SOFTWARE ENGINEERING LABORATORY SERIES

SEL-82-1206

ANNOTATED BIBLIOGRAPHY OF SOFTWARE ENGINEERING LABORATORY LITERATURE

NOVEMBER 1993

National Aeronautics and
Space Administration

Goddard Space Flight Center
Greenbelt, Maryland 20771

Foreword

The Software Engineering Laboratory (SEL) is an organization sponsored by the National Aeronautics and Space Administration/Goddard Space Flight Center (NASA/GSFC) and created to investigate the effectiveness of software engineering technologies when applied to the development of applications software. The SEL was created in 1976 and has three primary organizational members:

NASA/GSFC, Software Engineering Branch

University of Maryland, Department of Computer Science

Computer Sciences Corporation, Software Engineering Operation

The goals of the SEL are (1) to understand the software development process in the GSFC environment; (2) to measure the effect of various methodologies, tools, and models on this process; and (3) to identify and then to apply successful development practices. The activities, findings, and recommendations of the SEL are recorded in the Software Engineering Laboratory Series, a continuing series of reports that includes this document.

The major contributors to this document are

Linda Morusiewicz (Computer Sciences Corporation)
Jon Valett (NASA/GSFC)

Single copies of this document can be obtained by writing to

Software Engineering Branch
Code 552
Goddard Space Flight Center
Greenbelt, Maryland 20771

Abstract

This document is an annotated bibliography of technical papers, documents, and memorandums produced by or related to the Software Engineering Laboratory. Nearly 200 publications are summarized. These publications cover many areas of software engineering and range from research reports to software documentation.

This document has been updated and reorganized substantially since the original version (SEL-82-006, November 1982). All materials have been grouped into eight general subject areas for easy reference:

- The Software Engineering Laboratory
- The Software Engineering Laboratory: Software Development Documents
- Software Tools
- Software Models
- Software Measurement
- Technology Evaluations
- Ada Technology
- Data Collection

This document contains an index of these publications classified by individual author.

Table of Contents

Section 7—Technology Evaluations

Section 1—Introduction

This document is an annotated bibliography of technical papers, documents, articles, and memoranda produced by or related to the Software Engineering Laboratory (SEL). It is intended to provide a quick reference to the published results of SEL research and development activities.

More than 100 publications are summarized in this document. Each summary includes the size of the publication (number of pages), a description (abstract) of its contents, and its original citation. Previous versions and subsequent reprintings are also identified where appropriate.

The publications described here cover many aspects of software engineering and range from research reports to software documentation. They are divided into eight general subject areas:

- The Software Engineering Laboratory
- The Software Engineering Laboratory: Software Development Documents
- Software Tools
- Software Models
- Software Measurement
- Technology Evaluations
- Ada Technology
- Data Collection

Appendix A contains a list of references that are no longer available.

An index is included at the end of this document to assist in identifying materials by author. Publications, with their corresponding section numbers, are alphabetized in order of the last names of individual authors.

Copies of individual publications listed in this bibliography can be obtained from one or more of the sources shown in Table 1-1. The acronyms defined in the table appear after each abstract and indicate the document's availability. Any material not labeled with one of these acronyms can be obtained only from the author(s).

Table 1-1. Availability of SEL Literature

Acronym	Source	Address
SEB	Software Engineering Branch	Code 552 Goddard Space Flight Center Greenbelt, MD 20771
CASI	NASA Center for Aerospace Information[1] (and source above)	P.O. Box 8757 BWI Airport, MD 21240
NTIS	National Technological Information Service[2] (and sources above)	5285 Port Royal Road Springfield, VA 22161
JAO	Journals and other private publishers	See specific citation

[1] Open to Federal Government agencies only at no charge.

[2] There is a per-page charge for reprinting documents.

Section 2—The Software Engineering Laboratory

2.1 *Collected Software Engineering Papers: Volume I,* SEL-82-004, July 1982, 118 pages

This document is a collection of technical papers prepared by SEL participants during the 5-year period ending December 31, 1981. The 10 papers are organized into 4 major topics and each paper's section number within this Annotated Bibliography is provided for further reference. *NTIS*

Technical Paper	Section
THE SEL ORGANIZATION	
"The Software Engineering Laboratory: Objectives," V. R. Basili and M. V. Zelkowitz	2.29
"Operation of the Software Engineering Laboratory," V. R. Basili and M. V. Zelkowitz	2.30
RESOURCE MODELS	
"Resource Estimation for Medium-Scale Software Projects," M. V. Zelkowitz	5.2
"A Meta-Model for Software Development Resource Expenditures," J. W. Bailey and V. R. Basili	5.8
"Can the Parr Curve Help With Manpower Distribution and Resource Estimation Problems?" V. R. Basili and J. Beane	5.9
SOFTWARE MEASURES	
"Measuring Software Development Characteristics in the Local Environment," V. R. Basili and M. V. Zelkowitz	6.3
"Programming Measurement and Estimation in the Software Engineering Laboratory," V. R. Basili and K. Freburger	6.5
"Evaluating and Comparing Software Metrics in the Software Engineering Laboratory," V. R. Basili and T. Phillips	6.6
SOFTWARE ENGINEERING APPLICATIONS	
"Models and Metrics for Software Management and Engineering," V. R. Basili	5.5
"Use of Cluster Analysis to Evaluate Software Engineering Methodologies," E. Chen and M. V. Zelkowitz	7.4

2.2 *Collected Software Engineering Papers: Volume II*, SEL-83-003, November 1983, 100 pages

This document is a collection of technical papers prepared by SEL participants from January 1, 1982, through November 30, 1983. The nine papers are organized into four major topics and each paper's section number within this Annotated Bibliography is provided for further reference. *NTIS*

Technical Paper	Section
THE SOFTWARE ENGINEERING LABORATORY	
"Measuring Software Technology," W. W. Agresti, F. E. McGarry, D. N. Card, et al.	6.10
"Technical Summary 1982: Report to the National Aeronautics and Space Administration," V. R. Basili	Other References
RESOURCE MODELS	
"Comparison of Regression Modeling Techniques for Resource Estimation," D. N. Card	5.11
"Early Estimation of Resource Expenditures and Program Size," D. N. Card	6.7
SOFTWARE MEASURES	
"Metric Analysis and Data Validation Across FORTRAN Projects," V. R. Basili, R. W. Selby, and T. Phillips	6.9
"Monitoring Software Development Through Dynamic Variables," C. W. Doerflinger and V. R. Basili	5.12
"Software Errors and Complexity: An Empirical Investigation," V. R. Basili and B. T. Perricone	6.11
DATA COLLECTION	
A Methodology for Collecting Valid Software Engineering Data, V. R. Basili and D. M. Weiss	9.5
"Data Collection and Evaluation for Experimental Computer Research Data," M. V. Zelkowitz	9.4

2.3 *Collected Software Engineering Papers: Volume III*, SEL-85-003, November 1985, 132 pages

This document is a collection of technical papers prepared by SEL participants from November 30, 1983, through November 1, 1985. The 12 papers are organized into 3 major topics and each paper's section number within this Annotated Bibliography is provided for further reference. *NTIS*

Technical Paper	Section
THE SOFTWARE ENGINEERING LABORATORY	
"A Software Technology Evaluation Program," D. N. Card	7.14
"A Methodology for Collecting Valid Software Engineering Data," V. R. Basili and D. M. Weiss	9.6
TECHNOLOGY EVALUATION	
"Quantitative Evaluation of Software Methodology," V. R. Basili	7.13
"Four Applications of a Software Data Collection and Analysis Methodology," V. R. Basili and R. W. Selby, Jr.	7.12
"Measuring the Impact of Computer Resource Quality on the Software Development Process and Product," F. E. McGarry, J. Valett, and D. Hall	7.8
"Analyzing the Test Process Using Structural Coverage," J. Ramsey and V. R. Basili	7.7
"A Practical Experience With Independent Verification and Validation," G. Page, F. E. McGarry, and D. N. Card	7.6
"ARROWSMITH-P—A Prototype Expert System for Software Engineering Management," V. R. Basili and C. L. Ramsey	7.22
SOFTWARE MEASUREMENT	
"Finding Relationships Between Effort and Other Variables in the SEL," V. R. Basili and N. M. Panlilio-Yap	5.15
"Calculation and Use of an Environment's Characteristic Software Metric Set," V. R. Basili and R. W. Selby, Jr.	6.17
"Criteria for Software Modularization," D. N. Card, G. T. Page, and F. E. McGarry	6.16
"Evaluating Software Development by Analysis of Changes: Some Data From the Software Engineering Laboratory," D. M. Weiss and V. R. Basili	6.18

2.4 *Collected Software Engineering Papers: Volume IV*, SEL-86-004, November 1986, 108 pages

This document is a collection of technical papers prepared by SEL participants from November 1, 1985, through September 30, 1986. The six papers are organized into three major topics and each paper's section number within this Annotated Bibliography is provided for further reference. *NTIS*

Technical Paper	Section
THE SOFTWARE ENGINEERING LABORATORY	
"Experimentation in Software Engineering," V. R. Basili, R. W. Selby, Jr., and D. H. Hutchens	5.16
"An Approach for Assessing Software Prototypes," V. E. Church, D. N. Card, W. W. Agresti, and Q. L. Jordan	7.16
"Towards a General Object-Oriented Software Development Methodology," E. Seidewitz and M. Stark	8.3
TECHNOLOGY EVALUATION	
"An Empirical Study of Software Design Practices," D. N. Card, V. E. Church, and W. W. Agresti	7.15
"Designing with Ada for Satellite Simulation: A Case Study," W. W. Agresti, V. E. Church, D. N. Card, and P. L. Lo	8.2
SOFTWARE MEASUREMENT	
A Study on Fault Prediction and Reliability Assessment in the SEL Environment, V. R. Basili and D. Patnaik	5.17

2.5 *Collected Software Engineering Papers: Volume V*, SEL-87-009, November 1987, 284 pages

This document is a collection of technical papers prepared by SEL participants from September 1, 1986, through January 1, 1988. The 16 papers are organized into 3 major topics and each paper's section number within this Annotated Bibliography is provided for further reference. *NTIS*

Technical Paper	Section
SOFTWARE MEASUREMENT AND TECHNOLOGY STUDIES	
"A Summary of Software Measurement Experiences in the Software Engineering Laboratory," J. D. Valett and F. E. McGarry	6.28
"Evaluating Software Engineering Technologies," D. N. Card, F. E. McGarry, and G. T. Page	7.19
"Resolving the Software Science Anomaly," D. N. Card and W. W. Agresti	5.18
"A Controlled Experiment on the Impact of Software Structure on Maintainability," H. D. Rombach	6.20
"An Evaluation of Expert Systems for Software Engineering Management," C. L. Ramsey and V. R. Basili	7.17
"Comparing the Effectiveness of Software Testing Strategies," V. R. Basili and R. W. Selby	7.21
MEASUREMENT ENVIRONMENT STUDIES	
"Tailoring the Software Process to Project Goals and Environments," V. R. Basili and H. D. Rombach	5.19
"TAME: Tailoring an Ada™ Measurement Environment," V. R. Basili and H. D. Rombach	8.7
"TAME: Integrating Measurement Into Software Environments," V. R. Basili and H. D. Rombach	6.21
A Meta Information Base for Software Engineering, L. Mark and H. D. Rombach	5.21
Characterizing Resource Data: A Model for Logical Association of Software Data, D. R. Jeffery and V. R. Basili	*5.24*
ADA TECHNOLOGY STUDIES	
"Measuring Ada for Software Development in the Software Engineering Laboratory (SEL)," F. E. McGarry and W. W. Agresti	8.13

2.6 *Collected Software Engineering Papers: Volume VI*, SEL-88-002, November 1988, 153 pages

This document is a collection of technical papers prepared by SEL participants from June 1, 1987, through January 1, 1989. The 12 papers are organized into 3 major topics and each paper's section number within this Annotated Bibliography is provided for further reference. *NTIS*

Technical Paper	Section
SOFTWARE MEASUREMENT AND TECHNOLOGY STUDIES	
"The Effectiveness of Software Prototyping: A Case Study," M. V. Zelkowitz	7.18
"Measuring Software Design Complexity," D. N. Card and W. W. Agresti	6.25
"Quantitative Assessment of Maintenance: An Industrial Case Study," H. D. Rombach and V. R. Basili	7.20
"Resource Utilization During Software Development," M. V. Zelkowitz	6.22
MEASUREMENT ENVIRONMENT STUDIES	
"Generating Customized Software Engineering Information Bases from Software Process and Product Specifications," L. Mark and H. D. Rombach	5.22
"Software Process and Product Specifications: A Basis for Generating Customized SE Information Bases," H. D. Rombach and L. Mark	5.23
"The TAME Project: Towards Improvement-Oriented Software Environments," V. R. Basili and H. D. Rombach	6.24
"Validating the TAME Resource Data Model," D. R. Jeffery and V. R. Basili	6.23
ADA TECHNOLOGY STUDIES	
"Experiences in the Implementation of a Large Ada Project," S. Godfrey and C. Brophy	8.17
"General Object-Oriented Software Development with Ada: A Life Cycle Approach," E. Seidewitz	8.16
"Lessons Learned in the Implementation Phase of a Large Ada Project," C. E. Brophy, S. Godfrey, W. W. Agresti, and V. R. Basili	8.15
"Object-Oriented Programming in Smalltalk and Ada," E. Seidewitz	8.12

2.7 *Collected Software Engineering Papers: Volume VII*, SEL-89-006, November 1989, 157 pages

This document is a collection of technical papers prepared by SEL participants from December 1988 through October 1989. The seven papers are organized into three major topics and each paper's section number within this Annotated Bibliography is provided for further reference. *NTIS*

Technical Paper	Section
SOFTWARE MEASUREMENT AND TECHNOLOGY STUDIES	
Establishing a Measurement Based Maintenance Improvement Program: Lessons Learned in the SEL, H. Rombach and B. Ulery	6.29
Maintenance = Reuse-Oriented Software Development, V. Basili	5.26
Software Development: A Paradigm for the Future, V. Basili	5.27
MEASUREMENT ENVIRONMENT STUDIES	
Integrating Automated Support for a Software Management Cycle into the TAME System, T. Sunazuka and V. Basili	6.30
Towards A Comprehensive Framework for Reuse: A Reuse-Enabling Software Evolution Environment, V. Basili and H. Rombach	5.25
ADA TECHNOLOGY STUDIES	
"Evolution of Ada Technology in a Production Software Environment," F. McGarry, L. Esker, and K. Quimby	8.22
"Using Ada to Maximize Verbatim Software Reuse," M. Stark and E. Booth	8.23

2.8 *Collected Software Engineering Papers: Volume VIII*, SEL-90-005, November 1990, 125 pages

This document is a collection of technical papers prepared by SEL participants from November 1989 through October 1990. The seven papers are organized into four major topics and each paper's section number within this Annotated Bibliography is provided for further reference. *NTIS*

Technical Paper	Section
SOFTWARE MEASUREMENT STUDIES	
"Design Measurement: Some Lessons Learned," H. Rombach	6.31
SOFTWARE MODELS STUDIES	
Towards a Comprehensive Framework for Reuse: Model-Based Reuse Characterization Schemes, V. Basili and H. Rombach	5.28
"Viewing Maintenance as Reuse-Oriented Software Development," V. Basili	5.29
SOFTWARE TOOLS STUDIES	
"Evolution Towards Specifications Environment: Experience with Syntax Editors," M. Zelkowitz	7.23
ADA TECHNOLOGY STUDIES	
"On Designing Parametrized Systems using Ada," M. Stark	8.27
"PUC: A Functional Specification Language for Ada," P. Straub and M. Zelkowitz	8.28
"Software Reclamation: Improving Post-Development Reusability," J. Bailey and V. Basili	8.26

2.9 *Collected Software Engineering Papers: Volume IX*, SEL-91-005, November 1991, 199 pages

This document is a collection of technical papers prepared by SEL participants from November 1990 through November 1991. The eight papers are organized into three major topics and each paper's section number within this Annotated Bibliography is provided for further reference. *NTIS*

Technical Paper	Section
SOFTWARE MODELS STUDIES	
"Software Reuse: A Key to the Maintenance Problem," H. D. Rombach	5.30
Support for Comprehensive Reuse, V. R. Basili and H. D. Rombach	5.31
A Reference Architecture for the Component Factory, V. R. Basili and G. Caldiera	5.32
A Pattern Recognition Approach for Software Engineering Data Analysis, L. C. Briand, V. R. Basili, and W. M. Thomas	5.33
SOFTWARE MEASUREMENT STUDIES	
"Paradigms for Experimentation and Empirical Studies in Software Engineering," V. R. Basili and R. W. Selby	6.33
ADA TECHNOLOGY STUDIES	
"Object-Oriented Programming Through Type Extension in Ada 9X," E. Seidewitz	8.32
"An Object-Oriented Approach to Parameterized Software in Ada," E. Seidewitz and M. Stark	8.33
"Designing Configurable Software: COMPASS Implementation Concepts," E. W. Booth and M. E. Stark	8.35

2.10 *Collected Software Engineering Papers: Volume X*, SEL-92-003, November 1992, 164 pages

This document is a collection of technical papers prepared by SEL participants from October 1991 through November 1992. The 11 papers are organized into 5 major topics and each paper's section number within this Annotated Bibliography is provided for further reference. *NTIS*

Technical Paper	Section
THE SOFTWARE ENGINEERING LABORATORY	
"The Software Engineering Laboratory—An Operational Software Experience Factory," V. Basili, G. Caldiera, F. McGarry, et al.	2.34
SOFTWARE TOOLS STUDIES	
"Towards Automated Support for Extraction of Reusable Components," S. K. Abd-El-Hafiz, V. R. Basili, and G. Caldiera	4.5
"Automated Support for Experience-Based Software Management," J. D. Valett	4.7
SOFTWARE MODELS STUDIES	
"The Software-Cycle Model for Re-Engineering and Reuse," J. W. Bailey and V. R. Basili	5.35
"On the Nature of Bias and Defects in the Software Specification Process," P. A. Straub and M. V. Zelkowitz	5.36
"An Improved Classification Tree Analysis of High Cost Modules Based Upon an Axiomatic Definition of Complexity," J. Tian, A. Porter, and M. V. Zelkowitz	5.37
"Providing an Empirical Basis for Optimizing the Verification and Testing Phases of Software Development," L. C. Briand, V. R. Basili, and C. J. Hetmanski	5.38
"A Classification Procedure for the Effective Management of Changes During the Maintenance Process," L. C. Briand and V. R. Basili	5.39
SOFTWARE MEASUREMENT STUDIES	
"Toward Full Life Cycle Control: Adding Maintenance Measurement to the SEL," H. D. Rombach, B. T. Ulery, and J. D. Valett	6.34
ADA TECHNOLOGY STUDIES	
"Object-Oriented Programming with Mixins in Ada," E. Seidewitz	8.36
"Software Engineering Laboratory Ada Performance Study—Results and Implications," E. W. Booth and M. E. Stark	8.37

2.11 *Collected Software Engineering Papers: Volume XI*, SEL-93-001, November 1993, 99 pages

This document is a collection of technical papers prepared by SEL participants from November 1992 through November 1993. The five papers are organized into three major topics and each paper's section number within this Annotated Bibliography is provided for further reference. *NTIS*

Technical Paper	Section
SOFTWARE MODELS STUDIES	
"Developing Interpretable Models with Optimized Set Reduction for Identifying High Risk Software Components," L. C. Briand, V. R. Basili, and C. J. Hetmanski	5.40
"Modeling and Managing Risk Early in Software Development," L. C. Briand, W. M. Thomas, and C. J. Hetmanski	5.41
"An Information Model for Use in Software Management Estimation and Prediction," N. R. Li and M. V. Zelkowitz	5.42
SOFTWARE MEASUREMENT STUDIES	
"Measuring and Assessing Maintainability at the End of High Level Design," L. C. Briand, S. Morasca, and V. R. Basili	6.35
TECHNOLOGY EVALUATION STUDIES	
"Impacts of Object–Oriented Technologies: Seven Years of SEL Studies," M. Stark	7.25

2.12 *Proceedings from the First Summer Software Engineering Workshop,* SEL-76-001, August 1976, 194 pages

This document reproduces the presentations made by participants at the First Summer Software Engineering Workshop held on August 5, 1976, at GSFC. The general topic of the conference was software design. Also, available state-of-the-art software development techniques were surveyed and their applicability to the GSFC environment was considered. The presentations were grouped into the following panels:

- Requirements analysis and design methodologies
- Program design languages
- Automated software tools

Papers related to the SEL are

- V. Basili (University of Maryland), "Program Design Languages"
- E. Damon (NASA/GSFC), "DOMONIC As a Design and Management Tool"
- M. V. Zelkowitz (University of Maryland), "Automated Tools"

NTIS

2.13 *Proceedings from the Second Summer Software Engineering Workshop*, SEL-77-002, September 1977, 146 pages

This document reproduces the presentations made by participants at the Second Summer Software Engineering Workshop held on September 19, 1977, at GSFC. This second workshop attempted to communicate with the larger software engineering research community. Approaches and experiences with the design of experiments and data collection were reviewed. The presentations were grouped into the following panels:

- Experimental design
- Models, measures, and metrics
- Data collection
- Software engineering experiences

The only paper related to the SEL is V. R. Basili and M. V. Zelkowitz (University of Maryland), "Overview of the Software Engineering Laboratory." *NTIS*

2.14 *Proceedings from the Third Summer Software Engineering Workshop*, SEL-78-005, September 1978, 132 pages

This document reproduces the presentations made by participants at the Third Summer Software Engineering Workshop held on September 18, 1978, at GSFC. Many of the discussions at this third workshop dealt with how one collects software data and how one conducts successful software experiments. The presentations were grouped into the following panels:

- The Data collection process
- Validation of software development models
- Measuring software development methodologies
- Current activities and future directions

Papers related to the SEL are

- R. W. Reiter, Jr. (University of Maryland), "Investigating Software Development Approaches: A Synopsis"
- V. R. Basili and M. V. Zelkowitz (University of Maryland), "The Software Engineering Laboratory—1978"

NTIS

2.15 *Proceedings from the Fourth Summer Software Engineering Workshop*, SEL-79-005, November 1979, 282 pages

This document reproduces the presentations made by participants at the Fourth Summer Software Engineering Workshop held on November 19, 1979, at GSFC. This fourth workshop focused on actual experiences of data collection and the application of software methodologies, models, and tools. The presentations were grouped into the following panels:

- The SEL
- Data collection
- Experiments in methodology evaluation
- Software resource models
- Models and metrics of software development

Papers related to the SEL are

- F. McGarry (NASA/GSFC), "Overview of the Software Engineering Laboratory"
- V. E. Church (CSC), "Software Engineering Laboratory—The Data Collection Process"
- M. V. Zelkowitz and E. Chen (University of Maryland), "Software Engineering Laboratory: Data Validation"
- V. R. Basili (University of Maryland), "Investigations Into Software Development in the Software Engineering Laboratory"
- P. C. Belford, R. A. Berg (CSC), and T. L. Hannan (FAA), "Central Flow Control Software Development: A Case Study of the Effectiveness of Software Engineering Techniques"

NTIS

2.16 *Proceedings from the Fifth Annual Software Engineering Workshop*, SEL-80-006, November 1980, 242 pages

