)
Phoha (MITRE)
Roberts (MITRE)
Sheffield (SOFT)
Stein (MITRE)
Swanson (ADC)
Wong (SDC)

Appendix E

Subjects Suggested to the Working Groups

Group I
Cost Effective
Software Data Collection
on Defense Programs

- Defining a software WBS
- Defining technical data (cost drivers) to supplement cost data
- Determining optimum data collection levels/frequency
- Institutionalizing data collection in MIL-STDs and DIDs
- Cost/benefit of creating/maintaining a detailed data base

Group II
Integrating SCE
with Program Management

- Impact of SCE data collection on program managers (WBS and DID)
- Use of SCE data to measure software status
- Role of SCE in determining revised "estimates to complete"
- Maintaining a dual purpose data base (for SCE and program management)

Group III
Organization
and Performance of SCE

- Use of one or more SCE models
- Model calibration
- Modeling personnel requirements/costs
- Life Cycle Costing
- Model shortcomings and solutions
- Modeling system integration

Group IV
New Directions

- Identifying promising new methods and approaches to SCE
- Impact of new technologies/e.g., Ada, PDL, VLSI
- New acquisition strategies to reduce Conceptual Phase uncertainty
- Eliminating the nemesis to SCE: software size estimation
- Software Acquisition Process Model (SWAP)