This document reproduces the presentations made by participants at the Fifth Annual Software Engineering Workshop held on November 24, 1980, at GSFC. This fifth workshop focused on actual experiences with the application of software methodologies and models. The presentations were grouped into the following panels:

- The SEL
- Software cost/resource modeling
- Software reliability
- Measurement of the development process

Papers related to the SEL are

- F. McGarry (NASA/GSFC), "An Approach To Measuring Software Technology"
- J. Page (CSC), "Impacts of Experiments and Software Technology Changes in a Production Environment"
- V. Basili and J. Bailey (University of Maryland), "Measuring the Effects of Specific Software Methodologies Within the SEL"

NTIS

2.17 *Proceedings of the Sixth Annual Software Engineering Workshop*, SEL-81-013, December 1981, 282 pages

This document reproduces the presentations made by participants at the Sixth Annual Software Engineering Workshop held on December 2, 1981, at GSFC. This sixth workshop was an attempt to gather the experiences of software developers in applying modern programming practices and other software engineering techniques. The document also includes a summary of the presentations and audience comments. The presentations were grouped into the following panels:

- Evaluating software development characteristics
- Software metrics
- Software models
- Software methodologies

Papers related to the SEL are

- J. Page (CSC), "Methodology Evaluation: Effects of Independent Verification and Integration on One Class of Application"
- V. R. Basili (University of Maryland), "Assessment of Software Measures in the Software Engineering Laboratory"
- D. N. Card (CSC), "Identification and Evaluation of Software Measures"

NTIS

2.18 *Proceedings of the Seventh Annual Software Engineering Workshop*, SEL-82-007, December 1982, 400 pages

This document reproduces the presentations made by participants at the Seventh Annual Software Engineering Workshop held on December 1, 1982, at GSFC. The major emphasis of this seventh workshop was on reporting and discussing actual experiences with software methodologies, models, and tools. The document also includes a summary of the presentations and audience remarks. The presentations were grouped into the following panels:

- The SEL
- Software tools
- Software errors
- Software cost estimation

Papers related to the SEL are

- V. R. Basili and B. T. Perricone (University of Maryland), "Software Errors and Complexity: An Empirical Investigation"
- M. V. Zelkowitz (University of Maryland), "Software Prototyping in the Software Engineering Laboratory"
- I. Miyamoto (University of Maryland), "User Interface Design of a Software Tool System as a Technology Transfer Vehicle"

NTIS

2.19 *Proceedings of the Eighth Annual Software Engineering Workshop*, SEL-83-007, November 1983, 316 pages

This document reproduces the presentations made by participants at the Eighth Annual Software Engineering Workshop held on November 3, 1983 at GSFC. The document also includes a summary of the presentations and audience remarks. The presentations were grouped into the following panels:

- The SEL
- Software testing
- Human factors in software engineering
- Software quality assessment

Papers related to the SEL are

- D. N. Card (CSC), F. E. McGarry (GSFC), and G. Page (CSC), "Evaluating Software Engineering Technologies in the SEL"
- C. W. Doerflinger and V. R. Basili (University of Maryland), "Monitoring Software Development Through Dynamic Variables"
- J. Sukri and M. V. Zelkowitz (University of Maryland), "Characteristics of a Prototyping Experiment"
- J. Ramsey (University of Maryland), "Structural Coverage of Functional Testing"
- B. Schneiderman, C. Grantham, K. Norman, et al. (University of Maryland), "Evaluating Multiple Coordinated Windows for Programmer Workstations"

NTIS

2.20 *Proceedings of the Ninth Annual Software Engineering Workshop*, SEL-84-004, November 1984, 349 pages

This document reproduces the presentations made by participants at the Ninth Annual Software Engineering Workshop held on November 28, 1984, at GSFC. The document also includes a summary of the presentations and audience remarks. The presentations were grouped into the following panels:

- The SEL
- Software error studies
- Experiments with software development
- Software tools

Papers related to the SEL are

- W. W. Agresti (CSC), "An Approach to Developing Specification Measures"
- R. W. Selby, Jr., V. R. Basili (University of Maryland), J. Page (CSC), and F. E. McGarry (NASA/GSFC), "Evaluating Software Testing Strategies"
- V. R. Basili et al. (University of Maryland), "Software Development in Ada"
- H. D. Rombach (University of Maryland), "Design Metrics for Maintenance"

NTIS

2.21 *Proceedings of the Tenth Annual Software Engineering Workshop*, SEL-85-006, December 1985, 360 pages

This document reproduces the presentations made by participants at the Tenth Annual Software Engineering Workshop held on December 4, 1985, at GSFC. The document also includes a summary of the presentations and audience remarks. The presentations were grouped into the following panels:

- The SEL
- Tools for software management
- Software environments
- Experiments with Ada

Papers related to the SEL are

- B. Agresti (CSC), "Measuring Ada As a Software Development Technology in the SEL"
- V. Basili (University of Maryland), "Can We Measure Software Technology; Lessons From 8 Years of Trying"
- F. E. McGarry (NASA/GSFC), "Recent SEL Studies"
- J. Valett (NASA/GSFC) and A. Raskin (Yale), "DEASEL: An Expert System for Software Engineering"

NTIS

2.22 *Proceedings of the Eleventh Annual Software Engineering Workshop*, SEL-86-006, December 1986, 308 pages

This document reproduces the presentations made by participants at the Eleventh Annual Software Engineering Workshop held on December 3, 1987, at GSFC. The document also includes a summary of the presentations and audience remarks. The presentations were grouped into the following panels:

- The SEL
- Empirical studies of software technology
- Software environments
- Software testing

Papers related to the SEL are

- F. McGarry, S. Voltz, and J. Valett (NASA/GSFC), "Determining Software Productivity Leverage Factors in the SEL"
- V. R. Basili and H. D. Rombach (University of Maryland), "TAME: Tailoring A Measurement Environment"
- M. V. Zelkowitz (University of Maryland), "Automating the Design Process with Syntactic-Based Tools"
- W. W. Agresti (CSC), "SEL Ada Experiment: Status and Design Experiences"

NTIS

2.23 *Proceedings of the Twelfth Annual Software Engineering Workshop*, SEL-87-010, December 1987, 479 pages

This document reproduces the presentations made by participants at the Twelfth Annual Software Engineering Workshop held on December 2, 1987, at GSFC. The document also includes a summary of the presentations and audience remarks. The presentations were grouped into the following panels:

- Studies and experiments with Ada
- Experiments with environments
- Case studies
- Measures of cost and reliability

The only paper related to the SEL is

- S. Godfrey (NASA/GSFC) and C. Brophy (University of Maryland), "An Experiment in Ada in the Software Engineering Laboratory—Lessons Learned from the Ada Code/Unit Test Phase"

NTIS

2.24 *Proceedings of the Thirteenth Annual Software Engineering Workshop*, SEL-88-004, November 1988, 379 pages

This document reproduces the presentations made by participants at the Thirteenth Annual Software Engineering Workshop, held on November 30, 1988, at GSFC. The document also includes a summary of the presentations and audience remarks. The presentations were grouped into the following panels:

- Studies and experiments in the SEL
- Software models
- Study of software products
- Tools

Papers related to the SEL are

- F. McGarry (NASA/GSFC), L. Esker, and K. Quimby (CSC), "Evolving Impact of Ada on a Production Software Environment"
- V. R. Basili and H. D. Rombach (University of Maryland), "Towards a Comprehensive Framework for Reuse: A Reuse-Enabling Software Evolution Environment"
- J. D. Valett (NASA/GSFC), W. Decker, and J. Buell (CSC), "The Software Management Environment"

NTIS

2.25 *Proceedings of the Fourteenth Annual Software Engineering Workshop*, SEL-89-007, November 1989, 390 pages

This document reproduces the presentations made by participants at the Fourteenth Annual Software Engineering Workshop, held on November 29, 1989, at GSFC. The document also includes a summary of the presentations and audience remarks. The presentations were grouped into the following panels:

- Studies and Experiments in the SEL
- Methodologies
- Software Reuse
- Testing and Error Analysis

Papers related to the SEL are

- V. R. Basili (University of Maryland), "The Experience Factory: Packaging Software Experience"
- F. McGarry (NASA/GSFC), S. Waligora, and T. McDermott (CSC), "Experiences in the Software Engineering Laboratory (SEL) Applying Software Measurement"
- A. Kouchakdjian (University of Maryland), S. Green (NASA/GSFC), and V. Basili (University of Maryland), "Evaluation of the Cleanroom Methodology in the Software Engineering Laboratory"
- H. D. Rombach and B. T. Ulery (University of Maryland), J. Valett (NASA/GSFC), "Measurement Based Improvement of Maintenance in the SEL"

NTIS

2.26 *Proceedings of the Fifteenth Annual Software Engineering Workshop*, SEL-90-006, November 1990, 566 pages

This document reproduces the presentations made by participants at the Fifteenth Annual Software Engineering Workshop, held November 28 and 29, 1990, at GSFC. The document also includes a summary of the presentations and audience remarks. The presentations were grouped into the following sessions:

- The SEL at Age 15
- Process Improvement
- Measurement
- Reuse
- Process Assessment

The sessions were followed by two panel discussions:

- Experiences in Implementing an Effective Measurement Program
- Software Engineering in the 1980s: Most Significant Achievements/Greatest Disappointments

Papers related to the SEL are

- V. R. Basili (University of Maryland), "Towards a Mature Measurement Environment: Creating a Software Engineering Research Environment"
- G. H. Heller and G. T. Page (CSC), "Impact of a Process Improvement Program in a Production Software Environment: Are We Any Better?"
- F. McGarry and R. Pajerski (NASA/GSFC), "Towards Understanding Software—15 Years in the SEL"
- P. A. Straub and M. V. Zelkowitz (University of Maryland), "Bias and Design in Software Specifications"
- R. Kester (CSC), "SEL Ada Reuse Analysis and Representations"

NTIS

2.27 *Proceedings of the Sixteenth Annual Software Engineering Workshop*, SEL-91-006, December 1991, 364 pages

This document reproduces the presentations made by participants at the Sixteenth Annual Software Engineering Workshop, held December 4 and 5, 1991, at GSFC. Summaries of both the session and the panel presentations and transcripts of the panel discussions are included. The presentations were grouped into the following sessions:

- The SEL
- Investigating Errors
- Process Analysis
- Testing Verification
- Life-Cycle Issues

The sessions were followed by two panel discussions:

- An International Perspective on Software Engineering in the 80's: Most Significant Accomplishments and Greatest Disappointments
- SEI Process Maturity Model: Use/Misuse

Papers related to the SEL are

- V. R. Basili and G. Caldiera (University of Maryland), "Methodological and Architectural Issues in the Experience Factory"
- S. E. Green and R. Pajerski (NASA/GSFC), "Cleanroom Process Evolution in the SEL"
- F. McGarry (NASA/GSFC) and S. Waligora (CSC), "Experiments in Software Engineering Technology"
- A. Porter and L. Briand (University of Maryland), "Optimized Set Reduction for Empirically Guiding Software Development"

NTIS

2.28 *Proceedings of the Seventeenth Annual Software Engineering Workshop*, SEL-92-004, December 1992, 440 pages

This document reproduces the presentations made by participants at the Seventeenth Annual Software Engineering Workshop, held December 2 and 3, 1992, at GSFC. Summaries of all session presentations are provided, as well as transcripts of discussions following each session. Summaries of the panel presentations and discussion are also included. The presentations were grouped into the following sessions:

- The SEL
- Process Measurement
- Reuse
- Software Quality
- Lessons Learned

The sessions were followed by a panel discussion, "Is Ada Dying?" Papers related to the SEL are

- F. McGarry (NASA/GSFC), "Experimental Software Engineering: Seventeen Years of Lessons in the SEL"
- V. R. Basili (University of Maryland), "The Experience Factory: Can It Make You a 5?"
- M. Stark (NASA/GSFC), "Impacts of Object-Oriented Technologies: Seven Years of SEL Studies"
- S. Waligora and J. Langston (CSC), "Maximizing Reuse: Applying Common Sense and Discipline"
- R. D. Pendley, C. H. Noonan, and K. R. Hall (CSC), "Development and Application of an Acceptance Testing Model"

NTIS

2.29 "The Software Engineering Laboratory: Objectives," V. R. Basili and M. V. Zelkowitz, *Proceedings of the Fifteenth Annual Conference on Computer Personnel Research*, August 1977, 14 pages

This technical paper provides an overview of the SEL and its objectives. The original motivations for establishing the SEL were the high cost of software development and the subsequent need to optimize the development process. This paper discusses the following aspects of the SEL with respect to the following motivations:

- Specific objectives of the SEL
- Software development factors to be investigated
- Data collection techniques
- Early SEL research activities

The importance of defining consistent software development measures is a recurrent theme throughout the discussion. *JAO*

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

2.30 "Operation of the Software Engineering Laboratory," V. R. Basili and M. V. Zelkowitz, *Proceedings of the Second Software Life Cycle Management Workshop*, August 1978, 4 pages

This technical paper describes the operation of the SEL. Software engineering data is regularly collected by the SEL from flight dynamics software development projects at GSFC. The assembled data support an extensive program of software engineering research. This report also reviews SEL data collection and data processing activities and their relationship to the research program. It also summarizes some ongoing resource estimation and error analysis research projects. *JAO*

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

2.31 *The Software Engineering Laboratory*, SEL-81-104, D. N. Card, F. E. McGarry, G. Page, et al., February 1982, 121 pages

This document describes the history, organization, operation, and research results of the SEL. The SEL is a joint effort of GSFC, Computer Sciences Corporation, and the University of Maryland. The objective of the SEL is to study and improve the software development process in the GSFC environment. The SEL has conducted extensive research in the following areas of software engineering:

- Methodology evaluation
- Tool evaluation
- Resource models
- Reliability models
- Software measures

The document outlines SEL efforts in these areas and presents some preliminary conclusions based on this work. The appendixes include descriptions of the software projects studied and summary statistics derived from these data. *NTIS*

The previous version of this document was *The Software Engineering Laboratory*, SEL-81-004, D. N. Card, F. E. McGarry, G. Page, et al., September 1981. This document was also issued as Computer Sciences Corporation document CSC/TM-82/6033.

2.32 *Glossary of Software Engineering Laboratory Terms*, SEL-82-105, T. A. Babst, M. G. Rohleder, and F. E. McGarry, November 1983, 39 pages

This document is a glossary of terms used in the SEL. A list of acronyms is also included. The terms are defined within the context of the software development environment for flight dynamics at GSFC. The purposes of this document are to provide a concise reference for clarifying the language employed in SEL documents and data collection forms, establish standard definitions for use by SEL personnel, and explain basic software engineering concepts. *SEB*

The previous version of this document was *Glossary of Software Engineering Laboratory Terms*, SEL-82-005, M. G. Rohleder, December 1982. A version of this document was also issued as Computer Sciences Corporation document CSC/TM-83/6168.

2.33 *Software Engineering Laboratory (SEL) Data and Information Policy (Revision 1)*, SEL-91-102, F. McGarry, August 1991, 24 pages

This document presents the policies and overall procedures that are used in distributing and in making available products of the SEL. The products include project data and measures, source code, reports, and software tools. *NTIS*

The previous version of this document was *Software Engineering Laboratory (SEL) Data and Information Policy*, SEL-91-002, F. McGarry, April 1991.

2.34 "The Software Engineering Laboratory—An Operational Software Experience Factory," V. Basili, G. Caldiera, F. McGarry, et al., *Proceedings of the Fourteenth International Conference on Software Engineering (ICSE 92)*, May 1992, 12 pages

Software engineering technology transfer needs a top-down, experimental, evolutionary framework to produce models and an experimental laboratory to measure, evaluate, and refine those models. Currently, three major concepts support this vision: the Quality Improvement Paradigm, the Goal/Question/Metric Approach, and the Experience Factory. This paper discusses these concepts and summarizes the background, goals, and operations of the SEL and how they relate to these major concepts. It then maps the SEL iterative data analysis process to the experience factory functions, and concludes that the SEL is a functioning example of an operational software experience factory. Some lessons learned from 15 years of SEL operations, major benefits derived from the SEL's measurement program, and implications for development organizations outside the flight dynamics environment are included. *JAO*

This technical paper also appears in SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992.

Section 3—The Software Engineering Laboratory: Software Development Documents

3.1 *Recommended Approach to Software Development (Revision 3)*, SEL-81-305, L. Landis, S. Waligora, F. McGarry, et al., June 1992, 226 pages

This document presents guidelines for an organized, disciplined approach to software development that is based on studies conducted by the SEL since 1976. It describes methods and practices for each phase of a software development life cycle that starts with requirements definition and ends with acceptance testing. For each defined life cycle phase, this document presents guidelines for the development process and its management, and for the products produced and their reviews. *NTIS*

The previous versions of this document were *Standard Approach to Software Development*, SEL-81-005, V. E. Church, F. E. McGarry, and G. Page, September 1981; *Recommended Approach to Software Development*, SEL-81-105, S. Eslinger, F. E. McGarry, and G. Page, May 1982; and *Recommended Approach to Software Development*, SEL-81-205, F. E. McGarry, G. Page, S. Eslinger, et al., April 1983.

3.2 *Manager's Handbook for Software Development (Revision 1)*, SEL-84-101, L. Landis, F. McGarry, S. Waligora, et al., November 1990, 91 pages

This document presents methods and aids for the management of software development projects. The recommendations are based on analyses and experiences of the SEL with flight dynamics software development. The management aspects of the following subjects are described:

- Organizing the project
- Producing a development plan
- Estimating costs
- Scheduling
- Staffing
- Preparing deliverable documents
- Using management tools
- Monitoring the project
- Conducting reviews
- Auditing
- Testing
- Certifying

Revision 1 contains extensive updates, including additional material on management metrics and revisions to cost estimation factors, document contents, and testing procedures. *NTIS*

The previous version of this document was *Manager's Handbook for Software Development*, SEL-84-001, W. W. Agresti, F. E. McGarry, D. N. Card, et al., April 1984.

3.3 *Programmer's Handbook for Flight Dynamics Software Development*, SEL-86-001, R. Wood and E. Edwards, March 1986, 272 pages

Specific procedures, standards, and styles are provided as recommended guidelines for programmers' use during the detailed design and implementation phases of flight dynamics software development. Brief descriptions of the other life-cycle phases are included for reference and context. *NTIS*

3.4 *Software Verification and Testing,* SEL-85-005, D. N. Card, C. Antle, and E. Edwards, December 1985, 64 pages

General procedures for software verification and validation are provided as a guide for managers, programmers, and analysts involved in software development. The verification and validation procedures described are based primarily on testing techniques. Testing refers to the execution of all or part of a software system for the purpose of detecting errors. Planning, execution, and analysis of tests are outlined in this document. Code reading and static analysis techniques for software verification are also described. *NTIS*

3.5 *Product Assurance Policies and Procedures for Flight Dynamics Software Devlopment*, SEL-87-011, S. Perry, et al., March 1987, 106 pages

The product assurance policies and procedures necessary to support flight dynamics software development projects for Goddard Space Flight Center are presented. The quality assurance and configuration management methods and tools for each phase of the software development life cycle are described, from requirements analysis through acceptance testing. Maintenance and operation are not addressed. *CASI*

This document supersedes *Configuration Management and Control: Policies and Procedures*, SEL-84-002, Q. L. Jordan and E. Edwards, December 1984.

Section 4—Software Tools

4.1 *FORTRAN Static Source Code Analyzer Program (SAP) System Description (Revision 1)*, SEL-82-102, W. A. Taylor and W. J. Decker, April 1985, 217 pages

This document presents the FORTRAN Static Source Code Analyzer Program (SAP) System Description (Revision 1). SAP is a software tool designed to assist SEL personnel in conducting studies of FORTRAN programs. SAP scans FORTRAN source code and produces reports giving statistics and measures of statements and structures that make up a module. The document describes the processing performed by SAP; the routines, COMMON blocks, and files used by SAP; and the SAP system generation procedure.

This document follows the SAP tool specifics on the VAX-11/780, Version 3. The IBM 4341 is the batch Version 3. Departures from the VAX-11/780 version are noted. The PDP-11/70 version is an older version and is presented in SEL-82-002, *FORTRAN Static Source Code Analyzer Program (SAP) System Description*, August 1982. *SEB*

The previous version of this document was *FORTRAN Static Source Code Analyzer Program (SAP) System Description*, SEL-82-002, W. J. Decker and W. A. Taylor, August 1982.

4.2 *FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 3)*, SEL-78-302, W. J. Decker and W. A. Taylor, July 1986, 145 pages

This document presents the FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 3). SAP is a software tool designed to assist SEL personnel in conducting studies of FORTRAN programs. SAP scans FORTRAN source code and produces reports that present statistics and measures of statements and structures that make up a module. The document provides instructions for operating SAP and contains information useful in interpreting SAP output. It is a revision of the previous SAP user's guide, SEL-78-202, and is the result of integrating SAP into the Software Development Environment (SDE). *CASI*

The previous versions of this document were *FORTRAN Static Source Code Analyzer Program (SAP) User's Guide*, SEL-78-002, E. M. O'Neill, S. R. Waligora, C. E. Goorevich, et al., February 1978; *FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 1)*, SEL-78-102, W. J. Decker and W. A. Taylor, September 1982; and *FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 2)*, SEL-78-202, W. J. Decker and W. A. Taylor, April 1985.

4.3 *Flight Dynamics System Software Development Environment (FDS/SDE) Tutorial*, SEL-86-003, J. C. Buell and P. I. Myers, July 1986, 137 pages

The Flight Dynamics System Software Development Environment (FDS/SDE) is an interactive tool for developing software in the flight dynamics environment. It uses a menu-driven, fill-in-the-blanks format that permits the developer to input, edit, compile, link, and execute software. Online help is provided at all steps, minimizing training time to use the tool. This document provides the steps for a sample development scenario using the FDS/SDE, presenting sample displays and user responses. *NTIS*

4.4 *Software Management Environment (SME) Concepts and Architecture (Revision 1)*, SEL-89-103, R. Hendrick, D. Kistler, and J. Valett, September 1992, 94 pages

This document presents the concepts and architecture of the Software Management Environment (SME), developed for the Software Engineering Branch (Code 552) of the Flight Dynamics Division of GSFC. The SME provides an integrated set of experience-based management tools that can assist software development managers in managing and planning flight dynamics software development projects. This document provides a high-level description of the types of information required to implement such an automated management tool, and it presents an architectural framework in which a set of management services can be provided. *NTIS*

The previous version of this document was *Software Management Environment (SME) Concepts and Architecture*, SEL-89-003, W. Decker and J. Valett, August 1989.

4.5 "Towards Automated Support for Extraction of Reusable Components," S. K. Abd-El-Hafiz, V. R. Basili, and G. Caldiera, *Proceedings of the IEEE Conference on Software Maintenance—1991 (CSM 91)*, October 1991, 8 pages

Successful reuse can increase quality and productivity; however, several problems still limit reuse. Existing software models are not designed to benefit from or support reuse. They should be replaced with models that take advantage of reuse, introduce more reusable resources, and overcome existing reuse limitations. This paper presents, in detail, the features of extracting reusable components in the framework of the experience factory. It discusses an existing reuse-oriented process model to aid in component extraction. The paper then introduces a system, Computer-Aided Reuse Engineering (CARE), which has been designed to support the proposed process model. The system has two parts: a component identifier, which uses software metrics; and a component qualifier. The paper focuses on the component qualifier, which generates formal specifications and a significant set of test cases and packages them for future use. A prototype tool, the CARE Functional Specification Qualifier (FSQ), aids in understanding programs by using Mills' functional model of correctness to derive their specifications. It can be applied to complete programs or fragments. The formal foundation and implementation of the tool are discussed. The paper concludes with an example to demonstrate an operational scenario of the tool. *JAO*

This technical paper also appears in SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992.

4.6 *Software Managment Environment (SME) Installation Guide*, SEL-92-001, D. Kistler and K. Jeletic, January 1992, 42 pages

This document contains installation information for the Software Management Environment (SME), developed for the Software Engineering Branch (Code 552) of the Flight Dynamics Division of GSFC. The SME provides an integrated set of management tools that can be used by software development managers in their day-to-day management and planning activities. This document provides a list of hardware and software requirements as well as detailed installation instructions and troubleshooting information. *SEB*

4.7 "Automated Support for Experience-Based Software Management," J. D. Valett, *Proceedings of the Second Irvine Software Symposium (ISS '92)*, March 1992, 19 pages

To manage a software development project effectively, the software manager must have access to key information concerning a project's status. This information includes not only data relating to the project of interest, but also the experience of past development efforts within the environment. This paper describes the concepts and functionality of a software management tool, the Software Management Environment (SME), which is designed to provide this information. This tool enables the software manager to

- compare an ongoing development effort with previous efforts and with models of the "typical" project within the environment
- predict future project status
- analyze a project's strengths and weaknesses
- assess the project's quality

To provide these functions, the tool utilizes a vast corporate memory that includes

- a database of software metrics
- a set of models and relationships that describe the software development environment
- a set of rules that capture other knowledge and the experience of software managers within the environment

Because it integrates these major concepts into one software management tool, the SME is a model of the type of management tool needed for all software development organizations. *JAO*

This technical paper also appears in SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992.

Section 5—Software Models

5.1 *Applicability of the Rayleigh Curve to the SEL Environment*, SEL-78-007, T. E. Mapp, December 1978, 27 pages

This document reviews the resource utilization model for software development, which is based on the Rayleigh curve developed by Norden and Putnam. A Rayleigh curve is fit to data provided by the SEL. Parabolas, trapezoids, and straight lines are also fit to the same data. The parabola and trapezoid give about as good a fit as the Rayleigh curve. Therefore, this document concludes that although the Rayleigh curve may be an appropriate model for resource expenditures, it is not necessarily the best model for small to medium size projects. *CASI*

5.2 "Resource Estimation for Medium-Scale Software Projects," M. V. Zelkowitz, *Proceedings of the Twelfth Conference on the Interface of Statistics and Computer Science.* New York: IEEE Computer Society Press, 1979, 6 pages

This technical paper describes the analysis of resource estimation techniques that is being performed by the SEL. The data used in the analysis are collected from medium-scale flight dynamics software development projects at GSFC. A procedure to forecast accurately the cost and development time of these projects would be a valuable management tool in this environment. This paper documents a specific at tempt to verify the resource estimation model based on the Rayleigh curve that was developed by Norden and Putnam. *JAO*

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

5.3 *The Software Engineering Laboratory: Relationship Equations*, SEL-79-002, K. Freburger and V. R. Basili, May 1979, 67 pages

This document presents the results of an analysis of several factors affecting software development. The analysis was based on data collected by the SEL. Relationships among the following measures were studied:

- Total effort (staff-months)
- Lines of delivered code (thousands)
- Lines of developed code (thousands)
- Percentage of developed code
- Number of modules
- Number of developed modules
- Percentage of developed modules
- Project duration (months)
- Pages of documentation
- Productivity
- Average staff size

Estimating equations were derived from the measures by statistical analysis and were then compared with results obtained by Walston and Felix in a similar study. *CASI*

This document was also issued as University of Maryland Technical Report TR-764.

5.4 *Tutorial on Models and Metrics for Software Management and Engineering*, SEL-80-008, V. R. Basili, 1980, 349 pages

This document is a tutorial on quantitative methods of soft ware management and engineering. A quantitative methodology is needed to evaluate, control, and predict software development and maintenance costs. This quantitative approach allows cost, time, and quality tradeoffs to be made in a systematic manner. The tutorial focuses on numerical product-oriented measures such as size, complexity, and reliability and on resource-oriented measures such as cost, schedules, and resources. Twenty articles from software engineering literature are reprinted in this document. The articles are organized into the following sections:

- Resource models
- Changes and errors
- Product metrics
- Data collection

Successful application of these techniques, however, requires a thorough knowledge of the project under development and any assumptions made. Only then can these techniques augment good managerial and engineering judgment. *JAO*

This document was published as the IEEE tutorial, *Models and Metrics for Software Management and Engineering.* New York: IEEE Computer Society Press, 1980.

5.5 "Models and Metrics for Software Management and Engineering," V. R. Basili, *ASME Advances in Computer Technology*, January 1980, Vol. 1, 12 pages

This technical paper attempts to characterize several quantitative models and measures of the software development process. These models and measures deal with various aspects of the software process and product, including resource estimation, complexity, reliability, and size. The relationship of these models and measures to the software development life cycle is also discussed. Finally, the extent to which the various models have been applied in production environments and the success they have achieved is indicated. *JAO*

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

5.6 *A Study of the Musa Reliability Model*, SEL-80-005, A. M. Miller, November 1980, 94 pages

This document describes a study in which the Musa reliability model was applied to three software projects developed for GSFC, with the goal of determining whether the model could be used in the flight dynamics environment as a software management tool. One purpose of the model is to predict the total number of errors in a piece of software undergoing testing. Actual times between failures and their associated runtimes were fitted to the Musa equation in an iterative procedure. Of the three projects studied, the results for one converged to a value 25 percent higher than the actual number of errors; the other two did not converge at all.

The document discusses the assumptions underlying the model and evaluates the characteristics of the environment that could affect these assumptions. Suggestions are offered about changes that could be made in the environment to better meet the assumptions. *CASI*

This document was originally prepared as a Master's Thesis at the University of Maryland.

5.7 *An Appraisal of Selected Cost/Resource Estimation Models for Software Systems*, SEL-80-007, J. F. Cook and F. E. McGarry, December 1980, 41 pages

This document presents the results of an evaluation and comparison of seven cost/resource estimation models based on SEL data. The following models were considered:

- Doty
- Walston/Felix
- Tecolote
- GRC
- SLIM
- PRICE S3
- SEL Meta-Model

The validity of the theoretical bases of these models was not analyzed. The objective of the appraisal was simply to determine how well SEL data conformed to the predictions of various models. *NTIS*

This document was also issued as Goddard Space Flight Center document X-582-81-1.

5.8 "A Meta-Model for Software Development Resource Expenditures," J. W. Bailey and V. R. Basili, *Proceedings of the Fifth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1981, 10 pages

This technical paper describes an effort to produce a model of software development resource expenditures that can be generalized to a number of situations. Many models have been proposed over the last several years. However, experience has shown that differences in the data collected, types of projects developed, and environmental factors limit the transportability of these models from one organization to another. This conclusion is reasonable because a model developed in any given environment will reflect only the impact of factors that have a variable effect in that environment. Factors that are constant in that environment (and therefore do not affect productivity) may have different or variable effects in another environment.

This paper describes a model-generation process that permits the development of a resource estimation model for any particular organization. The process provides the capability to produce a model that is tailored to the organization and can be expected to be more effective than any model originally developed for another environment. The model is demonstrated here using data collected by the SEL at GSFC. *JAO*

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

5.9 "Can the Parr Curve Help With Manpower Distribution and Resource Estimation Problems?" V. R. Basili and J. Beane, *Journal of Systems and Software*, February 1981, Vol. 2, No. 1, 11 pages

This technical paper analyzes the resource utilization model developed by Parr. The curve predicted by the model is compared with several other curves, including the Rayleigh curve, a parabola, and a trapezoid, with respect to how well they fit manpower utilization. The evaluation is performed for several flight dynamics projects of the 6- to 12-man-year effort range that were studied by the SEL.

The conclusion drawn is that the Parr curve can be made to fit the data better than the other curves. However, because of the noise in the data, it is difficult to confirm the shape of the manpower distribution from the data alone; therefore it is difficult to validate any particular model. Moreover, since the parameters used in the curve are not easily calculable or estimable from known data, the curve is not effective for resource estimation. *JAO*

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

5.10 *The Rayleigh Curve as a Model for Effort Distribution Over the Life of Medium Scale Software Systems*, SEL-81-012, G. O. Picasso, December 1981, 153 pages

This document discusses some of the factors affecting the accuracy of resource models applied to medium-scale software systems. Putnam has shown that the Rayleigh curve is an adequate model for the life-cycle effort distribution of large-scale systems. Previous investigations of the applicability of this model to medium-scale software development efforts have met with mixed results. The results of the earlier investigations are confirmed in this analysis. The reasons for the failure of the models are found in the subcycle (phase) effort data. There are four contributing factors: uniqueness of the environment studied, the influence of holidays, varying management techniques, and differences in the data studied. *SEB*

This document was also issued as University of Maryland Technical Report TR-1186.

5.11 "Comparison of Regression Modeling Techniques for Resource Estimation," D. N. Card, Computer Sciences Corporation, Technical Memorandum, November 1982, 21 pages

This technical memorandum presents the results of a study conducted to compare three alternative regression procedures by examining the results of their application to one commonly accepted equation for resource estimation. Linear, log-linear, and nonlinear procedures were considered. The memorandum summarizes the data studied, describes the resource estimation equation, explains the regression procedures, and compares the results obtained from the procedures. The regression procedures were evaluated with respect to numerical accuracy, conceptual accuracy, and computational cost. This study is based on data collected from 22 flight dynamics software projects studied by the SEL. *SEB*

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

5.12 "Monitoring Software Development Through Dynamic Variables," C. W. Doerflinger and V. R. Basili, *Proceedings of the Seventh International Computer Software and Applications Conference*. New York: IEEE Computer Society Press, 1983, 30 pages

This paper summarizes the SEL document (SEL-83-106) of the same name. It describes research conducted by the SEL on the use of dynamic variables as a tool for monitoring soft ware development. The intent of the project, which examined several FORTRAN projects with similar profiles, was to identify project-independent measures. The projects developed serve similar functions, and because the projects are similar, some underlying relationships exist that are invariant between the projects. These relationships, once well defined, may be used to compare the development of different projects to determine whether they are evolving in the same way previous projects in this environment evolved. *JAO*

This technical paper also appears in SEL-83-003, *Collected Software Engineering Papers: Volume II*, November 1983, and as SEL-83-006, October 1983.

5.13 *Monitoring Software Development Through Dynamic Variables (Revision 1)*, SEL-83-106, C. W. Doerflinger, November 1989, 116 pages

This document describes research conducted by the SEL on the use of dynamic variables as a tool for monitoring software development. The intent of the project, which examined several FORTRAN projects with similar profiles, was to identify project-independent measures. The projects developed serve similar functions, and because the projects are similar, some underlying relationships exist that are invariant between the projects. These relationships, once well defined, may be used to compare the development of different projects to determine whether they are evolving in the same way previous projects in this environment evolved. *SEB*

The previous version of this document was *Monitoring Software Development Through Dynamic Variables*, SEL-83-006, C. W. Doerflinger, November 1983. This document was originally prepared as a Master's Thesis at the University of Maryland.

5.14 *An Approach to Software Cost Estimation*, SEL-83-001, F. E. McGarry, G. Page, D. N. Card, et al., February 1984, 73 pages

This document describes the general procedures for software cost estimation in any environment. First, the basic concepts of work and effort estimation are explained, some popular resource estimation models are reviewed, and the accuracy of resource estimates is investigated. Next, general guidelines are presented for cost estimation throughout the software life cycle. The sources of information and relevant parameters available during each phase cycle are identified. Finally, a comprehensive software cost prediction procedure based on the experiences of the SEL in the flight dynamics area and incorporating management expertise, cost models, and historical data is provided. The methodology developed incorporates these elements into a customized management tool for software cost prediction. *NTIS*

This document was also issued as Computer Sciences Corporation document CSC/TM-83/6076.

5.15 "Finding Relationships Between Effort and Other Variables in the SEL," V. R. Basili and N. M. Panlilio-Yap, *Proceedings of the Ninth International Computer Software and Applications Conference*. New York: IEEE Computer Society Press, 1985, 7 pages

This study examines the relationship between effort and other variables for 23 SEL projects that were developed for NASA/ GSFC. These variables fell into two categories: those that can be determined in the early stages of project development and may therefore be useful in a baseline equation for predicting effort in future projects, and those that can be used mainly to characterize or evaluate effort requirements and thus enhance our understanding of the software development process in this environment. Some results of the analyses are presented in this paper. *JAO*

This technical paper also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985, and as University of Maryland Technical Report TR-1520, July 1985.

5.16 "Experimentation in Software Engineering," V. R. Basili, R. W. Selby, Jr., and D. H. Hutchens, *IEEE Transactions on Software Engineering*, July 1986, 11 pages

This paper presents a framework for analyzing most of the experimental work performed in software engineering over the past several years. The framework of experimentation consits of four phases of the experimentation process:

- Definition—Motivation, object, purpose, perspective, domain, and scope
- Planning—Design, criteria, and measurement
- Operation—Preparation, execution, and analysis
- Interpretation—Interpretation context, extrapolation, and impact

The paper describes a variety of experiments in the framework and discusses their contribution to the software engineering discipline. Some useful recommendations for the application of the experimental process in software engineering are included. *JAO*

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

5.17 *A Study on Fault Prediction and Reliability Assessment in the SEL Environment*, V. R. Basili and D. Patnaik, TR-1699, University of Maryland, Technical Report, August 1986, 24 pages

This technical report presents an empirical study on fault estimation and prediction, prediction of fault detection and correction effort, and reliability assessment in the SEL environment. Fault estimation using empirical relationships and fault prediction using curve-fitting methods are investigated. Relationships between debugging efforts (fault detection and correction effort) in different test phases are provided, to make an early estimate of future debugging effort. The report concludes with the fault analysis, application of a reliability model, and analysis of a normalized metric for reliability assessment and monitoring during software development. *SEB*

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

5.18 "Resolving the Software Science Anomaly," D. N. Card and W. W. Agresti, *Journal of Systems and Software*, 1987, 6 pages

This study reexamines one basic relationship proposed by the Halstead theory, which appears to provide a comprehensive model of the program construction process: that between estimated and actual program length. The results show that the apparent agreement between these quantities is a mathematic artifact. Analyses of both Halstead's own data and another larger data set confirm this conclusion. Software science has neither a firm theoretical nor an empirical foundation. *JAO*

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

5.19 "Tailoring the Software Process to Project Goals and Environments," V. R. Basili and H. D. Rombach, *Proceedings of the 9th International Conference on Software Engineering*, March 1987, 12 pages

This paper presents a methodology for improving the software process by tailoring it to the specific project goals and environment. This improvement process is aimed at the global software process model as well as methods and tools supporting that model. The basic idea is to use defect profiles to help characterize the environment and evaluate the project goals and the effectiveness of methods and tools in a quantitative way. The improvement process is implemented iteratively by setting project improvement goals, characterizing those goals and the environment, in part, via defect profiles in a quantitative way, choosing methods and tools fitting those characteristics, evaluating the actual behavior of the chosen set of methods and tools, and refining the project goals based on the evaluation results. All these activities require analysis of large amounts of data and therefore require support by an automated tool. Such a tool—Tailoring A Measurement Environment (TAME)—is currently being developed. *JAO*

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

5.20 *Guidelines for Applying the Composite Specification Model (CSM)*, SEL-87-003, W. W. Agresti, June 1987, 37 pages

This document provides guidelines for applying the Composite Specification Model (CSM), an approach to representing software requirements, and for developing each of the three descriptive views of the software:

- The contextual view, using entities and relationships
- The dynamic view, using states and transitions
- The functional view, using data flows and processes

Using CSM results in a software specification document, which is outlined in this document. *CASI*

5.21 *A Meta Information Base for Software Engineering*, L. Mark and H. D. Rombach, TR-1765, University of Maryland, Technical Report, July 1987, 34 pages

This paper proposes a meta model and a graphical notation for specifying software engineering processes and products. This meta model leads to the view that a software engineering information base needs to support the storage and retrieval of process and product descriptions as well as all data related to the executions of process descriptions and the actual instances of product descriptions generated during the course of a software engineering project. A meta schema for information bases is presented that allows the authors to deal with this type of information in a natural way. In addition, software engineering information bases need to be adaptable to changing process and product descriptions based on changing project goals and characteristics of the project environment and the organization. The meta schema of an information base allows for the generation of a customized information base for a given set of processes and products specified according to the software engineering meta model. The idea for this research originated in the TAME project at the University of Maryland aiming at the development of a measurement, feedback, and planning environment. Currently, the authors have implemented a first prototype information base as part of the prototype TAME system customized to the specific needs of the NASA/SEL environment. The schema of this first prototype was defined by hand and is implemented on a relational database system. Developing the idealized information base for software engineering requires more research in the areas of software engineering and databases. In the area of software engineering, the authors need to improve their understanding of the software process and product in order to be able to construct more formal specifications; in the area of databases, they need to develop a database technology for properly mirroring the specific engineering concepts, including self-adaptability. *SEB*

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

5.22 "Generating Customized Software Engineering Information Bases from Software Process and Product Specifications," L. Mark and H. D. Rombach, *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, January 1989, 9 pages

This paper presents the information base oriented part of the "Meta Information Base for Software Engineering" project at the University of Maryland. The idea of this project is to generate customized software engineering information bases from formal specifications of software engineering processes and products. The generator approach acknowledges the fact that software engineering changes not only from environment to environment but also from project to project. If an information base is expected to truly mirror and support a given software engineering project, it needs to be tailorable to the changing characteristics of the software project itself. The generator bases approach suggested by this project seems to be the natural approach to satisfy this important need. This paper discusses how to represent a set of software process and product type specifications in a database and how to use these to automatically generate database support for process executions and product in stances. *JAO*

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

5.23 "Software Process and Product Specifications: A Basis for Generating Customized SE Information Bases," H. D. Rombach and L. Mark, *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, January 1989, 10 pages

This paper presents the software engineering oriented part of the "Meta Information Base for Software Engineering" project at the University of Maryland. The aim of this project is to generate customized software engineering information bases from formal specifications of software engineering processes and products. Systematic improvement of software processes and products, learning about software engineering approaches and reusing software engineering related experience cannot be achieved without having a specification of the objects to be improved. This paper discusses general requirements for software process specification languages, presents a first prototype software process specification language, demonstrates the application of this language, and derives software engineering related requirements for a supporting information base. The actual efforts aimed at implementing these information base requirements are briefly mentioned in the conclusion. *JAO*

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

5.24 *Characterizing Resource Data: A Model for Logical Association of Software Data*, D. R. Jeffery and V. R. Basili, TR-1848, University of Maryland, Technical Report, May 1987, 35 pages

This paper presents a conceptual model of software development resource data. A conceptual model, such as this, is a prerequisite to the development of integrated project sup port environments that aim to assist in the processes of resource estimation, evaluation, and control. The model proposed is a four-dimensional view of resources that can be used for resource estimation, utilization, and review. A process model is presented showing the use of the data model, and instances of the goal, question, metric paradigm are presented to show the applicability of the models to the measurement task. The model is validated by reference to published literature on resource databases, and the implications of the model in these database environments are discussed. *SEB*

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

5.25 *Towards a Comprehensive Framework for Reuse: A Reuse Enabling Software Evolution Environment*, V. Basili and H. Rombach, TR-2158, University of Maryland, Technical Report, December 1988, 23 pages

This paper motivates and outlines the scope of a comprehensive framework for understanding, planning, evaluating, and motivating reuse practices and the necessary research activities. As a first step toward such a framework, a reuse enabling software evolution environment model is introduced, which provides a basis for the effective recording of experience, the generalization and tailoring of experience, the formalization of experience, and the (re-)use of experience. *SEB*

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

5.26 *Maintenance = Reuse-Oriented Software Development*, V. Basili, TR-2244, University of Maryland, Technical Report, May 1989, 12 pages

This paper views maintenance as a reuse process and, in this context, discusses a set of models that can be used to support the maintenance process. It presents a high-level reuse framework that characterizes the object of reuse, the process for adapting that object for its target application, and the reuse object within its target application. Based on this framework, a qualitative comparison is presented of the three maintenance process models, with regard to their strengths and weaknesses, and the circumstances in which they are appropriate. Providing a more systematic, quantitative approach for evaluating the appropriateness of the particular maintenance model, a measurement scheme, based on the reuse framework, is presented in the form of an organized set of questions that need to be answered. A set of reuse enablers is discussed to support the reuse perspective. *SEB*

This technical paper also appears in SEL-89-006, *Collected Software Engineering Papers: Volume VII*, November 1989.

5.27 *Software Development: A Paradigm for the Future*, V. Basili, TR-2263, University of Maryland, Technical Report, June 1989, 29 pages

This paper offers a new paradigm for software development that treats software development as an experimental activity. The paradigm provides built-in mechanisms for learning how to better develop software and for reusing experience in the forms of knowledge, processes, and products. Models and measures are used to aid in characterization, evaluation, and motivation. An organization scheme is proposed for separating the project-specific focus from the organization's learning and reuse focuses of software development. The paper discusses the implications of this approach for corporations, research, and education and presents some research activities currently underway at the University of Maryland that support this approach. *SEB*

This technical paper also appears in SEL-89-006, *Collected Software Engineering Papers: Volume VII*, November 1989.

5.28 *Towards a Comprehensive Framework for Reuse: Model-Based Reuse Characterization Schemes*, V. Basili and H. Rombach, TR-2446, University of Maryland, Technical Report, April 1990, 33 pages

This technical report discusses the key role of reuse in enabling the software industry to achieve the dramatic improvements in productivity and quality required to meet growing demands. Starting from assumptions about both software reuse and the software development process, it develops the characteristics required of successful reuse models. The report examines state-of-the-art reuse characterization schemes to evaluate how well they meet the required characteristics. It then defines a model-based reuse characterization scheme, demonstrates the applicability of the scheme by applying it to three reuse scenarios, and, finally, presents a reuse-oriented software environment model. *SEB*

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

5.29 "Viewing Maintenance as Reuse-Oriented Software Development," V. Basili, *IEEE Software*, January 1990, 7 pages

This technical paper examines software maintenance as reuse-oriented development. It presents several maintenance models, culminating with a full-reuse model. The paper introduces a framework for reuse and compares the maintenance models and their application in the reuse framework. It concludes with a discussion of support mechanisms that enable reuse, including the goal/question/metric improvement paradigm. *JAO*

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

5.30 "Software Reuse: A Key to the Maintenance Problem," H. D. Rombach, *Butterworth Journal of Information and Software Technology*, January/February 1991, 7 pages

This paper describes software maintenance, points out that maintenance is inherently reuse oriented, and identifies some crucial maintenance problems. It then discusses similar problems in the area of software reuse, presents a comprehensive framework that has been proposed to address the reuse problems, and suggests how software maintenance may benefit by adopting a reuse-oriented framework. The paper concludes with an overview of reuse-related research at the University of Maryland. *JAO*

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

5.31 "Support for Comprehensive Reuse," V. R. Basili and H. D. Rombach, *Software Engineering Journal*, September 1991, 14 pages

This paper presents a comprehensive framework for reuse consisting of a reuse model, a model-based characterization scheme, and the Tailoring A Measurement Environment (TAME) model describing the integration of reuse into the software development process. Three hypothetical reuse scenarios—generic Ada packages (product reuse), design inspections (process reuse), and cost models (knowledge reuse)—are used to illustrate the approach and applicability of the proposed model and characterization scheme.

A number of assumptions regarding software development in general and reuse in particular have resulted from more than 15 years of analyzing software processes and products. These assumptions have led to the identification of four requirements essential for any useful reuse model and related characterization scheme. The paper identifies several existing reuse models and characterization schemes and illustrates that they only partially satisfy the requirements. A new reuse model that satisfies all the requirements is introduced and refined to derive a scheme for characterizing reuse candidates, reuse needs, and the reuse process. According to the proposed model, effective reuse requires an environment that supports continuous improvement. The remainder of the paper discusses the reuse-oriented model proposed by the TAME, compares the TAME model with the proposed reuse model, describes mechanisms for effective reuse in the context of the TAME model, and discusses aspects of the TAME research prototypes. *JAO*

This technical paper also appears in SEL-91-005, *Collected Software Engineering Papers: Volume IX*, November 1991, and as University of Maryland Technical Report TR-2606, Februrary 1991.

5.32 "A Reference Architecture for the Component Factory," V. R. Basili, G. Caldiera, and G. Cantone, *ACM Transactions on Software Engineering and Methodology,* January 1992, 28 pages

This paper explores an approach for increasing the quality and productivity of software development through reuse that expands on the traditional reuse of code to the reuse of software objects, their relationships, and associated experience. For reuse to be easy and effective, an organizational structure is needed that is flexible and allows continuous improvement. This paper proposes an organizational framework that separates project-specific activities into a project organization and reuse packaging activities into an experience factory; these two groups interact, but their methods and tools are independent. The experience factory is further divided into subgroups: the domain factory and the component factory. The component factory provides reusable software components (RSCs) to projects upon demand and creates and maintains a repository of those components for future use. An RSC is defined to be not just code, but code packaged with everything necessary for its reuse and maintenance in an application system. The organizational framework is represented by a reference architecture using different levels of abstraction to obtain a flexible and evolutionary organizational design. This paper outlines a reference architecture and discusses the instantiation process, a methodology for deriving a particular environment from the general one. Some theoretical examples and a real case study are presented to illustrate the concept of reference architecture and the instantiation methodology. *JAO*

This technical paper also appears in SEL-91-005, *Collected Software Engineering Papers: Volume IX,* November 1991, and as University of Maryland Technical Report TR-2607, March 1991.

5.33 *A Pattern Recognition Approach for Software Engineering Data Analysis*, L. C. Briand, V. R. Basili, and W. M. Thomas, TR-2672, University of Maryland, Technical Report, May 1991, 37 pages

Managing a large-scale software development project requires the use of quantitative models to provide insight from the historical data of similar projects. This paper identifies how such models can be used to predict, understand, evaluate, and control the software development process, and defines the requirements for an effective data analysis procedure. It discusses the use of statistical analysis in the software engineering field and why the classical approaches do not meet most of the requirements for an effective data analysis procedure. The paper presents a new approach, Optimized Set Reduction (OSR), which is based on pattern recognition techniques tailored to the software engineering field and offers advantages that overcome many of the problems associated with classical statistical analyses. It provides experimental results to demonstrate the effectiveness of the OSR approach for cost estimation modeling, discusses issues related to data analysis disturbances, and shows how an OSR approach might deal with them. The paper also describes the Improvement Paradigm and shows how the OSR performs with the learning and model refinement issues in the Improvement Paradigm framework. Finally, this paper presents eight positive characteristics of OSR that allow prediction, risk assessment, and quality evaluation, and identifies future research directions for work on the OSR approach. *SEB*

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

5.34 *Software Engineering Laboratory (SEL) Cleanroom Process Model*, SEL-91-004, S. Green, November 1991, 74 pages

This document describes the Software Engineering Laboratory (SEL) cleanroom process model. The model is based on data and analysis from previous cleanroom efforts within the SEL and is tailored to serve as a guideline in applying the methodology to future software production efforts. It describes the phases that are part of the process model life cycle from the delivery of requirements to the start of acceptance testing. For each defined phase, a set of specific activities is discussed, and the appropriate data flow is described. This document also presents pertinent managerial issues, key similarities and differences between the SEL's cleanroom process model and the standard development approach used on SEL projects, and significant lessons learned from prior cleanroom projects. It is intended that the process model described in this document will be further tailored as additional SEL cleanroom projects are analyzed. *NTIS*

5.35 "The Software-Cycle Model for Re-Engineering and Reuse," J. W. Bailey and V. R. Basili, *Proceedings of the ACM Tri-Ada 91 Conference*, October 1991, 15 pages

This paper provides examples, using Ada, of techniques for choosing, re-engineering, and recombining components into programs. It describes rudimentary methods for quantifying the effort to extract reusable components from existing programs and the effort to recombine them into new programs. The paper proposes the software-cycle model based on the cycle of software development, use, re-engineering, and reuse. It is a structured model of information identification and reuse that is both feasible and suitable for further development and refinement. To support the process, three styles of software component reuse that are currently being practiced are identified and examined for their adaptability to the model. These styles are layered reuse, tailored reuse, and generated reuse. All three reuse methods are described and examples of each are provided. As an example, a simple electronic mail system, in Ada, is put through transformations to yield components that can be combined using all three reuse methods. The paper concludes with a measurement summary and a discussion of future work. *JAO*

This technical paper also appears in SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992.

5.36 "On the Nature of Bias and Defects in the Software Specification Process," P. A. Straub and M. V. Zelkowitz, *Proceedings of the Sixteenth International Computer Software and Applications Conference (COMPSAC 92)*, September 1992, 8 pages

Bias causes two effects in specifications: They are biased or they are incomplete for fear of bias. A specification is considered biased if it contains extraneous or imposed requirements. This paper presents a framework for multi-attribute specifications and discusses the problem of bias in specifications. It describes a formula to analyze the correctness of a specification with respect to a problem. The paper identifies and characterizes the subprocesses that occur in the software specification and design process. It then introduces an explanatory model of the specification and design process. The model describes the relative and unavoidable nature of bias and distinguishes bias from designed and other requirements in a specification. The model does not lead to any definite method to detect bias. While studying how bias is introduced into a specification, the authors realized that software defects and bias are related issues: they are both manifestations of either misconceptions with respect to the problem or preconceptions with respect to the solution. The authors studied coding fault data collected by the SEL. They concluded that many faults found during the coding phase have roots in the specification process and that implementation bias plays an important role in faults related to changes due to specification issues. *JAO*

This technical paper also appears in SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992.

5.37 "An Improved Classification Tree Analysis of High Cost Modules Based Upon an Axiomatic Definition Of Complexity," J. Tian, A. Porter, And M. V. Zelkowitz, *Proceedings of the Third IEEE International Symposium on Software Reliability Engineering (ISSRE 92)*, October 1992, 9 pages

Identification of high cost modules has been viewed as one mechanism to improve overall system reliability, since such modules tend to produce more than their share of problems. A decision tree model has been used to identify such modules. In this paper, a previously developed axiomatic model of program complexity is merged with the previously developed decision tree process for an improvement in the ability to identify such modules. This improvement has been tested using data from the SEL. *JAO*

This technical paper also appears in SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992.

5.38 "Providing an Empirical Basis for Optimizing the Verification and Testing Phases of Software Development," L. C. Briand, V. R. Basili, and C. J. Hetmanski, *Proceedings of the Third IEEE International Symposium on Software Reliability Engineering (ISSRE 92)*, October 1992, 12 pages

Applying equal testing and verification effort to all parts of a software system is not very efficient, especially when resources are limited and scheduling is tight. It is desirable to be able to differentiate low and high fault density components so that testing and verification effort can be concentrated where needed. This paper presents an alternative approach to the standard statistical techniques for component classification (e.g., logistic regression). It presents the basic principles of the Optimized Set Reduction (OSR) algorithm and a formal definition of the OSR process. It discusses the issue of building models based on partial information and suggests solutions to the problem of partial information within the OSR framework. The paper then presents a process called "pattern merging"; the goal of this process is to facilitate interpretation and learning based on generated models. It discusses pattern interpretation rules and shows how pattern merging facilitates the identification of high-risk components based on metrics obtainable at the end of the coding phase. The paper describes an experiment using OSR and SEL data. The experiment used the OSR technique to build classification models to provide an indication of the fault density of a component. The experiment results indicate that it is possible to build useful models for assessing the fault density of software components. *JAO*

This technical paper also appears in SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992.

5.39 "A Classification Procedure for the Effective Management of Changes During the Maintenance Process," L. C. Briand and V. R. Basili, *Proceedings of the 1992 IEEE Conference on Software Maintenance (CSM 92)*, November 1992, 12 pages

During software operation, maintainers are often faced with numerous change requests. Given available resources such as effort and calendar time, changes have to be planned to fit within budget and schedule constraints. This paper addresses the issue of assessing the difficulty of a change based on known or predictable data. It should be considered a first step toward constructing a predictive effort model for changes during the maintenance phase. The paper proposes a modeling approach, based on regular statistical techniques, that can be used in a variety of software maintenance environments. It involves four steps:

- Identify predictable metrics
- Identify significant predictable metrics
- Generate a classification function
- Validate the model

This approach can be easily automated and is simple to use, even for people with limited statistical experience. Moreover, it deals effectively with the uncertainty usually associated with both model inputs and outputs. The modeling approach is validated on a data set provided by the SEL which shows it has been effective in classifying changes with respect to the effort involved in implementing them. Advantages of the approach are discussed, along with recommendations for improving the data collection process. *JAO*

This technical paper also appears in SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992.

5.40 *Developing Interpretable Models with Optimized Set Reduction for Identifying High Risk Software Components*, L. C. Briand, V. R. Basili, and C. J. Hetmanski, TR-3048, University of Maryland, Technical Report, March 1993, 31 pages

There is a need to identify software components that are likely to produce a large number of faults so that the verification and testing efforts can be concentrated on them, optimizing the reliability of a software system with minimum cost. A modeling process is needed that allows for the reliable classification of high-risk components and aids in the understanding of the software development process, allowing remedial actions and better process decisions in the future. This paper presents the Optimized Set Reduction (OSR) approach for constructing such models. Its approach to classification is to measure the software system and build multivariate stochastic models for predicting high-risk system components. The paper presents experimental results obtained by classifying Ada components into classes: likely or not likely to generate faults during system and acceptance test. The paper concludes that the OSR approach successfully integrates statistical and machine learning approaches in empirical modeling with respect to specific software engineering needs: it supports dealing with partial information and model interpretation and is not based on a severely constraining set of hypotheses. *SEB*

This technical paper also appears in SEL-93-001, *Collected Software Engineering Papers: Volume XI*, November 1993.

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

It is often noted that a small number of software components are responsible for a large number of the problems that arise during software development. Once these "high-risk" components have been identified, the development process can be optimized to reduce risk. A modeling process is needed that will allow for the reliable detection of potential problem areas and for easy interpretation of the cause of problems so that the most appropriate remedial actions can be taken. This paper presents an automated modeling technique, Optimized Set Reduction (OSR), that can be used as an alternative to regression techniques. It then shows how OSR can be used to help identify and interpret the significant trends that characterize high-risk components in several Ada systems. Finally, the paper evaluates the effectiveness of the suggested technique on the basis of a comparison with logistic regression based models. *JAO*

This technical paper also appears in SEL-93-001, *Collected Software Engineering Papers: Volume XI*, November 1993

5.42 "An Information Model for Use in Software Management Estimation and Prediction," N. R. Li and M. V. Zelkowitz, *Proceedings of the Second International Conference on Information and Knowledge Management*, November 1993, 9 pages

It is difficult for software managers to evaluate the status or quality of a software development project and to make correct decisions without accurate, reliable measurement models and data. Lines of code is still the most widely used measure for cost and error analysis, even though it is known to be inaccurate. Because lines of code is not known until the completion of a project, its use as a predictive measure is not reliable; more accurate models of the software development process are needed. The approach in this paper is to use the data collected on a development project to develop dynamic models of software development that reflect the changing nature of the development process. Cluster analysis is a means for determining the underlying information model present in the collected software engineering development data. This paper describes the investigation of the use of cluster analysis within the Software Management Environment (SME), one of the tools developed within the NASA/GSFC Software Engineering Laboratory (SME), one of the tools developed within the NASA/GSFC Software Engineering Laboratory (SEL). It proposes modifications to allow the SME to develop dynamic models for better prediction of attributes of the software development process. *JAO*

This technical paper also appears in SEL-93-001, *Collected Software Engineering Papers: Volume XI*, November 1993

Section 6—Software Measurement

6.1 "Designing a Software Measurement Experiment," V. R. Basili and M. V. Zelkowitz, *Proceedings of the Software Life Cycle Management Workshop*, September 1977, 13 pages

This technical paper explains the research approach employed by the SEL to study the development of actual software development projects. The following types of experiments are performed by the SEL:

- Screening
- Semicontrolled
- Controlled

This paper discusses these experimental designs, potential confounding effects, and the statistical techniques used to evaluate results. The effects on software developers of both learning during the experiment and an awareness of the experimental process itself are examined in detail. Fully controlled experiments are especially difficult to implement in a production environment, but sufficient control is possible to evaluate the effects of software development methodologies. *JAO*

6.2 "Analyzing Medium Scale Software Development," V. R. Basili and M. V. Zelkowitz, *Proceedings of the Third International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1978, 8 pages

This technical paper surveys SEL research activities in software engineering. The collection and analysis of data from software development projects is necessary for the definitive evaluation of software engineering methodologies and techniques. This paper describes the structure of the SEL and some of the early projects that were monitored. It also discusses the application of this data to resource utilization models and reliability studies. The principal contribution of the SEL, as reported in the paper, is the establishment of a facility for collecting the detailed data necessary for these analyses. *JAO*

6.3 "Measuring Software Development Characteristics in the Local Environment," V. R. Basili and M. V. Zelkowitz, *Computers and Structures*, August 1978, Vol. 10, 5 pages

This technical paper discusses the role of data collection in forecasting and monitoring software development projects in a production environment. The specific procedures of the SEL are reviewed, and SEL data collection forms are described. The paper also gives some examples of analyses that can be performed to support managing, understanding, and characterizing software development. The sample analyses identify specific measures for collection. *JAO*

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

6.4 "Evaluating Automatable Measures for Software Development," V. R. Basili and R. Reiter, *Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity, and Cost.* New York: IEEE Computer Society Press, 1979, 10 pages

This technical paper describes an approach to developing and evaluating automatable software measures. The experience of the SEL has shown that software data collection is an expensive activity that can significantly affect the software development process. Costs and effects can be minimized and data quality can be improved by automating the collection of measures wherever possible.

This paper presents a set of automatable measures that were implemented and evaluated in a controlled experiment. The measures include computer job steps, program changes, program size, and software complexity. The results of the experiment indicate that the automated collection of these measures can be implemented effectively in production environments. *JAO*

6.5 "Programming Measurement and Estimation in the Software Engineering Laboratory," V. R. Basili and K. Freburger, *Journal of Systems and Software*, February 1981, Vol. 2, No. 1, 11 pages

This technical paper presents an examination of a set of basic relationships among various software development measures, including size, effort, project duration, staff size, and productivity. Correlations among these measures are computed. The data used come from 15 flight dynamics software development projects studied by the SEL. Certain relationships are derived in the form of equations, and these equations are compared with a set derived by Walston and Felix for IBM Federal Systems Division project data. Logarithmic transformations were performed on the data for some analyses. Although the equations do not have the same coefficients, they are seen to have similar exponents. In fact, the SEL-derived equations tend to be within one standard error of the estimates provided in the IBM equations. *JAO*

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

6.6 "Evaluating and Comparing Software Metrics in the Software Engineering Laboratory," V. R. Basili and T. Phillips, *Proceedings of the ACM Sigmetrics Symposium/Workshop: Quality Metrics*, March 1981, 19 pages

This technical paper describes an effort to identify the best measures of software development effort and software complexity. Four software projects studied by the SEL provide the data for the analysis. The data are screened to ensure their validity. Then estimating equations are derived for effort and errors using the various measures studied in the analysis. Correlations are shown to increase as the reliability of the data increases due to screening. Thus, a procedure is demonstrated for removing noise from the data and making meaningful comparisons of software metrics possible. *JAO*

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

6.7 "Early Estimation of Resource Expenditures and Program Size," D. N. Card, Computer Sciences Corporation, Technical Memorandum, June 1982, 24 pages

This technical memorandum evaluates the suitability of several software measures as estimators of resource expenditures and program size early in the software life cycle. The estimating equation based on the most commonly employed measure, lines of source code, is explained and its limitations are identified. Several alternative measures are investigated and found to give good results. The memorandum also includes computer-generated output of the least-squares regression analyses on which the conclusions are based. *SEB*

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

6.8 *Evaluation of Management Measures of Software Development*, SEL-82-001, G. Page, D. N. Card, and F. E. McGarry, September 1982, Vol. 1: 143 pages, Vol. 2: 379 pages

This two-volume document reports the results of an evaluation of a large set of software development measures relevant to the GSFC environment. The purposes of the analysis were to characterize the current software development process in one environment by identifying important qualities and corresponding measures and to evaluate the effectiveness of specific tools and techniques in this environment. The measures studied were counts, ratios, and management supplied ratings of various elements of the software development process. The measures are high level in that each describes some aspect of an entire software project (or a large part of it) rather than individual components of the project.

Volume 1 explains a conceptual model of software development, the data classification scheme, and the analytic procedures. Factor analysis, cluster analysis, and a test of normality were used. This volume summarizes the results of those analyses and recommends specific software measures for collection and monitoring. Volume 1 also reproduces in full the results of the computer analyses.

Volume 2 presents a detailed description of the data analyzed, including definitions of measures, lists of values, and summary statistics. Although the information contained in Volume 2 was essential to the development of the explanation and summary presented in Volume 1, it is not essential to the understanding of that explanation and summary. However, Volume 2 is useful in its own right as a source of data and a reference for future research. *SEB*

This document was also issued as Computer Sciences Corporation document CSC/TM-82/6063.

6.9 "Metric Analysis and Data Validation Across FORTRAN Projects," V. R. Basili, R. W. Selby, and T. Phillips, *IEEE Transactions on Software Engineering*, November 1983, 43 pages

This technical paper reports the results of an analysis of the relationship of Halstead measures, McCabe complexity measures, and other software measures to software development effort and errors. Effort is defined in terms of staff-hours from the establishing of functional specifications through acceptance testing. Errors are counted discretely and weighed according to effort to correct. The data studied were collected by the SEL in a production environment. Cross-checks of the data indicated a need for large-scale data validation. The strongest correlations were obtained when the modules of individual programmers were considered independently. However, neither Halstead's effort measure, McCabe's cyclomatic complexity measure, nor lines of source code was convincingly more accurate as an estimator than the others. *JAO*

This technical paper also appears in SEL-83-003, *Collected Software Engineering Papers: Volume II*, November 1983, and as University of Maryland Technical Report TR-1228.

6.10 "Measuring Software Technology," W. W. Agresti, F. E. McGarry, D. N. Card, et al., *Program Transformation and Programming Environments*. New York: Springer-Verlag, 1984, 6 pages

This paper summarizes the results of several recent SEL research efforts. The areas of software engineering discussed are programmer productivity, cost models, and technology evaluations. This paper stresses the importance of establishing an organizational memory to provide a reference for evaluating software engineering techniques. The SEL data collection program is outlined as an example of such a mechanism. *JAO*

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

6.11 "Software Errors and Complexity: An Empirical Investigation," V. R. Basili and B. T. Perricone, *Communications of the ACM*, January 1984, 11 pages

This paper reports the results of an analysis of error data obtained from a flight dynamics software project studied by the SEL. The distributions of errors by type and location are identified and discussed. Correlations among module size, complexity, and error rate are then described and evaluated. Modified and new modules are shown to have similar error characteristics. An alternative error classification scheme is developed. Finally, an attempt is made to compare these results with those of other researchers in the field. *JAO*

This technical paper also appears in SEL-83-003, *Collected Software Engineering Papers: Volume II*, November 1983, and as University of Maryland Technical Report TR-1195, August 1982.

6.12 *Measures and Metrics for Software Development*, SEL-83-002, D. N. Card, F. E. McGarry, G. Page, et al., March 1984, 80 pages

This document reports the evaluations of and recommendations for the use of software development measures based on the practical and analytical experience of the SEL. It describes the basic concepts of measurement and a system of classification for measures. The principal classes of measures defined are explicit, analytic, and subjective. Some of the major software measurement schemes appearing in the literature are reviewed. The applications of specific measures in a production environment are explained. These applications include the following:

- Prediction and Planning
- Review and Assessment
- Evaluation and Selection

An appendix describes the use of software development histories to manage ongoing software development projects. *NTIS*

This document was also issued as Computer Sciences Corporation document CSC/TM-83/6061.

6.13 "Characteristics of FORTRAN Modules," D. N. Card, Q. L. Jordan, and V. E. Church, Computer Sciences Corporation, Technical Memorandum, June 1984, 62 pages

This study analyzes the characteristics of a large sample of FORTRAN modules produced by professional programmers. The proper organization and content of a software module are basic concerns of software developers. Although many strategies and standards for software development are in use, few are based on any empirical evidence. The data studied were collected by the SEL and include error reports, staff charges, and source code measures. This study attempts to determine whether structural and quality differences exist among different classes of software and how knowledge of structural characteristics can be used to maximize software quality. *JAO*

6.14 *Structural Coverage of Functional Testing,* V. R. Basili and J. Ramsey, TR-1442, University of Maryland, Technical Report, September 1984, 59 pages

This technical report describes a study directed at understanding and improving the acceptance test process in the NASA/GSFC SEL environment. A large, commercially developed FORTRAN program was modified to produce structural coverage metrics. The modified program was executed on a set of functionally generated acceptance tests and a large sample of operational usage cases. The resulting structural coverage metrics are combined with fault and error data to evaluate structural coverage in the SEL environment.

It is shown that, in this environment, the functionally generated tests seem to be a good approximation of operational use. The relative proportions of the exercised statement subclasses (executable, assignment, CALL, DO, IF, READ, WRITE) change as the structural coverage of the program increases. A method is proposed for determining whether two sets of input data exercise a program in a similar manner.

Evidence is also provided implying that, in this environment, faults revealed in a procedure are independent of the number of times the procedure is executed and that it may be reasonable to use procedure coverage in software models that use statement coverage. Finally, the evidence suggests that it may be possible to use structural coverage to aid the management of the acceptance test process. *SEB*

6.15 *Investigation of Specification Measures for the Software Engineering Laboratory*, SEL-84-003, W. W. Agresti, V. E. Church, and F. E. McGarry, December 1984, 78 pages

This document presents an investigation of requirements specification measures for potential application in the SEL. Eighty-seven candidate measures are defined; sixteen are recommended for use. Most measures are derived from a new representation, the Composite Specification Model (CSM), which is introduced in this document. CSM incorporates functional, entity/relationship, and state machine views of software requirements. The results of extracting the specification measures from the requirements of a real system are described. *NTIS*

6.16 "Criteria for Software Modularization," D. N. Card, G. Page, and F. E. McGarry, *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985, 6 pages

This paper reports an attempt to determine the effectiveness of two widely used criteria for software modularization, strength, and size, in reducing fault rate and development cost. The study was prompted by a central issue in programming practice that involves determining the appropriate size and information content of a software module. Data from 453 FORTRAN modules developed by professional programmers were analyzed. The results indicated that module strength is a good criterion with respect to fault rate, whereas arbitrary module size limitations inhibit programmer productivity. This analysis is a first step toward defining empirically based standards for software modularization. *JAO*

This technical paper also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985.

6.17 "Calculation and Use of an Environment's Characteristic Software Metric Set," V. R. Basili and R. W. Selby, Jr., *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985, 6 pages

This paper presents an approach for customizing a characteristic set of software metrics to an environment, since both cost/quality goals and production environments differ. The approach is applied in the SEL to 49 candidate process and product metrics of 652 modules from six projects (of 51,000 to 112,000 lines). For this particular environment, the method yielded the characteristic metric set (source lines, fault correction effort per executable statement, design effort, code effort, number of I/O parameters, number of versions). The uses examined for a characteristic metric set include forecasting the effort for development, modification, and fault correction of modules based on historical data. *JAO*

This technical paper also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985.

6.18 "Evaluating Software Development by Analysis of Changes: Some Data from the Software Engineering Laboratory," D. M. Weiss and V. R. Basili, *IEEE Transactions on Software Engineering*, February 1985, 12 pages

This paper describes the application of an effective data collection methodology for evaluating software development methodologies to five different software development projects. Results and data from three of the projects are presented. Goals of the data collection included characterizing changes, errors, projects, and programmers; identifying effective error detection and correction techniques; and investigating ripple effects.

The data collected consisted of changes (including error corrections) made to the software after code was written and baselined, but before testing began. Data collection and validation were concurrent with software development. Changes reported were verified by interviews with programmers. Analysis of the data showed patterns that were used in satisfying the goals of the data collection. *JAO*

This technical paper also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985.

6.19 *Measuring Software Design*, SEL-86-005, D. N. Card, W. Agresti, V. Church, et al., November 1986, 45 pages

This paper describes extensive series of studies of software design measures conducted by the SEL. Included are the objectives and results of the studies, the method used to perform the studies, and the problems encountered. The document should be useful to researchers planning similar studies as well as to managers and designers concerned with applying quantitative design measures. *NTIS*

6.20 "A Controlled Experiment on the Impact of Software Structure on Maintainability," H. D. Rombach, *IEEE Transactions on Software Engineering*, March 1987, 11 pages

This paper describes a study on the impact of software structure on maintainability aspects such as comprehensibility, locality, modifiability, and reusability in a distributed system environment. The study was part of a project at the University of Kaiserslautern, Germany, to design and implement LADY, a LAnguage for Distributed sYstems. The study addressed the impact of software structure from two perspectives:

- The language designer's perspective was to evaluate the general impact of the set of structural concepts chosen for LADY on the maintainability of software systems implemented in LADY.
- The language user's perspective was to derive structural criteria (metrics), measurable from LADY systems, that allow the explanation or prediction of the software maintenance behavior.

A controlled maintenance experiment was conducted involving 12 medium-sized distributed software systems; 6 of these systems were implemented in LADY, the other 6 systems were implemented in an extended version of sequential Pascal. The benefits of the structural LADY concepts were judged based on a comparison of the average maintenance behavior of the LADY systems and the Pascal systems; the maintenance metrics were derived by analyzing the interdependence between structure and maintenance behavior of each individual LADY system. *JAO*

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

6.21 *TAME: Integrating Measurement Into Software Environments*, V. R. Basili and H. D. Rombach, TR-1764, University of Maryland, Technical Report, June 1987, 33 pages

This paper describes the TAME project. Based on a dozen years of analyzing software engineering processes and products, we propose a set of software engineering process and measurement principles. These principles lead to the view that an integrated Software Engineering Environment (ISEE) should support multiple process models across the full software life cycle, the technical and management aspects of software engineering, and the planning, construction, and feedback and learning activities. These activities need to be tailored to the specific project under development, and they must be tractable for management control. The tailorability and tractability attributes require the support of a measurement process. The measurement process needs to be top-down, based on operationally defined goals.

The TAME project uses the goal/question/metric paradigm to support this type of measurement paradigm. It provides for the establishment of project-specific goals and corporate goals for planning software provides for the tracing of these goals throughout the life cycle via feedback and post mortem analysis, and offers a mechanism for long-range improvement of all aspects of software development.

The TAME system automates as much of this process as possible by supporting goal development into measurement via models and templates, providing evaluation and analysis of the development and maintenance processes, and creating and using databases of historical data and knowledge bases that incorporate experience from prior projects. *SEB*

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

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

This paper discusses resource utilization over the life cycle of software development and discusses the role that the current "waterfall" model plays in the actual software life cycle. Software production in the NASA environment was analyzed to measure these differences. The results indicate that the waterfall model is not very realistic in practice, and that as technology introduces further perturbations to this model with concepts like executable specifications, rapid prototyping, and wide-spectrum languages, we need to modify our model of this process. *JAO*

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

6.23 "Validating the TAME Resource Data Model," D. R. Jeffery and V. R. Basili, *Proceedings of the Tenth International Conference on Software Engineering*, April 1988, 15 pages

This paper presents a conceptual model of software development resource data and validates the model by reference to the published literature on necessary resource data for development support environments. The conceptual model presented here was developed using a top-down strategy. A resource data model is a prerequisite to the development of integrated project support environments that aim to assist in the processes of resource estimation, evaluation, and control. The model proposed is a four-dimensional view of resources that can be used for resource estimation, utilization, and review. This model is validated by reference to three publications on resource databases, and the implications of the model arising out of these comparisons are discussed. *JAO*

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

6.24 "The TAME Project: Towards Improvement-Oriented Software Environments," V. R. Basili and H. D. Rombach, *IEEE Transactions on Software Engineering*, June 1988, 16 pages

This paper presents and discusses the improvement-oriented software engineering process model underlying the Tailoring A Measurement Environment (TAME) project, its automated support by the TAME system, and the first TAME system prototype. The TAME project at the University of Maryland involves the development of an improvement-oriented software engineering process model that uses the goal/question/metric paradigm to integrate the constructive and analytic aspects of software development. The model provides a mechanism for formalizing the characterization and planning tasks, controlling and improving projects based on quantitative analysis, learning in a deeper and more systematic way about the software process and product, and feeding the appropriate experience back into the current and future projects.

The TAME system is an instantiation of the TAME software engineering process model as an Integrated Software Engineering Environment (ISEE). The first prototype in a series of TAME system prototypes has been developed. An assessment of experience with this first limited prototype is presented, including a reassessment of its initial architecture. The long-term goal of this building effort is to develop a better understanding of appropriate ISEE architectures that optimally support the improvement-oriented TAME software engineering process model. *JAO*

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

6.25 "Measuring Software Design Complexity," D. N. Card and W. W. Agresti, *Journal of Systems and Software*, June 1988, 13 pages

This paper explains a new approach to measuring software design complexity that considers the structure of the overall system as well as the complexity incorporated in individual components. Architectural design complexity derives from two sources: structural (or intermodule) complexity and local (or intramodule) complexity. These complexity attributes can be defined in terms of functions of the number of input/output variables and fanout of the modules comprising the design. A complexity indicator based on these measures showed good agreement with a subjective assessment of design quality but even better agreement with an objective measure of software error rate. Although they are based on a study of only eight medium-scale scientific projects, the data strongly support the value of the proposed complexity measure in this context. Furthermore, graphic representations of the software designs demonstrate structural differences corresponding to the results of the numerical complexity analysis. The proposed complexity indicator seems likely to be a useful tool for evaluating design quality before committing the design to code. *JAO*

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

6.26 *Evaluating Software Development by Analysis of Change Data*, SEL-81-011, D. M. Weiss, November 1981, 272 pages

This document reports the results of an analysis of change data from five different software development projects in two different environments. A common data collection methodology was applied at both GSFC and the Naval Research Laboratory (NRL). This document describes the data collection methodology employed, software projects studied, and the effects of changes on software development.

The results of this study indicate that the data collection methodology is effective and easily extendable to new software development environments. Although the GSFC and NRL environments differed somewhat in their objectives and approach to software development, the software produced by both groups was similar with respect to changes and errors. The results presented in this document include (1) distributions of causes of change, sources of errors, and difficulty of finding errors and (2) tabulations of changes according to number of components changed, changes according to subsystem, difficulty of change (error) according to source of change (error), and source of error according to programmer. *SEB*

6.27 *Evaluating Software Development by Analysis of Changes: The Data from the Software Engineering Laboratory*, SEL-82-008, V. R. Basili and D. M. Weiss, December 1982, 77 pages

This document reports the results of a study for evaluating software development by analyzing changes to the software. The specific goals of the study were to

- Characterize changes and errors
- Characterize projects and programmers
- Identify effective error detection and correction techniques
- Investigate ripple effects in the software caused by changes

The data collected for the report consisted of changes (including error corrections) made to the software after code was written and baselined, but before testing began. Data collection and validation were concurrent with software development. Changes reported were verified by interviews with the originating programmers. Analysis of the data used in the study showed patterns that were used in satisfying the goals of the data collection. (Also see Section 5.3.) *NTIS*

A version of this document was also issued as University of Maryland Technical Report TR-1236.

6.28 "A Summary of Software Measurement Experiences in the Software Engineering Laboratory," J. D. Valett and F. E. McGarry, *Proceedings of the 21st Annual Hawaii International Conference on System Sciences*, January 1988, 9 pages

This paper covers aspects of data collection and software measurement as they have been applied by one particular organization. The measurement results include the experiences and lessons learned through numerous experiments conducted by the SEL on nearly 60 flight dynamics software projects. These experiments have attempted to determine the effect of various software development technologies on overall software project quality and on specific measures such as productivity, reliability, and maintainability. *JAO*

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

6.29 *Establishing a Measurement Based Maintenance Improvement Program: Lessons Learned in the SEL*, H. Rombach and B. Ulery, TR-2252, University of Maryland, Technical Report, May 1989, 27 pages

This paper discusses the use of a goal-oriented approach to measurement to establish a maintenance improvement program within the SEL. Differences are found to exist between the initial phase of the program and its routine application. The approach is demonstrated through concrete examples, and lessons learned in the establishment of a measurement-based, maintenance improvement program are summarized. *SEB*

This technical paper also appears in SEL-89-006, *Collected Software Engineering Papers: Volume VII*, November 1989.

6.30 *Integrating Automated Support for a Software Management Cycle Into the TAME System*, T. Sunazuka and V. Basili, TR-2289, University of Maryland, Technical Report, July 1989, 18 pages

This paper discusses a metric setting procedure based on the Goal/Question/Metric (GQM) paradigm, a management system called the Software Management Cycle (SMC), and its application to a case study based on NASA/SEL data. The Tailoring A Measurement Environment (TAME) methodology, developed at the University of Maryland, is based on the improvement paradigm and the GQM paradigm. The Software Quality Measurement and Assurance Technology (SQMAT), developed at NEC Corporation, is a software quality metric system and methodology applied to the development processes. TAME and SQMAT methodologies are integrated to realize goal-oriented measurement, process control, and visual management. The SMC is a substantiation of these concepts. The paper also describes a method for evaluating the SMC process. The expected effects of SMC are quality improvement, managerial cost reduction, accumulation and reuse of experience, and a highly visual management reporting system. *SEB*

This technical paper also appears in SEL-89-006, *Collected Software Engineering Papers: Volume VII*, November 1989.

6.31 "Design Measurement: Some Lessons Learned," H. Rombach, *IEEE Software*, March 1990, 9 pages

This technical paper presents lessons learned about measurement in general and design measurement in particular. The lessons tend to fall into three categories:

- How measurement must be applied in individual experiments or case studies
- How measurement can help continuously improve an organization's state of the practice
- Why measurement requires automated support

The lessons are illustrated with examples from the Distos/ Incas experiment, the SEL and the goal/question/metric paradigm, and the Tailoring A Measurement Environment (TAME) project.

The paper discusses various design measurements and then proposes a comprehensive design measurement framework. *JAO*

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

6.32 *Software Engineering Laboratory (SEL) Relationships, Models, and Management Rules*, SEL-91-001, W. Decker, R. Hendrick, and J. Valett, February 1991, 93 pages

This document captures over 50 individual SEL research results, extracted from a review of published SEL documentation, that can be applied directly to managing software development projects. Four basic categories of results are defined and discussed: environment profiles, relationships, models, and management rules. In each category, research results are presented on a single page that summarizes the individual result, lists potential uses of the result by managers, and references the original SEL documentation where the result was found. The document serves as a concise reference summary of applicable research for SEL managers. *NTIS*

6.33 "Paradigms for Experimentation and Empirical Studies in Software Engineering," V. R. Basili and R. W. Selby, *Reliability Engineering and System Safety*, January 1991, 21 pages

Measurement techniques and empirical methods must be adopted by software researchers and practitioners to meet the high quality and productivity standards demanded by the complex systems of the future. This paper outlines the following four paradigms for experimentation and empirical study and describes their interrelationships:

- Improvement Paradigm—A long-term, quality-oriented, organizational meta-life-cycle model that includes characterizing the environment, planning, execution, analysis, learning, and feedback. Use of this model by the Tailoring A Measurement Environment (TAME) project is discussed.
- Goal/Question/Metric (GQM) Paradigm—A mechanism for defining and evaluating a set of operational goals using measurement on a specific project.
- Experimentation Framework Paradigm—A refinement of the GQM paradigm that consists of four categories corresponding to phases of the experimentation process: definition, planning, operation, and interpretation.
- Classification Paradigm—A specific instantiation of the improvement paradigm for product assessment involving data management and calibration, classification tree generation, and analysis and feedback.

JAO

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

6.34 "Toward Full Life Cycle Control: Adding Maintenance Measurement to the SEL," H. D. Rombach, B. T. Ulery, and J. D. Valett, *Journal of Systems and Software*, May 1992, 14 pages

Organization-wide measurement of software products and processes is needed to establish full life-cycle control over software products. The SEL started measuring software development more than 15 years ago. Recently, the measurement of maintenance has been added to the scope of the SEL. In this article, the maintenance measurement program is presented as an addition to the already existing and well-established SEL development measurement program and evaluated in terms of its immediate benefits and long-term improvement potential. Immediate benefits of this program for the SEL include an increased understanding of the maintenance domain, the differences and similarities between development and maintenance, and the cause-effect relationships between development and maintenance. Initial results from a sample maintenance study are presented to substantiate these benefits. The long-term potential of this program includes the use of maintenance baselines to better plan and manage future projects and to improve development and maintenance practices for future projects wherever warranted. *JAO*

This technical paper also appears in SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992.

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

It has been demonstrated that system architecture has a heavy impact on maintainability. To be able to predict and assess maintainability early in the development process, it is necessary to be able to measure the high-level design characteristics that affect the change process. This paper presents a comprehensive approach for evaluating the high-level design of software systems that includes the following elements:

- Metrics that are available early in the design process, are based on precisely defined assumptions, and are related without ambiguity to the defined change process model
- Definitions of module cohesion, module coupling, and visibility control consistently based on the notion of interaction, which is closely related to the phenomenon of change side effects
- An Object-Oriented Design (OOD) view of a software module in place of the usual subroutine perspective of coupling and cohesion evaluation

The approach focuses on the change process during acceptance testing and maintenance, and was developed within the well-defined framework of Ada and OOD. However, the paper attempts to separate Ada-specific concepts from language-independent concepts to identify the part of the approach that is reusable for other programming languages. *JAO*

This technical paper also appears in SEL-93-001, *Collected Software Engineering Papers: Volume XI*, November 1993, and as University of Maryland Technical Report TR-3105, July 1993.

Section 7—Technology Evaluations

7.1 *GSFC Software Engineering Research Requirements Analysis Study*, SEL-78-006, P. A. Scheffer and C. E. Velez, November 1978, 26 pages

This document reports the results of a study of the applicability of requirements languages to flight dynamics software development at GSFC. The specific objectives of the study, which are explained in this document, were to

- Determine the impact of requirements language use on software design
- Demonstrate the application of a requirements language on a flight dynamics development problem
- Evaluate the utility of the Multi-Level Expression Design Language – Requirement Level (MEDL-R) in the GSFC environment
- Determine the desirable characteristics of a requirements language tool for use in the GSFC environment

CASI

This document was also issued as a Martin Marietta Corporation Technical Memorandum.

7.2 *Evaluation of the Caine, Farber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Design Environment*, SEL-79-004, C. E. Goorevich, A. L. Green, and W. J. Decker, September 1979, 46 pages

This document reports the results of a study of the usefulness of program design languages (PDLs) for flight dynamics software development at GSFC. The following PDLs were examined and compared:

- Telemetry Computation Branch PDL
- Linger and Mills PDL
- Caine, Farber, and Gordon PDL

The last PDL was selected for intensive study. Its advantages and disadvantages in the flight dynamics environment were evaluated. Appendixes include examples of the use of the Caine, Farber, and Gordon PDL and the processor output. *CASI*

This document was also issued as Computer Sciences Corporation document CSC/TM-79/6263.

7.3 *Multi-Level Expression Design Language–Requirement Level (MEDL-R) System Evaluation*, SEL-80-002, W. J. Decker and C. E. Goorevich, May 1980, 91 pages

This document presents the results of an evaluation of the suitability of the Multi-Level Expression Design Language Requirement Level (MEDL-R) for use in flight dynamics software development at GSFC. The evaluation team studied the MEDL-R concept of requirements languages, the functions performed by MEDL-R, and the MEDL-R language syntax. The document contains recommendations for changes to the MEDL-R system that would make it more useful in the flight dynamics environment. *CASI*

This document was also issued as Computer Sciences Corporation document CSC/TM-80/6093.

7.4 "Use of Cluster Analysis to Evaluate Software Engineering Methodologies," E. Chen and M. V. Zelkowitz, *Proceedings of the Fifth International Conference on Software Engineering.* New York: IEEE Computer Society Press, 1981, 7 pages

This technical paper describes an attempt to identify the characteristic effects of various methodologies on software development. Data collected by the SEL from five software projects were studied. Several objective measures were derived from the data, and their relationships to methodology use were studied with cluster analysis techniques. The analysis showed that the measures reflected the effects of methodologies on software development. *JAO*

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

7.5 "A Software Engineering View of the Flight Dynamics Analysis System (FDAS): Parts I and II," D. N. Card, W. W. Agresti, V. E. Church, and Q. L. Jordan, Computer Sciences Corporation, Technical Memorandum, December 1983 (Part I) and March 1984 (Part II), 58 pages

This report presents the results of an assessment, from the software engineering point of view, of the Flight Dynamics Analysis System (FDAS) at one step in the requirements definition process—a prototype support environment. FDAS is intended to provide an integrated software development support environment for research applications in the areas of orbit, attitude, and mission analysis, and it was conceived to assist users in the preparation, execution, and interpretation of software experiments. A prototype FDAS was constructed to aid in clarifying the requirements for such a system and to test some concepts of language, software structure, and user interface designs. Part I of the report discusses the general approaches to FDAS adapted by the development team. Part II presents a detailed examination of some high-level FDAS design issues and summarizes some similar systems from other environments. *SEB*

7.6 "A Practical Experience with Independent Verification and Validation," G. Page, F. E. McGarry, and D. N. Card, *Proceedings of the Eighth International Computer Software and Applications Conference*. New York: IEEE Computer Society Press, 1984, 5 pages

This paper describes an attempt to assess the benefits and limitations of the application of independent verification and validation (IV&V) in the flight dynamics area at NASA/ GSFC. The SEL applied the IV&V methodology to two medium-sized flight dynamics software development projects. Then, to measure the effectiveness of the IV&V approach, the SEL compared these two projects with two similar past projects, using measures like productivity, reliability, and maintainability. Results indicated that the use of the IV&V methodology did not help the overall process or improve the product in these cases. *JAO*

This technical paper also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985.

7.7 "Analyzing the Test Process Using Structural Coverage," J. Ramsey and V. R. Basili, *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985, 7 pages

This paper reports the results of a study to understand and improve the acceptance test process in the SEL environment. An SEL program, the MAL language preprocessor (a subset of a satellite attitude maintenance system), has been modified to produce structural coverage metrics. It was modified to measure both procedure coverage and statement coverage. Coverage is also computed for several statement subclasses. The modified program was executed on a set of functionally generated acceptance tests and a large sample of operational usage cases. The resulting structural coverage metrics are combined with fault and error data to evaluate structural coverage in the SEL environment.

It is shown that, in this environment, the functionally generated tests seem to be a good approximation of operational use. The relative proportions of the exercised statement subclasses change as the structural coverage of the program increases. A method is proposed for evaluating whether two sets of input data exercise a program in a similar manner. Evidence also shows that (1) faults revealed in a procedure are independent of the number of times the procedure is executed and (2) it may be reasonable to use procedure coverage in software models that use statement coverage. Finally, the evidence suggests that it may be possible to use structural coverage to aid in managing the acceptance test process. *JAO*

This technical paper also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985, and as *Structural Coverage of Functional Testing*, University of Maryland Technical Report TR-1442, September 1984.

7.8 "Measuring the Impact of Computer Resource Quality on the Software Development Process and Product," F. E. McGarry, J. Valett, and D. Hall, *Proceedings of the Hawaii International Conference on System Sciences*, January 1985, 9 pages

This study examined the relationship between computer resources and the software development process and product as exemplified by NASA/GSFC data. Data have been extracted and examined from nearly 50 software development projects varying in size from 3,000 to 130,000 lines of code. All have been related to the support of satellite flight dynamics ground-based computations. As a result of changing situations and technology, the computer support environment has varied widely. Some projects enjoyed fast response time, excess memory, and state-of-the-art tools, whereas others endured slow computer response time, archaic tool support, and limited terminal access to the development machine. Based on the results of this study, a number of computer resource-related implications are provided. *JAO*

This technical paper also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985.

7.9 *Comparison of Software Verification Techniques*, SEL-85-001, D. N. Card, R. W. Selby, F. E. McGarry, et al., April 1985, 90 pages

This document describes a controlled experiment performed by the SEL to compare the effectiveness of code reading, functional testing, and structural testing as software verification techniques. It is one of a series of three experiments organized by R. W. Selby as part of his doctoral dissertation. The experiment results indicate that code reading provides the greatest error detection capability at the lowest cost, whereas structural testing is the least effective technique. This document explains the experiment plan, describes the experiment results, and discusses related results from other studies. It also considers the application of these results to the development of software in the flight dynamics environment. Appendixes summarize the experiment data and list the test programs. A separate Data Supplement contains original materials collected from the participants. *NTIS*

7.10 *Evaluations of Software Technologies: Testing, Cleanroom, and Metrics*, SEL-85-004, R. W. Selby, Jr., May 1985, 183 pages

This document describes a seven-step approach for quantitatively evaluating software technologies, coupling software methodology evaluation with software measurement. The approach is applied in depth in the following three areas:

- Software Testing Strategies—A 74-subject study, including 32 professional programmers and 42 advanced university students, compared code reading, functional testing, and structural testing in a fractional factorial design.
- Cleanroom Software Development—Fifteen 3-person teams separately built a 1200-line message system to compare Cleanroom software development (in which software is developed completely offline) with a more traditional approach.
- Characteristic Software Metric Sets—In the SEL production environment, a study of 65 candidate product and process measures of 652 modules from 6 projects of 51,000 to 112,000 lines yielded a characteristic set of software cost/quality metrics.

SEB

This technical paper was also issued as University of Maryland Technical Report TR-1500, May 1985.

7.11 *Evaluation of an Independent Verification and Validation (IV&V) Methodology for Flight Dynamics*, SEL-81-110, G. Page, F. E. McGarry, and D. N. Card, June 1985, 53 pages

This document describes an experiment in the application of an independent verification and validation (IV&V) methodology to the development of flight dynamics software at GSFC. IV&V is the systematic evaluation of computer software by an organization that is independent of the development organization. IV&V is expected to provide earlier error detection and better quality control over the development process.

This document describes the environment, staffing, and results of the experiment. Costs and error rates are compared with those of similar projects developed without IV&V. An IV&V methodology is found to be appropriate for very large projects and for those with high reliability requirements. *SEB*

The previous version of this document was *Performance and Evaluation of an Independent Software Verification and Integration Process*, SEL-81-010, G. Page and F. E. McGarry, May 1981. This document was also issued as Computer Sciences Corporation document CSC/TM-85/6045.

7.12 "Four Applications of a Software Data Collection and Analysis Methodology," V. R. Basili and R. W. Selby, Jr., *Proceedings of the NATO Advanced Study Institute*, August 1985, 15 pages

This paper presents a seven-step data collection and analysis methodology that couples software technology evaluation with software measurement. Four in-depth applications of the methodology are presented. The four studies represent each of the general categories of analyses on the software product and development process: blocked subject-project studies, replicated project studies, multiproject variation studies, and single project studies. The four applications are in the areas of software testing strategies, Cleanroom software development, characteristic software metric sets, and software error analysis, respectively. *JAO*

This technical paper also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985.

7.13 "Quantitative Evaluation of Software Methodology," V. R. Basili, *Proceedings of the First Pan-Pacific Computer Conference*, September 1985, 21 pages

This paper presents a paradigm for evaluating software development methods and tools. The basic idea is to generate a set of goals that are refined into quantifiable questions. These questions specify the metrics to be collected on the software development and maintenance process and product. The metrics can be used to characterize, evaluate, predict, and motivate. They can be used in an active as well as passive way by learning from analyzing the data and applying what is learned to improving the methods and tools employed in practice. Several examples were given representing each of the different approaches to evaluation. *JAO*

This technical paper also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985, and as University of Maryland Technical Report TR-1519, July 1985.

7.14 "A Software Technology Evaluation Program," D. N. Card, *Annais do XVIII Congresso Nacional de Informatica*, October 1985, 6 pages

This paper describes an ongoing technology evaluation program conducted by the SEL that is intended to resolve certain issues in the application of tools, practices, and techniques by software developers. A wealth of potentially beneficial software engineering tools, practices, and techniques has emerged in the past several years. Simultaneously, realization has grown that all software engineering technologies are not equally effective for all software development problems and environments. The steps to technology improvement include measurement, evaluation, and transference. The SEL collects measures on the production of FORTRAN software for spacecraft navigation systems. Recent SEL investigations demonstrated that the use of structured programming and quality assurance improves software reliability. Also, intensive computer use appears to be associated with low productivity. However, the major factor in both productivity and reliability continues to be personnel capability. Such technology evaluation programs provide an empirical basis for defining software development standards and selecting tools. *JAO*

This technical paper also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985. This material was also presented at the ACM Computer Science Conference, New Orleans, Louisiana, March 1985.

7.15 "An Empirical Study of Software Design Practices," D. N. Card, V. E. Church, and W. W. Agresti, *IEEE Transactions on Software Engineering*, February 1986, 8 pages

This paper reports the results of an empirical study of software design practices in one specific environment. The practices examined affect module size, module strength, data coupling, descendant span, unreferenced variables, and software reuse. Measures characteristic of these practices were extracted from 887 FORTRAN modules developed for five flight dynamics software projects monitored by the Software Engineering Laboratory. The relationship of these measures to cost and fault rate was analyzed using a contingency table procedure. The results show that some recommended design practices, despite their intuitive appeal, are ineffective in this environment, whereas others are very effective. *JAO*

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

7.16 "An Approach for Assessing Software Prototypes," V. E. Church, D. N. Card, W. W. Agresti, and Q. L. Jordan, *ACM Software Engineering Notes*, July 1986, 12 pages

This paper presents a procedure for evaluating a software prototype. The need to assess the prototype itself arises from the use of prototyping to demonstrate the feasibility of a design or development strategy. The assessment procedure can also be of use in deciding whether to evolve a prototype into a complete system. The procedure consists of identifying evaluation criteria, defining alternative design approaches, and ranking the alternatives according to the criteria. *JAO*

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

7.17 "An Evaluation of Expert Systems for Software Engineering Management," C. L. Ramsey and V. R. Basili, *IEEE Transactions on Software Engineering*, June 1989, 12 pages

This report provides an evaluation of the use of expert systems for software engineering management. Although the field of software engineering is relatively new, it can benefit from the use of expert systems. Four prototype expert systems have been developed to aid in software engineering management. Given the values for certain metrics, these systems will provide interpretations that explain any abnormal patterns of these values during the development of a software project. The four expert systems, which solve the same problem, were built using two different approaches to knowledge acquisition, a bottom-up approach and a top-down approach, and two different expert system methods, rule-based deduction and frame-based abduction. A comparison was performed to see which methods best suit the needs of this field. It was found that the bottom-up approach led to better results than did the top-down approach, and the rule-based deduction systems using simple rules provided more complete and correct solutions than did the frame-based abduction systems. *JAO*

This technical paper also appears in SEL-87-009, *Collected Software Engineering Papers: Volume V*, November 1987, and as University of Maryland Technical Report TR-1708, September 1986.

7.18 "The Effectiveness of Software Prototyping: A Case Study," M. V. Zelkowitz, *Proceedings of the 26th Annual Technical Symposium of the Washington, D.C., Chapter of the ACM*, June 1987, 9 pages

This paper discusses resource utilization over the life cycle of software development and discusses the role that the current "waterfall model" plays in the actual software life cycle. The effects of prototyping are measured with respect to the life-cycle model. Software production in the NASA environment was analyzed to measure these differences. The data collected from 13 different projects and 1 prototype development were collected by the SEL and analyzed for similarities and differences. The results indicate that the waterfall model is not very realistic in practice and that a prototype development follows a similar life cycle as a production system. *JAO*

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

7.19 "Evaluating Software Engineering Technologies," D. N. Card, F. E. McGarry, and G. T. Page, *IEEE Transactions on Software Engineering*, July 1987, 6 pages

The objectives of this study were to measure technology use in a production environment, develop a statistical model for evaluating the effectiveness of technologies, and evaluate the effects of some specific technologies on productivity and reliability. A carefully matched sample of 22 projects from the SEL database was studied using an analysis-of-covariance procedure. Limited use of the technologies considered in the analysis produced approximately a 30 percent increase in software reliability. These technologies did not demonstrate any direct effect on development production. *JAO*

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

7.20 "Quantitative Assessment of Maintenance: An Industrial Case Study," H. D. Rombach and V. R. Basili, *Proceedings from the Conference on Software Maintenance*, September 1987, 11 pages

This paper discusses a study aimed at the improvement of measurement and evaluation procedures used in an industrial maintenance environment. The following topics are discussed:

- General measurement, evaluation, and improvement goals important to this environment
- A set of metrics derived for quantifying those goals
- Suggested changes to the current data collection procedures
- Preliminary analysis results based on a limited set of already available data
- Ideas for automating the proposed quantitative assessment approach

This paper emphasizes the steps of introducing such a quantitative maintenance approach into an industrial setting rather than the environment-specific analysis results. The analysis results are intended to demonstrate the practical applicability and feasibility of the proposed methodology for evaluating and improving maintenance aspects in an industrial environment. *JAO*

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

7.21 "Comparing the Effectiveness of Software Testing Strategies," V. R. Basili and R. W. Selby, *IEEE Transactions on Software Engineering*, December 1987, 60 pages

This study applies an experimentation methodology to compare three state-of-the-practice software testing techniques: (1) code reading by stepwise abstraction, (2) functional testing using equivalence partitioning and boundary value analysis, and (3) structural testing using 100 percent statement coverage criteria. The study compares the strategies in three aspects of software testing: fault detection effectiveness, fault detection cost, and classes of faults detected. Thirty-two professional programmers and 42 advanced students applied the 3 techniques to 4 unit-sized programs in a fractional factorial experimental design. The major results of this study are the following.

- With the professional programmers, code reading detected more software faults and had a higher fault detection rate than functional or structural testing did, while functional testing detected more faults than structural testing did, but functional and structural testing were not different in fault detection rate.
- In one advanced student subject group, code reading and functional testing were not different in faults found but were both superior to structural testing while in the other advanced student subject group there was no difference among the techniques.
- With the advanced student subjects, the three techniques were not different in fault detection rate.
- Number of faults observed, fault detection rate, and total effort in detection depended on the type of software tested.
- Functional testing detected more control faults than the other methods did.
- When asked to estimate the percentage of faults detected, code readers gave the most accurate estimates while functional testers gave the least accurate estimates.

JAO

This technical paper also appears in SEL-87-009, *Collected Software Engineering Papers: Volume V*, November 1987, and as University of Maryland Technical Report TR-1501, May 1985.

7.22 "ARROWSMITH-P—A Prototype Expert System for Software Engineering Management," V. R. Basili and C. L. Ramsey, *Proceedings of the IEEE/MITRE Expert Systems in Government Symposium*, October 1985, 13 pages

This paper evaluates two prototype expert systems, collectively named ARROWSMITH-P. ARROWSMITH-P is intended to aid the manager of a software development project in an automated manner. The systems work as follows. First, it is determined whether a software project is following normal development patterns by comparing measures such as programmer hours per line of source code against historical, environment-specific baselines of such measures. The "manifestations" detected by this comparison, such as an abnormally high rate of programmer hours per line of source code, then serve as input to each expert system. Finally, each system attempts to determine the reasons, such as low productivity, for any abnormal software development patterns. These systems can be updated as the environment changes and as more is learned in the field of software engineering.

The two systems, which solve the same problem, were built using different methods: rule-based deduction and frame-based abduction. A comparison was performed to determine which method better suits the needs of this field. It was found that both systems performed moderately well, but the rule-based deduction system using simple rules provided more complete solutions than did the frame-based abduction system. *JAO*

This technical paper also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985.

7.23 "Evolution Towards Specifications Environment: Experience with Syntax Editors," M. Zelkowitz, *Information and Software Technology*, April 1990, 8 pages

This technical paper discusses efforts to extend the integrated environment of Support, a syntax-directed editor, to the specification domain. Syntax-directed editors have been studied over the last 10 years and have not proven as effective as originally thought. This paper analyzes the experience of 3 years of student use of Support. In general, features that are useful to novice programmers tend to interfere with more experienced users.

The paper describes two projects to extend Support into system specification. The first supports axiomatic specifications, while the second supports functional specification. Further work is needed to test the applicability of this form of environment. *JAO*

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

7.24 *The Cleanroom Case Study in the Software Engineering Laboratory: Project Description and Early Analysis*, SEL-90-002, S. Green et al., March 1990, 65 pages

This case study analyzes the application of the Cleanroom software development methodology to the development of production software at NASA/GSFC. The Cleanroom methodology emphasizes human discipline in program verification to produce reliable software products that are "right the first time."

Preliminary analysis of the Cleanroom case study shows that the method can be applied successfully in the GSFC Flight Dynamics Division (FDD) environment and may increase staff productivity and product quality. Compared to typical SEL activities, there is evidence of lower failure rates, a more complete and consistent set of inline code documentation, a different distribution of phase effort activity, and a different growth profile in terms of lines of code developed. *NTIS*

7.25 "Impacts of Object-Oriented Technologies: Seven Years of SEL Studies," M. Stark, *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications*, September 1993, 9 pages

This paper examines the premise that object-oriented technology (OOT) is the most significant technology ever examined by the Software Engineering Laboratory (SEL). It describes the evolution of the use of OOT in the SEL "Experience Factory" in terms of the SEL's original expectations, focusing on how successive generations of projects have used OOT. The SEL concluded that, when coupled with domain analysis, OOT enables high reuse across a range of applications in a given environment. The SEL also concluded that OOT is the first technology that covers the entire development life cycle in the Flight Dynamics Division (FDD). It is a completely new problem-solving paradigm, not simply a new way of performing familiar tasks in a traditional life cycle. However, the use of OOT was not as intuitive as expected, partly because the technique was new to an organization with a mature structured development process. *JAO*

This technical paper also appears in SEL-93-001, *Collected Software Engineering Papers: Volume XI*, November 1993.

Section 8—Ada Technology

8.1 *Ada Training Evaluation and Recommendations from the Gamma Ray Observatory Ada Development Team*, SEL-85-002, R. Murphy and M. Stark, October 1985, 41 pages

This document presents the Ada training experiences of the Gamma Ray Observatory Ada development team and recommendations for future Ada training of software developers. Training methods are evaluated; deficiencies in the training program are noted; and a recommended approach, including course outline, time allocation, and reference materials, is offered. *NTIS*

8.2 "Designing With Ada for Satellite Simulation: A Case Study," W. W. Agresti, V. E. Church, D. N. Card, and P. L. Lo, *Proceedings of the First International Symposium on Ada for the NASA Space Station*, June 1986, 14 pages

This paper compares a FORTRAN-oriented and an Ada-oriented design for the same system to learn whether an essentially different design was produced using Ada. The designs were produced by an experiment that involves the parallel development of software for a spacecraft dynamics simulator. Design differences are identified in the use of abstractions, system structure, and simulator operations. Although the designs were significantly different, this result may be influenced by some special characteristics discussed in the paper. *JAO*

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

8.3 "Towards a General Object-Oriented Software Development Methodology," E. Seidewitz and M. Stark, *Proceedings of the First International Symposium on Ada for the NASA Space Station*, June 1986, 14 pages

This paper presents an overview of a general approach that allows a designer to apply powerful, object-oriented principles to a wide range of applications at all stages of design. Object-oriented design is the technique of using objects (abstract software models of problem domain entities) as the basic units of modularity in system design. This method is discussed in terms of the identification of objects, object diagrams, the principles of abstractions and information hiding, and hierarchies (parent-child and seniority). This paper also considers how object-oriented designs fits into the overall software life cycle. *JAO*

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

8.4 *General Object-Oriented Software Development*, SEL-86-002, E. Seidewitz and M. Stark, August 1986, 79 pages

This report describes a general approach to object-oriented design, which synthesizes the principles of previous object-oriented methods into a unified framework. Further, this approach fits into the overall software life cycle, providing transitions from specification to design and from design to code. It therefore provides the basis for a general object-oriented development methodology. *NTIS*

8.5 "Towards a General Object-Oriented Ada Lifecyle," M. Stark and E. Seidewitz, *Proceedings of the Joint Ada Conference*, March 1987, 10 pages

This paper provides a distillation of our experiences with object-oriented software development. It considers the use of entity-relationship and object data-flow techniques for an object-oriented specification, which leads smoothly into our design and implementation methods as well as an object-oriented approach to reusability in Ada. *JAO*

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

8.6 "A Structure Coverage Tool for Ada™ Software Systems," L. Wu, V. R. Basili, and K. Reed, *Proceedings of the Joint Ada Conference*, March 1987, 9 pages

This paper proposes a number of coverage measures for Ada features such as packages, generic units, and tasks, and discusses their interpretation in relation to the traditional coverage metrics. It also proposes a mechanism for collecting these coverage measures. In addition, this paper suggests that coverage metrics may also be interpreted as indicators of dynamic system performance. *JAO*

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

8.7 "TAME: Tailoring an Ada™ Measurement Environment," V. R. Basili and H. D. Rombach, *Proceedings of the Joint Ada Conference*, March 1987, 7 pages

This paper presents and discusses the Tailoring an Ada Measurement Environment (TAME) project, which aims at the development of a prototype measurement and evaluation environment that supports activities including the process of setting up measurement and evaluation goals and deriving supportive measures. The TAME requirements and architectural design, the status of the first prototype, and the expected impact of this project on Ada projects and Ada Programming Support Environments (APSEs) are discussed. The prototype currently under development does not interface with an APSE; however, it is designed for being integrated into an APSE in the future. *JAO*

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

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

This paper describes a Flight Dynamics Division (at NASA Goddard Space Flight Center) study that analyzes the effects of Ada on the development of their software. This project is one of the first to use Ada in this environment. In the study, two teams are each developing satellite simulators from the same specifications, one in Ada and one in FORTRAN, the standard language in this environment. This paper addresses the lessons learned during the design phase, including the effect of specifications on Ada-oriented design, the importance of the design method chosen, the importance of the documentation style for the chosen design method, and the effects of Ada-oriented design on the software development life cycle. It is hoped that the issues faced in this project will show more clearly what may be expected in designing with Ada-oriented design methods. *JAO*

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

8.9 *Ada® Style Guide (Version 1.1)*, SEL-87-002, E. Seidewitz, May 1987, 90 pages

This document provides guidance on the appropriate use of Ada's powerful features. The Goddard Space Flight Center Ada User's Group has produced this style guide, which addresses "program style" issues. The guide covers three areas of Ada program style:

- Structural decomposition of a program
- Coding and use of specific Ada features
- Textual formatting of a program

NTIS

8.10 *Assessing the Ada® Design Process and Its Implications: A Case Study*, SEL-87-004, C. Brophy and S. Godfrey, July 1987, 45 pages

This document presents the results of a case study to analyze the approach taken and the lessons learned during the design of the Gamma Ray Observatory Dynamics Simulator in Ada (GRODY). Included are recommendations for defining the design phase and outlining the products that should be developed during this phase of the software development life cycle for future flight dynamics software systems developed in Ada. *CASI*

8.11 "ASAP: An Ada Static Source Code Analyzer Program," D. L. Doubleday, University of Maryland, Technical Memorandum, August 1987, 100 pages

This paper describes and provides a user's manual for ASAP, an automated tool for static source code analysis of programs written in the Ada programming language. The purpose of the analysis is to collect and store information pertaining to the analyzed Ada compilation unit's size, complexity, usage of Ada language constructs and features, and static interface with other Ada compilation units. *NTIS*

8.12 "Object-Oriented Programming in Smalltalk and Ada," E. Seidewitz, *Proceedings of the 1987 Conference on Object-Oriented Programming Systems, Languages and Applications*, October 1987, 12 pages

This paper compares the capabilities of modular languages such as Ada and Modula-2 with an archetypal object-oriented language such as Smalltalk. The comparison in this paper is in terms of the basic properties of encapsulation, inheritance, and binding, with examples given in both languages. This comparison highlights the strengths and weaknesses of both types of languages from an object-oriented perspective. It also provides a basis for the application of experience from Smalltalk and other object-oriented languages to increasingly widely used modular languages such as Ada and Modula-2. *JAO*

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

8.13 "Measuring Ada for Software Development in the Software Engineering Laboratory (SEL)," F. E. McGarry and W. W. Agresti, *Proceedings of the 21st Annual Hawaii International Conference on System Sciences*, January 1988, 9 pages

This paper presents a SEL study of the parallel development of a production flight dynamics system by two teams of professional programmers. Both teams worked from the same set of requirements, with one team required to use the normal development process (FORTRAN), while the second team used the Ada development language. Detailed data were collected during the development phases to support the analysis. A discussion of the experimental approach and some of the key results from early, completed studies are presented. *JAO*

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

8.14 "General Object-Oriented Software Development: Background and Experience," E. Seidewitz, *Proceedings of the 21st Hawaii International Conference on System Sciences*, January 1988, 9 pages

The effective use of Ada requires the adoption of modern software engineering techniques such as object-oriented methodologies. A Goddard Space Flight Center Software Engineering Laboratory Ada pilot project has provided an opportunity for studying object-oriented design in Ada. The project involves the development of a simulation system in Ada in parallel with a similar FORTRAN development. As part of the project, the Ada development team trained and evaluated object-oriented and process-oriented design methodologies for Ada. Finding these methodologies limited in various ways, the team created a general object-oriented development methodology which they applied to the project. This paper discusses some background on the development of the methodology, describes the main principles of the approach and presents some experiences with using the methodology, including a general comparison of the Ada and FORTRAN simulator designs. *JAO*

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

8.15 "Lessons Learned in the Implementation Phase of a Large Ada Project," C. E. Brophy, S. Godfrey, W. W. Agresti, and V. R. Basili, *Proceedings of the Washington Ada Technical Conference*, March 1988, 8 pages

This paper discusses lessons learned during the implementation phase of an ongoing Ada project in the Flight Dynamics Division of the NASA/Goddard Space Flight Center. It is part of a series of lessons-learned documents being written for each development phase. Topics that are discussed include (1) use of nesting versus library units, (2) code reading, (3) unit testing, and (4) lessons learned using special Ada features. *JAO*

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

8.16 "General Object-Oriented Software Development with Ada: A Life Cycle Approach," E. Seidewitz, *Proceedings of the Case Technology Conference*, April 1988, 15 pages

This paper discusses the advantages of using object-oriented concepts throughout the entire Ada software life cycle. The information presented here was obtained from an experience with the development of a simulation system in Ada. The paper considers the use of entity-relationship and process/ data flow techniques for an object-oriented specification that leads smoothly into design and implementation methods, as well as an object-oriented approach to reusability in Ada. *JAO*

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

8.17 "Experiences in the Implementation of a Large Ada Project," S. Godfrey and C. Brophy, *Proceedings of the 1988 Washington Ada Symposium*, June 1988, 7 pages

This paper discusses some of the similarities and differences between the implementation of an Ada project and the implementation of a FORTRAN project. During the past several years, the Software Engineering Laboratory (SEL) of Goddard Space Flight Center has been conducting an experiment in Ada to determine the cost effectiveness and feasibility of using Ada to develop flight dynamics software and to assess the effect of Ada on the flight dynamics environment. This experiment consists of near-parallel developments of a dynamics simulator in both FORTRAN and Ada. A study team consisting of members from the SEL has monitored development progress and has collected data on both projects throughout their development. This paper compares the two projects and discusses some of the interesting lessons learned during the code/unit test and integration phases of the Ada project. *JAO*

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

8.18 *System Testing of a Production Ada Project—The GRODY Study*, SEL-88-001, J. Seigle and Y. Shi, November 1988, 26 pages

This paper discusses the impact that the Ada language and design methodologies that utilize its features have on the system test phase of the software development project life cycle. It is one of a series of lessons-learned reports examining each development phase. It evaluates the impact of the use of Ada when compared to FORTRAN in the system test phases of two projects. The paper concludes that although planning for system testing and conducting system tests were not generally affected by the use of Ada, the solving of problems found in system testing was generally facilitated by Ada constructs and design methodology. Most problems found in system testing were not due to difficulty with the language or methodology but rather due to lack of experience with the application. *CASI*

8.19 *Evolution of Ada Technology in the Flight Dynamics Area: Design Phase Analysis*, SEL-88-003, K. Quimby and L. Esker, December 1988, 65 pages

The software engineering issues related to the use of the Ada programming language during the design phase of an Ada project are analyzed. Discussion shows how an evolving understanding of these issues is reflected in the design processes of three "generations" of Ada projects. *CASI*

8.20 *Proceedings of the First NASA Ada User's Symposium*, SEL-88-005, December 1988, 225 pages

This document reproduces the presentations made by participants at the First NASA Ada User's Symposium held in December 1988 at GSFC. The presentations were grouped into the following Ada categories:

- Experiences
- Applications
- Directions and Implications

The document also includes a summary of an open discussion among panelists. The following are the key points discussed in the summary:

- Transition
- Methodology
- Training
- Reuse
- Real Time

CASI

8.21 *Implementation of a Production Ada Project: The GRODY Study*, SEL-89-002, S. Godfrey and C. Brophy, May 1989, 125 pages

The SEL conducted an experiment in parallel development of two flight dynamics systems in FORTRAN and Ada. This document describes the differences observed during the implementation, unit testing, and integration phases of the two projects and outlines the lessons learned during the implementation phase of the Ada development. Included are recommendations for future Ada development projects. *CASI*

8.22 "Evolution of Ada Technology in a Production Software Environment," F. McGarry, L. Esker, and K. Quimby, *Proceedings of the Washington Ada Symposium (WADAS)*, June 1989, 9 pages

The SEL performs studies and measurement related to evolving software technologies. The studies are aimed at understanding both the software development process and the impacts that evolving software practices may have on the software process and product. The SEL has conducted over 65 experiments by applying selected techniques to specific development efforts and measuring the resulting process and product.

This paper analyzes the findings of an effort the SEL initiated in early 1985 to study the characteristics, applications, and impacts of Ada. Beginning with a relatively small practice problem (6,000 source lines of Ada), the SEL has collected detailed development data from a total of eight Ada projects (some of which are ongoing). The projects range in size from 6,000 lines to approximately 160,000 lines of code. *JAO*

This technical paper also appears in SEL-89-006, *Collected Software Engineering Papers: Volume VII*, November 1989.

8.23 "Using Ada to Maximize Verbatim Software Reuse," M. Stark and E. Booth, *Proceedings of Tri-Ada 1989*, October 1989, 13 pages

This paper presents the lessons learned on several simulator projects in the Flight Dynamics Division (FDD) environment that exploit features of the Ada language, such as packages and generics, to achieve verbatim reuse. Verbatim reuse means that no changes are made to the component. These simulators are divided into two separate, but related, problem domains, dynamics simulators, and telemetry simulators. FDD began using Ada in 1985 with the development of the Gamma Ray Observatory Attitude Dynamics Simulator (GRODY). Since that time, six additional simulator projects have been started. With each successive project, a concentrated effort is made to use the lessons learned from previous Ada simulator development projects.

This paper focuses on the concepts used in the projects that have had the most impact on verbatim software reuse in the FDD environment: GRODY, the Upper Atmosphere Research Satellite Telemetry Simulator (UARSTELS), and the Generic Dynamics and Telemetry Simulator (GENSIM). This paper defines underlying design principles, discusses how Ada features support these principles for reuse in the small, and shows how these principles are used to achieve reuse in the large. Finally, this paper presents supporting data from current reusability results. *JAO*

This technical paper also appears in SEL-89-006, *Collected Software Engineering Papers: Volume VII*, November 1989.

8.24 *Evolution of Ada Technology in the Flight Dynamics Area: Implementation/Testing Phase Analysis*, SEL-89-004, K. Quimby, L. Esker, L. Smith, M. Stark, and F. McGarry, November 1989, 100 pages

This document presents an analysis of the software engineering issues related to the use of Ada for the implementation and systems testing phases of four Ada projects developed in the flight dynamics area. These projects reflect an evolving understanding of more effective use of Ada features. In addition, the testing methodology used on these projects has changed substantially from that used on previous FORTRAN projects. *CASI*

8.25 *Lessons Learned in the Transition to Ada from FORTRAN at NASA/Goddard*, SEL-89-005, C. Brophy, November 1989, 90 pages

This document describes a case study performed at GSFC, in which two dynamics satellite simulators were developed from the same requirements, one in Ada and the other in FORTRAN. The purpose of the research was to find out how well the prescriptive Ada development model worked to develop the Ada simulator. The FORTRAN simulator development, as well as past FORTRAN developments, provided a baseline for comparison. Since this was the first simulator developed here, the prescriptive Ada development model had many similarities to the usual FORTRAN development model. However, it was modified to include longer design and shorter testing phases, which is generally expected with Ada developments. This document provides the results of the study as well as recommendations for future Ada project development. *NTIS*

8.26 "Software Reclamation: Improving Post-Development Reusability," J. Bailey and V. Basili, *Proceedings of the Eighth Annual National Conference on Ada Technology*, March 1990, 15 pages

This paper describes part of a multiyear study of software reuse. It explores Ada program transformation techniques that preserve function but result in more independent and therefore presumably more reusable program components. Goals include a precise specification of the transform technique and its application in a large development organization. Expected results are the identification of reuse promoters and inhibiters both in the problem space and in the solution space; the development of a set of metrics that can be applied to both the developing and completed software to reveal the expected degree of reusability; and the development of guidelines for both software developers and reviewers that can help ensure the desired reusability.

The advantages of transforming existing software into reusable components, rather than creating reusable components as an independent activity, include (1) using archives of previous projects that can yield reusable components, thus eliminating risk or development overhead, (2) accomplishing transformation work in parallel with line developments but separately funded (particularly advantageous when developing software for an outside customer unwilling to sustain the additional costs and risks of reusable code development), (3) creating components guaranteed to be relevant to the application area, and (4) ensuring low and controllable cost. *JAO*

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

8.27 "On Designing Parametrized Systems Using Ada," M. Stark, *Proceedings of the Seventh Washington Ada Symposium*, June 1990, 7 pages

This technical paper introduces the concept of a parametrized (reconfigurable) system. The Generic Simulator prototyping project (GENSIM) and planned Combined Operational Mission Planning and Attitude Support System (COMPASS) are discussed as illustrations of reconfigurable systems. A reuse model that supports parametrized systems is introduced, an example system is designed, and the direction of future work is discussed. *JAO*

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

8.28 "PUC: A Functional Specification Language for Ada," P. Straub and M. Zelkowitz, *Proceedings of the Tenth International Conference of the Chilean Computer Science Society*, July 1990, 13 pages

This technical paper presents PUC, a specification language for Ada that addresses programmers' concerns for understandability. PUC is a functional language whose syntax and data types resemble Ada's, although it has features like polymorphism and high order functions. This paper shows the requirements for the language PUC, presents an overview of the language and how it is used in the specification of Ada programs, and gives the requirements and strategies for a semiautomatic translator from PUC to Ada. *JAO*

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

8.29 *Proceedings of the Second NASA Ada Users' Symposium*, SEL-89-008, November 1989, 308 pages

This document reproduces the presentations made by the participants at the Second NASA Ada Users' Symposium held in November 1989 at GSFC. The presentations were grouped into the following Ada categories:

- NASA-Wide Activities
- Center and Project Activities
- Space Station Activities

The document also includes a summary of the presentations and audience remarks. Approximately 380 people, representing 75 private corporations, 3 universities, 17 agencies of the Federal Government, and 7 NASA centers, attended the symposium. *NTIS*

8.30 *A Study of the Portability of an Ada System in the Software Engineering Laboratory (SEL)*, SEL-90-003, L. Jun et al., June 1990, 75 pages

This document discusses a particular porting effort and gives statistics on the portability of Ada and the total effort required to accomplish the rehost, comparing this effort to past experiments on the rehosting of FORTRAN systems. Also included is an analysis of the types of errors encountered during rehosting, the changes required to rehost the system, experiences with the Alsys IBM Ada compiler, the impediments encountered, and the lessons learned during the study. *NTIS*

8.31 *Gamma Ray Observatory Dynamics Simulator in Ada (GRODY) Experiment Summary*, SEL-90-004, T. McDermott et al., September 1990, 74 pages

This document presents the final report on the Gamma Ray Observatory (GRO) Dynamics Simulator in Ada (GRODY) experiment. The experiment consisted of the parallel development of dynamics simulators for GRO, one in FORTRAN and the other in Ada, to evaluate the applicability of Ada in the NASA/GSFC Flight Dynamics environment. Also evaluated by the Ada team were training approaches, an Ada methodology appropriate to the Flight Dynamics environment, and a baseline for evaluating future Ada projects. *CASI*

8.32 "Object-Oriented Programming Through Type Extensions in Ada 9X," E. Seidewitz, *Ada Letters*, March/April 1991, 12 pages

This paper discusses several object-oriented programming features (inheritance, polymorphism, and dynamic binding) and proposes methods to incorporate them into the new Ada language standard. These additions would build on the existing Ada 83 features of typing and encapsulation without violating the current language design philosophy. To support inheritance and dynamic binding, syntax is presented to declare unconstrained types as extendible; extended types would inherit operations from the parent type and the operations on subtypes of extendible types would be dynamically overloaded. The paper proposes syntax to define subtypes so that one single subprogram defined on a parent type could act on all subtypes without requiring overloaded subprograms for each type, thus supporting the feature of polymorphism. To retain the benefits of polymorphism and dynamic binding along with encapsulation, the paper introduces a "private extension" mechanism, which allows private types to be extended subtypes. The author recognizes that other issues exist and must be resolved in a complete design for object-oriented features in Ada, but believes that inclusion of these features would be a start. *JAO*

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

8.33 "An Object-Oriented Approach to Parameterized Software in Ada," E. Seidewitz and M. Stark, *Proceedings of the Eighth Washington Ada Symposium*, June 1991, 15 pages

This paper discusses techniques for specifying and implementing parameterized software systems and the distinctions between parameterized systems and other object-oriented development. It introduces the Combined Operational Mission Planning and Attitude Support System (COMPASS), a parameterized system being developed by the GSFC Flight Dynamics Division (FDD). COMPASS is a large-scale system that covers three related domains. First, the use of domain analysis in the COMPASS specification approach is presented. The specification concepts are strongly object oriented. Objects, classes, categories, and subsystems are defined and discussed. The configuration of domain objects into a specific application is also addressed. A considerable amount of domain analysis has been done on COMPASS using these specification concepts. Second, the Ada implementation concepts planned for COMPASS (including class packages, module packages, and configuration) are presented. To date, these concepts have been applied only to a small amount of prototype code. However, both the specification and implementation concepts are firmly based on experience with Ada, generic simulation components, and object-oriented methodology, and have been tailored specifically for parameterized systems. The authors believe that the concepts applied on COMPASS, though still evolving, can provide a comprehensive approach to specifying and implementing parameterized systems in Ada. *JAO*

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

8.34 *Software Engineering Laboratory (SEL) Ada Performance Study Report*, SEL-91-003, E. W. Booth and M. E. Stark, July 1991, 67 pages

This document presents the goals, methods, and results of the Ada Performance Study. The goals and scope of the study as well as the background of Ada development in the Flight Dynamics Division (FDD) are discussed. The organization, purpose, methods, and results of each test are detailed, and analyses of the test results are provided. The document concludes with guidelines for future development efforts based on the analysis of results from this Ada Performance Study. An explanation of the approach used on the performance tests is included in the appendix. *NTIS*

8.35 "Designing Configurable Software: COMPASS Implementation Concepts," E. W. Booth and M. E. Stark, *Proceedings of Tri-Ada 1991*, October 1991, 12 pages

This paper presents an overview of the Combined Operational Mission Planning and Attitude Support System (COMPASS) and discusses COMPASS implementation concepts, COMPASS prototyping status, and the prototype team. The introduction briefly describes the development history of spacecraft attitude simulators in the GSFC Flight Dynamics Division (FDD) using object-oriented methods and Ada and the dramatic improvements in software reuse achieved by the use of generic architectures. The FDD is applying these successful reuse strategies on COMPASS, the largest Ada software system to be developed by the FDD. The COMPASS implementation concepts are based on the layered software reuse model and the use of Ada generics. This paper describes the COMPASS detailed implementation concepts associated with each of the following three major layers (from the lowest to the highest) and the two levels under each layer:

- Services layer (system-independent and system-dependent levels)
- Problem domain layer (domain language and domain classes levels)
- Application layer (application modules and application architecture components levels)

Each level uses and builds on the functionality provided by lower levels. Levels are not allowed to use or depend on functionality provided at higher levels. The generic unit feature of the Ada language allows parameterization of component dependencies within and between these levels. *JAO*

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

8.36 "Object-Oriented Programming with Mixins in Ada," E. Seidewitz, *Ada Letters*, March/April 1992, 15 pages

This paper presents a fairly natural object-oriented style that can be developed in Ada 83. This style is useful in that it

- provides a useful style for programming object-oriented applications in Ada 83 until new features become available with Ada 9X
- demystifies many of the mechanisms that seem to be "magic" in most object-oriented programming languages by making them explicit
- identifies areas in Ada 83 that are and are not in need of change to make object-oriented programming more natural in Ada 9X

The paper addresses the issues of object-oriented classes, mixins, self-reference, and supertyping. These topics are presented through a series of examples. The paper concludes with a summary highlighting the following areas where the mixin-based style presented is awkward with Ada 83:

- a way to create a subclass type by simple extension of a class type and to parameterize the extension with a mixin
- a simpler way to achieve self-reference during the combination of mixins
- a mechanism for constructing supertypes without having to code dispatch operations explicitly

The author concludes by identifying those features proposed for Ada 9X that would fill the needs identified above. *JAO*

This technical paper also appears in SEL-92-003, *Collected Software Engineering Papers: Volume X*, November 1992.

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

This paper is a summary of the SEL Ada Performance Study, documented in detail in the *Software Engineering Laboratory Ada Performance Study Report*, SEL-91-003. It describes the background of Ada in the Flight Dynamics Division (FDD) of NASA/GSFC and presents the scope of the study. Three study objectives are identified: (1) determine which design and implementation alternatives lead to accelerated run times; (2) determine tradeoffs, if any, made by choosing these alternatives; and (3) develop guidelines to aid future Ada development efforts in the FDD. The paper identifies the two fundamental approaches used to measure the runtime performance of design alternatives and language features. Where the change could be isolated, the runtime improvement of the system was measured after an alternative had been incorporated into a baseline version of the system. Otherwise, the ACM SIGAda Performance Issues Working Group (PIWG) test suite and other tests specific to the flight dynamics environment were used. Ten test groups were developed and each group represented a design or implementation issue. The paper includes a brief description of the purpose of each design test group (scheduling, unconstrained structures, initialization and elaboration, generic units, conditional compilation, and object-oriented programming) and each implementation test group (matrix storage, logical operators, pragma inline, and string-to-enumeration conversion). The paper presents the format used to document each performance test. It concludes with a summary of the impact of the measured performance results and some conclusions of the Ada Performance Study. *JAO*

Section 9—Data Collection

9.1 *A Comparison of RADC and NASA/SEL Software Development Data*, C. Turner and G. Caron, Data & Analysis Center for Software, May 1981, 31 pages

This document reports the results of an analysis of the relationship between project size and several other software measures. These measures include productivity, effort, duration, errors, and error rate. The analysis used data from two sources: Rome Air Development Center (RADC) and the SEL. Least-squares regression techniques were applied to both sets of data. Results obtained from the two sets of data were comparable. The conclusion cited in the report is that RADC and SEL data can be combined, in most cases, to obtain a larger sample without undesirable side effects. *SEB*

9.2 *Automated Collection of Software Engineering Data in the Software Engineering Laboratory (SEL)*, SEL-81-014, A. L. Green, W. J. Decker, and F. E. McGarry, September 1981, 72 pages

This document presents the results of an analysis of SEL data collection procedures. The principal questions addressed are what current manual procedures could be automated and how these automated procedures could be incorporated in the SEL database system. The functional requirements of such a system are identified and explained. The automatable sources of data identified in this report include the following:

- Computer accounting information
- Requirements language tools
- Program design language tools
- Programmer workbench features
- Source code analyzer program

CASI

This document was also issued as Computer Sciences Corporation document CSC/TM-81/6222.

9.3 *Guide to Data Collection*, SEL-81-101, V. E. Church, D. N. Card, and F. E. McGarry, August 1982, 123 pages

This document presents guidelines and recommendations for collecting software development data. The guide describes the motivation, planning, implementation, and management of a data collection effort. Other topics covered include types, sources, and availability of data; methods and costs of data collection; types of analyses supported; and warnings and suggestions based on SEL experience. The appendixes include facsimiles of SEL data collection forms and a glossary of software engineering terms.

This document, abstracted and generalized from 5 years of SEL data collection experience, is intended to be a practical guide for software managers and engineers. *NTIS*

The previous version of this document was *Guide to Data Collection*, SEL-81-001, V. E. Church, D. N. Card, and F. E. McGarry, September 1981. This document was also issued as Computer Sciences Corporation document CSC/TM-82/6137.

9.4 "Data Collection and Evaluation for Experimental Computer Science Research," M. V. Zelkowitz, *Empirical Foundations for Computer and Information Science* (Proceedings), November 1982, 15 pages

This technical paper reviews the data collection procedures of the SEL and shows how they are used to generate data with one particular software engineering experiment. The SEL collects process and product measures from actual scientific software projects using a combination of automated tools and questionnaires. A project currently in progress, from which data are being collected, is a software prototyping effort. The goal of this experiment is to determine the costs and benefits of developing a prototype before developing a full system. *JAO*

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

9.5 *A Methodology for Collecting Valid Software Engineering Data*, V. R. Basili and D. M. Weiss, TR-1235, University of Maryland, Technical Report, December 1982, 22 pages

This technical report describes an effective data collection method for evaluating software methodologies and for studying the software development process. The purpose of the report is to show how to obtain valid data that may be used both to learn more about the software development process and to evaluate software development methodologies in a production environment. The data collected during the study describe changes made to the software during development and are obtained when the changes are made. To ensure accuracy of the data, validation is performed concurrently with software development as part of the data collection process. Validation is based on interviews with the programmers supplying the data. The feasibility of the data collection methodology was demonstrated by applying it to five difference project in two different production environments. *SEB*

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

9.6 "A Methodology for Collecting Valid Software Engineering Data," V. R. Basili and D. M. Weiss, *IEEE Transactions on Software Engineering*, November 1984, 11 pages

This paper describes an effective data collection method for evaluating software development methodologies and studying the software development process. The method uses goal-directed data collection to evaluate methodologies with respect to the claims made for them. Such claims are used as a basis for defining the goals of the data collection, establishing a list of questions of interest to be answered by data analysis, defining a set of data categorization schemes, and designing a data collection form.

The data to be collected are based on the changes made to the software during development and are obtained when the changes are made. To ensure accuracy of the data, validation is performed concurrently with software development and data collection. Validation is based on interviews with those people supplying the data. Results from using the methodology show that data validation is a necessary part of change data collection. Without it, as much as 50 percent of the data may be erroneous.

Feasibility of the data collection methodology was demonstrated by applying it to five different projects in two different environments. The application showed that the methodology was both feasible and useful. *JAO*

This technical paper also appears in SEL-85-003, *Collected Software Engineering Papers: Volume III*, November 1985.

9.7 *Data Collection Procedures for the Software Engineering Laboratory (SEL) Database*, SEL-92-002, G. Heller, J. Valett, and M. Wild, March 1992, 148 pages

This document is a guidebook to collecting software engineering data on software development and maintenance efforts, as practiced in the SEL. It presents an overview of SEL data collection and the types of data the SEL collects. It then presents procedures to be followed on software development and maintenance projects in the Flight Dynamics Division (FDD) of NASA/GSFC for collecting data in support of SEL software engineering research activities. These procedures include detailed instructions for the completion and submission of SEL data collection forms. *NTIS*

This document supersedes *Data Collection Procedures for the Rehosted SEL Database*, SEL-87-008, G. Heller, October 1987.

9.8 *Software Engineering Laboratory (SEL) Database Organization and User's Guide (Revision 2)*, SEL-89-201, L. Morusiewicz, J. Bristow, et al., October 1992, 270 pages

This document presents the organization of the SEL database. Included are definitions and detailed descriptions of the database tables and views, the SEL data, and system support data. The mapping from the SEL and system support data to the base tables is described. In addition, techniques for accessing the database through the Database Access Manager for the SEL (DAMSEL) system and via the ORACLE structured query language (SQL) are discussed. *CASI*

The previous versions of this document were *SEL Data Base Organization and User's Guide*, SEL-89-001, M. So et al., April 1989, and *Software Engineering Laboratory (SEL) Database Organization and User's Guide (Revision 1)*, SEL-89-101, M. So et al., February 1990.

9.9 *Database Access Manager for the Software Engineering Laboratory (DAMSEL) User's Guide*, SEL-90-001, M. Buhler et al., March 1990, 338 pages

This document presents step-by-step operational instructions for the Database Access Manager for the Software Engineering Laboratory (DAMSEL) system. It includes detailed instructions for performing various data entry and report generation activities with sample sessions showing the user interface display screens and output for each of the reports. In addition, it groups the available functions by the classes of users that may access them. *CASI*

Appendix A—Other References

The following is a list of the references that are no longer included in the Annotated Bibliography of SEL Literature.

"The Nature, Organization, Measurement, and Management of Software Complexity," R. W. Reiter (paper prepared for the University of Maryland, December 1976)—Paper not available

The Software Engineering Laboratory, SEL-77-001, V. R. Basili, M. V. Zelkowitz, F. E. McGarry, et al., May 1977—Superseded by *The Software Engineering Laboratory*, SEL-81-104, D. N. Card, F. E. McGarry, G. Page, et al., February 1982

"GSFC NAVPAK Design Higher Order Languages Study: Addendum," P. A. Scheffer and C. E. Velez, Martin Marietta Corporation, Technical Memorandum, September 1977—Document outdated

"Operational Aspects of a Software Measurement Facility," M. V. Zelkowitz and V. R. Basili, *Proceedings of the Software Life Cycle Management Workshop*, September 1977—Document outdated

Structured FORTRAN Preprocessor (SFORT), SEL-77-003, B. Chu and D. S. Wilson, September 1977—Software no longer supported

GSFC NAVPAK Design Specification Languages Study, SEL-77-004, P. A. Scheffer and C. E. Velez, October 1977—Document outdated

"Software Engineering Course Evaluation," G. Page, Computer Sciences Corporation, Technical Memorandum, December 1977—Evaluated courses no longer current

FORTRAN Static Source Code Analyzer Design and Module Descriptions, SEL-78-001, E. M. O'Neill, S. R. Waligora, and C. E. Goorevich, February 1978—Superseded by *FORTRAN Static Source Code Analyzer Program (SAP) System Description (Revision 1)*, SEL-82-102, W. A. Taylor and W. J. Decker, April 1985

"Concepts Used in the Change Report Form," F. Parr and D. Weiss, NASA Goddard Space Flight Center, Technical Memorandum, May 1978—Change Report Form superseded by several new versions

Evaluation of Draper NAVPAK Software Design, SEL-78-003, K. Tasaki and F. E. McGarry, June 1978—Document outdated

Structured FORTRAN Preprocessor (SFORT) PDP-11/70 User's Guide, SEL-78-004, B. Chu and D. S. Wilson, September 1978—Software no longer supported

"A Child's Garden of Complexity Measures," S. F. Lange (paper prepared for the University of Maryland, December 1978)—Topic as applied to the SEL is covered in *Metric*

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

"A Model of the Software Life Cycle," K. Freburger (paper prepared for the University of Maryland, December 1978)—Topic as applied to the SEL is covered in *The Rayleigh Curve as a Model for Effort Distribution Over the Life of Medium Scale Software Systems*, SEL-81-012, G. O. Picasso, December 1981

"A Survey of Several Reliability Models," A. M. Miller (paper prepared for the University of Maryland, December 1978)—Paper not available

"Some Tests of Halstead Measures," G. Hilstop (paper prepared for the University of Maryland, December 1978)—Paper not available

"Error and Change Analysis," D. M. Weiss, Naval Research Laboratory, Technical Memorandum, July 1979—Paper not available

"Resource Model Testing and Information," I. M. Williamson, Naval Research Laboratory, Technical Memorandum, July 1979—Models no longer available

SIMPL-D Data Base Reference Manual, SEL-79-001, M. V. Zelkowitz, July 1979—Technology no longer recommended

Common Software Module Repository (CSMR) System Description and User's Guide, SEL-79-003, C. E. Goorevich, A. L. Green, and S. R. Waligora, August 1979—Software no longer supported

"Software Engineering Laboratory (SEL) Relationships for Programming Measurement and Estimation," V. R. Basili, University of Maryland, Technical Memorandum, October 1979—Topic covered in *An Approach to Software Cost Estimation*, SEL-83-001, F. E. McGarry, G. Page, D. N. Card, et al., February 1984

Functional Requirements/Specifications for Code 580 Configuration Analysis Tool (CAT), SEL-80-001, F. K. Banks, A. L. Green, and C. E. Goorevich, February 1980—Software superseded

"Configuration Analysis Tool (CAT) Design," F. K. Banks, Computer Sciences Corporation, Technical Memorandum, March 1980—Software superseded

NASA Software Research and Technology Workshop (Proceedings), National Aeronautics and Space Administration, March 1980—Document outdated

Multimission Modular Spacecraft Ground Support Software System (MMS/GSSS) State-of-the-Art Computer Systems/Compatibility Study, SEL-80-003, T. Welden, M. McClellan, and P. Liebertz, May 1980—Document outdated

Cost and Reliability Estimation Models (CAREM) User's Guide, SEL-81-008, J. F. Cook and E. Edwards, February 1981—Software no longer supported

Software Engineering Laboratory Programmer Workbench Phase 1 Evaluation, SEL-81-009, W. J. Decker and F. E. McGarry, March 1981—Document outdated

NASA/SEL Data Compendium, C. Turner, G. Caron, and G. Brement, Data & Analysis Center for Software, Special Publication, May 1981—Document outdated

Software Engineering Laboratory (SEL) Compendium of Tools (Revision 1), SEL-81-107, W. Decker, W. Taylor, E. Smith, et. al., February 1982—Tools no longer supported

Configuration Analysis Tool (CAT) System Description and User's Guide (Revision 1), SEL-80-104, W. Decker and W. Taylor, December 1982—Software superseded

"Technical Summary 1982: Report to the National Aeronautics and Space Administration," V. R. Basili—Document outdated

"Analysis Software Requirements for the Data Retrieval System," D. N. Card and V. E. Church, Computer Sciences Corporation, Technical Memorandum, March 1983—Software superseded

Software Engineering Laboratory (SEL) Data Base Organization and User's Guide (Revision 1), SEL-81-102, P. Lo and D. Wyckoff, July 1983—Software superseded

Software Engineering Laboratory (SEL) Data Base Reporting Software User's Guide and System Description, Vol. 1 and Vol. 2, SEL-82-003, P. Lo, S. Eslinger, et al., August 1983—Software superseded

Definition of Specification Measures for the Software Engineering Laboratory (SEL), CSC/TM-84/6085, W. W. Agresti, Computer Sciences Corporation, Technical Memorandum, June 1984—Superseded by *Investigation of Specification Measures for the Software Engineering Laboratory*, SEL-84-003, W. W. Agresti, V. E. Church, and F. E. McGarry, December 1984

Software Engineering Laboratory (SEL) Data Base Maintenance System (DBAM) User's Guide and System Description, SEL-81-203, P. Lo, D. Card, et al., June 1984—Software superseded

Software Engineering Laboratory (SEL) Data Base Retrieval System (DARES) System Description, SEL-83-105, P. Lo, W. Decker, and W. Miller, August 1984—Software superseded

Software Engineering Laboratory (SEL) Data Base Retrieval System (DARES) User's Guide, SEL-83-104, T. Babst, W. Decker, P. Lo, et al., September 1984—Software superseded

Configuration Management and Control: Policies and Procedures, SEL-84-002, Q. L. Jordan and E. Edwards, December 1984—Superseded by *Product Assurance Policies and Procedures for Flight Dynamics Soft ware Development*, SEL-87-001, S. Perry et al., March 1987

Software Engineering Laboratory (SEL) Document Library (DOCLIB) System Description and User's Guide (Revision 1), SEL-81-106, W. Taylor, W. Decker, et al., March 1985—Software superseded

Data Collection Procedures for the Rehosted SEL Database, SEL-87-008, G. Heller, October 1987—Superseded by *Data Collection Procedures for the Software Engineering Laboratory (SEL) Database*, SEL-92-002, G. Heller, J. Valett, and M. Wild, March 1992

Index of Authors

Author	Year	Section	Title
Agresti, W. W. (Cont'd)	1986	7.15	"An Empirical Study of Software Design Practices," D. N. Card, V. E. Church, and W. W. Agresti, *IEEE Transactions on Software Engineering*, February 1986
	1986	7.16	"An Approach for Assessing Software Prototypes," V. E. Church, D. N. Card, W. W. Agresti, and Q. L. Jordan, *ACM Software Engineering Notes*, July 1986
	1986	8.2	"Designing With Ada for Satellite Simulation: A Case Study," W. W. Agresti, V. E. Church, D. N. Card, and P. L. Lo, *Proceedings of the First International Symposium on Ada for the NASA Space Station*, June 1986
	1987	5.18	"Resolving the Software Science Anomaly," D. N. Card and W. W. Agresti, *Journal of Systems and Software*, 1987
	1987	5.20	*Guidelines for Applying the Composite Specification Mode (CSM)*, SEL-87-003, W. W. Agresti, June 1987
	1987	8.8	"Lessons Learned in Use of Ada™-Oriented Design Methods," C. E. Brophy, W. W. Agresti, and V. R. Basili, *Proceedings of the Joint Ada Conference*, March 1987
	1988	6.25	"Measuring Software Design Complexity," D. N. Card and W. W. Agresti, *Journal of Systems and Software*, June 1988
	1988	8.13	"Measuring Ada for Software Development in the Software Engineering Laboratory (SEL)," F. E. McGarry and W. W. Agresti, *Proceedings of the 21st Annual Hawaii International Conference on System Sciences*, January 1988
	1988	8.15	"Lessons Learned in the Implementation Phase of a Large Ada Project," C. E. Brophy, S. Godfrey, W. W. Agresti, and V. R. Basili, *Proceedings of the Washington Ada Technical Conference*, March 1988
Antle, C.	1985	3.4	*Software Verification and Testing*, SEL-85-005, D. N. Card, C. Antle, and E. Edwards, December 1985

Author	Year	Section	Title
Babst, T. A.	1983	2.32	*Glossary of Software Engineering Laboratory Terms*, SEL-82-105, T. A. Babst, M. G. Rohleder, and F. E. McGarry, November 1983
Bailey, J. W.	1980	2.16	"Measuring the Effects of Specific Software Methodologies Within the SEL," V. Basili and J. Bailey, *Proceedings from the Fifth Annual Software Engineering Workshop*, SEL-80-006, November 1980
	1981	5.8	"A Meta-Model for Software Development Resource Expenditures," J. W. Bailey and V. R. Basili, *Proceedings of the Fifth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1981
	1990	8.26	"Software Reclamation: Improving Post-Development Reusability," J. Bailey and V. Basili, *Proceedings of the Eighth Annual National Conference on Ada Technology*, March 1990
	1991	5.35	"The Software-Cycle Model for Re-Engineering and Reuse," J. W. Bailey and V. R. Basili, *Proceedings of the ACN Tri-Ada 91 Conference*, October 1991
Basili, V. R.	1976	2.12	"Program Design Languages," V. Basili, *Proceedings from the First Summer Software Engineering Workshop*, SEL-76-001, August 1976
	1977	2.13	"Overview of the Software Engineering Laboratory," V. R. Basili and M. V. Zelkowitz, *Proceedings from the Second Summer Software Engineering Workshop*, SEL-77-002, September 1977
	1977	2.29	"The Software Engineering Laboratory: Objectives," V. R. Basili and M. V. Zelkowitz, *Proceedings of the Fifteenth Annual Conference on Computer Personnel Research*, August 1977
	1977	6.1	"Designing a Software Measurement Experiment," V. R. Basili and M. V. Zelkowitz, *Proceedings of the Software Life Cycle Management Workshop*, September 1977
	1978	2.14	"The Software Engineering Laboratory—1978," V. R. Basili and M. V. Zelkowitz, *Proceedings from the Third Summer Software Engineering Workshop*, SEL-78-005, September 1978

Author	Year	Section	Title
Basili, V. R. (Cont'd)	1978	2.30	"Operation of the Software Engineering Laboratory," V. R. Basili and M. V. Zelkowitz, *Proceedings of the Second Software Life Cycle Management Workshop*, August 1978
	1978	6.2	"Analyzing Medium Scale Software Development," V. R. Basili and M. V. Zelkowitz, *Proceedings of the Third International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1978
	1978	6.3	"Measuring Software Development Characteristics in the Local Environment," V. R. Basili and M. V. Zelkowitz, *Computers and Structures*, August 1978, Vol. 10
	1979	2.15	"Investigations Into Software Development in the Software Engineering Laboratory," V. Basili, *Proceedings from the Fourth Summer Software Engineering Workshop*, SEL-79-005, November 1979
	1979	5.3	*The Software Engineering Laboratory: Relationship Equations*, SEL-79-002, K. Freburger and V. R. Basili, May 1979
	1979	6.4	"Evaluating Automatable Measures for Software Development," V. R. Basili and R. Reiter, *Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity, and Cost.* New York: IEEE Computer Society Press, 1979
	1980	2.16	"Measuring the Effects of Specific Software Methodologies Within the SEL," V. Basili and J. Bailey, *Proceedings from the Fifth Annual Software Engineering Workshop*, SEL-80-006, November 1980
	1980	5.4	*Tutorial on Models and Metrics for Software Management and Engineering*, SEL-80-008, V. R. Basili, 1980
	1980	5.5	"Models and Metrics for Software Management and Engineering," V. R. Basili, *ASME Advances in Computer Technology*, January 1980, Vol. 1
	1981	2.17	"Assessment of Software Measures in the Software Engineering Laboratory," V. R. Basili, *Proceedings of the Sixth Annual Software Engineering Workshop*, SEL-81-013, December 1981

Author	Year	Section	Title
Basili, V. R. (Cont'd)	1981	5.8	"A Meta-Model for Software Development Resource Expenditures," J. W. Bailey and V. R. Basili, *Proceedings of the Fifth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1981
	1981	5.9	"Can the Parr Curve Help With Manpower Distribution and Resource Estimation Problems?" V. R. Basili and J. Beane, *Journal of Systems and Software*, February 1981, Vol. 2, No. 1
	1981	6.5	"Programming Measurement and Estimation in the Software Engineering Laboratory," V. R. Basili and K. Freburger, *Journal of Systems and Software*, February 1981, Vol. 2, No. 1
	1981	6.6	"Evaluating and Comparing Software Metrics in the Software Engineering Laboratory," V. R. Basili and T. Phillips, *Proceedings of the ACM Sigmetrics Symposium/Workshop: Quality Metrics*, March 1981
	1982	2.18	"Software Errors and Complexity: An Empirical Investigation," V. R. Basili and B. T. Perricone, *Proceedings of the Seventh Annual Software Engineering Workshop*, SEL-82-007, December 1982
	1982	6.27	*Evaluating Software Development by Analysis of Changes: The Data From the Software Engineering Laboratory*, SEL-82-008, V. R. Basili and D. M. Weiss, December 1982
	1982	9.5	*A Methodology for Collecting Valid Software Engineering Data*, V. R. Basili and D. M. Weiss, University of Maryland, Technical Report, December 1982
	1983	2.19	"Monitoring Software Development Through Dynamic Variables," C. W. Doerflinger and V. R. Basili, *Proceedings of the Eighth Annual Software Engineering Workshop*, SEL-83-007, November 1983

Author	Year	Section	Title
Basili, V. R. (Cont'd)	1983	5.12	"Monitoring Software Development Through Dynamic Variables," C. W. Doerflinger and V. R. Basili, *Proceedings of the Seventh International Computer Software and Applications Conference*. New York: IEEE Computer Society Press, 1983
	1983	6.9	"Metric Analysis and Data Validation Across FORTRAN Projects," V. R. Basili, R. W. Selby, and T. Phillips, *IEEE Transactions on Software Engineering*, November 1983
	1984	2.20	"Evaluating Software Testing Strategies," R. W. Selby, Jr., V. R. Basili, J. Page, and F. E. McGarry, *Proceedings of the Ninth Annual Software Engineering Workshop*, SEL-84-004, November 1984
	1984	2.20	"Software Development in Ada," V. R. Basili et al., *Proceedings of the Ninth Annual Software Engineering Workshop*, SEL-84-004, November 1984
	1984	6.11	"Software Errors and Complexity: An Empirical Investigation," V. R. Basili and B. T. Perricone, *Communications of the ACM*, January 1984
	1984	6.14	*Structural Coverage of Functional Testing*, TR-1442, V. R. Basili and J. Ramsey, University of Maryland, Technical Report, September 1984
	1984	9.6	"A Methodology for Collecting Valid Software Engineering Data," V. R. Basili and D. M. Weiss, *IEEE Transactions on Software Engineering*, November 1984
	1985	2.21	"Can We Measure Software Technology; Lessons from 8 Years of Trying," V. R. Basili, *Proceedings of the Tenth Annual Software Engineering Workshop*, SEL-85-006, December 1985
	1985	5.15	"Finding Relationships Between Effort and Other Variables in the SEL," V. R. Basili and N. M. Panlilio-Yap, *Proceedings of the Ninth International Computer Software and Applications Conference*. New York: IEEE Computer Society Press, 1985

Author	Year	Section	Title
Basili, V. R. (Cont'd)	1985	6.17	"Calculation and Use of an Environment's Characteristic Software Metric Set," V. R. Basili and R. W. Selby, Jr., *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985
	1985	6.18	"Evaluating Software Development by Analysis of Changes: Some Data From the Software Engineering Laboratory," D. M. Weiss and V. R. Basili, *IEEE Transactions on Software Engineering*, February 1985
	1985	7.7	"Analyzing the Test Process Using Structural Coverage," J. Ramsey and V. R. Basili, *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985
	1985	7.12	"Four Applications of a Software Data Collection and Analysis Methodology," V. R. Basili and R. W. Selby, Jr., *Proceedings of the NATO Advanced Study Institute*, August 1985
	1985	7.13	"Quantitative Evaluation of Software Methodology," V. R. Basili, *Proceedings of the First Pan-Pacific Computer Conference*, September 1985
	1985	7.22	"ARROWSMITH-P—A Prototype Expert System for Software Engineering Management," V. R. Basili and C. L. Ramsey, *Proceedings of the IEEE/MITRE Expert Systems in Government Symposium*, October 1985
	1986	2.21	"TAME: Tailoring A Measurement Environment," V. R. Basili and H. D. Rombach, *Proceedings of the Eleventh Annual Software Engineering Workshop*, SEL-86-006, December 1986
	1986	5.16	"Experimentation in Software Engineering," V. R. Basili, R. W. Selby, Jr., and D. H. Hutchens, *IEEE Transactions on Software Engineering*, July 1986
	1986	5.17	*A Study on Fault Prediction and Reliability Assessment in the SEL Environment*, TR-1699, V. R. Basili and D. Patnaik, University of Maryland, Technical Report, August 1986

Author	Year	Section	Title
Basili, V. R. (Cont'd)	1987	5.19	"Tailoring the Software Process to Project Goals and Environments," V. R. Basili and H. D. Rombach, *Proceedings of the 9th International Conference on Software Engineering*, March 1987
	1987	5.24	*Characterizing Resource Data: A Model for Logical Association of Software Data*, D. R. Jeffery and V. R. Basili, TR-1848, University of Maryland, Technical Report, May 1987
	1987	6.21	*TAME: Integrating Measurement Into Software Environments*, V. R. Basili and H. D. Rombach, TR-1764, University of Maryland, Technical Report, June 1987
	1987	7.20	"Quantitative Assessment of Maintenance: An Industrial Case Study," H. D. Rombach and V. R. Basili, *Proceedings from the Conference on Software Maintenance*, September 1987
	1987	7.21	"Comparing the Effectiveness of Software Testing Strategies," V. R. Basili and R. W. Selby, *IEEE Transactions on Software Engineering*, December 1987
	1987	8.6	"A Structure Coverage Tool for Ada™ Software Systems," L. Wu, V. R. Basili, and K. Reed, *Proceedings of the Joint Ada Conference*, March 1987
	1987	8.7	"TAME: Tailoring an Ada™ Measurement Environment," V. R. Basili and H. D. Rombach, *Proceedings of the Joint Ada Conference*, March 1987
	1987	8.8	"Lessons Learned in Use of Ada™-Oriented Design Methods," C. E. Brophy, W. W. Agresti, and V. R. Basili, *Proceedings of the Joint Ada Conference*, March 1987
	1988	2.24	"Towards a Comprehensive Framework for Reuse: A Reuse-Enabling Software Evolution Environment," V. R. Basili and H. D. Rombach, *Proceedings of the Thirteenth Annual Software Engineering Workshop, SEL-88-004, November 1988*

Author	Year	Section	Title
Basili, V. R. (Cont'd)	1988	5.25	*Towards a Comprehensive Framework for Reuse: A Reuse-Enabling Software Evolution Environment*, V. Basili and H. Rombach, TR-2158, University of Maryland, Technical Report, December 1988
	1988	6.23	"Validating the TAME Resource Data Model," D. R. Jeffery and V. R. Basili, *Proceedings of the 10th International Conference on Software Engineering*, April 1988
	1988	6.24	"The TAME Project: Towards Improvement-Oriented Software Environments," V. R. Basili and H. D. Rombach, *IEEE Transactions on Software Engineering*, June 1988
	1988	8.15	"Lessons Learned in the Implementation Phase of a Large Ada Project," C. E. Brophy, S. Godfrey, W. W. Agresti, and V. R. Basili, *Proceedings of the Washington Ada Technical Conference*, March 1988
	1989	2.25	"The Experience Factory: Packaging Software Experience," V. R. Basili, *Proceedings of the Fourteenth Annual Software Engineering Workshop*, SEL-89-007, November 1989
	1989	2.25	"Evaluation of the Cleanroom Methodology in the Software Engineering Laboratory," A. Kouchakdjian, S. Green, and V. Basili, *Proceedings of the Fourteenth Annual Software Engineering Workshop*, SEL-89-007, November 1989
	1989	5.26	*Maintenance = Reuse-Oriented Software Development*, V. Basili, TR-2244, University of Maryland, Technical Report, May 1989
	1989	5.27	*Software Development: A Paradigm for the Future*, V. Basili, TR-2263, University of Maryland, Technical Report, June 1989
	1989	6.30	*Integrating Automated Support for a Software Management Cycle Into the TAME System*, V. Basili and T. Sunazuka, TR-2289, University of Maryland, Technical Report, July 1989

Author	Year	Section	Title
Basili, V. R. (Cont'd)	1989	7.17	"An Evaluation of Expert Systems for Software Engineering Management," C. L. Ramsey and V. R. Basili, *IEEE Transactions on Software Engineering*, June 1989
	1990	2.26	"Towards a Mature Measurement Environment: Creating a Software Engineering Research Environment," V. R. Basili, *Proceedings of the Fifteenth Annual Software Engineering Workshop*, SEL-90-006, November 1990
	1990	5.28	*Towards a Comprehensive Framework for Reuse: Model-Based Reuse Characterization Schemes*, V. Basili and H. Rombach, TR-2446, University of Maryland, Technical Report, April 1990
	1990	5.29	"Viewing Maintenance as Reuse-Oriented Software Development," V. Basili, *IEEE Software*, January 1990
	1990	8.26	"Software Reclamation: Improving Post-Development Reusability," J. Bailey and V. Basili, *Proceedings of the Eighth Annual National Conference on Ada Technology*, March 1990
	1991	2.27	"Methodological and Architectural Issues in the Experience Factory," V. R. Basili and G. Caldiera, *Proceedings of the Sixteenth Annual Software Engineering Workshop*, SEL-91-006, December 1991
	1991	4.5	"Towards Automated Support for Extraction of Reusable Components," S. K. Abd-El-Hafiz, V. R. Basili, and G. Caldiera, *Proceedings of the IEEE Conference on Software Maintenance—1991 (CSM 91)*, October 1991
	1991	5.31	"Support for Comprehensive Reuse," V. R. Basili and H. D. Rombach, *Software Engineering Journal*, September 1991
	1991	5.33	*A Pattern Recognition Approach for Software Engineering Data Analysis*, L. C. Briand, V. R. Basili, and W. M. Thomas, TR-2672, University of Maryland, Technical Report, March 1991

Author	Year	Section	Title
Basili, V. R. (Cont'd)	1991	5.35	"The Software-Cycle Model for Re-Engineering and Reuse," J. W. Bailey and V. R. Basili, *Proceedings of the ACM Tri-Ada 91 Conference*, October 1991
	1991	6.33	"Paradigms for Experimentation and Empirical Studies in Software Engineering," V. R. Basili and R. W. Selby, *Reliability Engineering and System Safety*, January 1991
	1992	2.34	"The Software Engineering Laboratory— An Operational Software Experience Factory," V. Basili, G. Caldiera, F. McGarry, et al., *Proceedings of the Fourteenth International Conference on Software Engineering (ICSE 92)*, May 1992
	1992	5.32	"A Reference Architecture for the Component Factory," V. R. Basili, G. Caldiera, and G. Cantone, *ACM Transactions on Software Engineering and Methodology*, January 1992
	1992	5.38	"Providing an Empirical Basis for Optimizing the Verification and Testing Phases of Software Development," L. C. Briand, V. R. Basili, and C. J. Hetmanski, *Proceedings of the Third IEEE International Symposium on Software Reliability Engineering (ISSRE 92)*, October 1992
	1992	5.39	"A Classification Procedure for the Effective Management of Changes During the Maintenance Process," L. C. Briand and V. R. Basili, *Proceedings of the 1992 IEEE Conference on Software Maintenance (CSM 92)*, November 1992
	1992	2.28	"The Experience Factory: Can It Make You a 5?" V. R. Basili, *Proceedings of the Seventeenth Annual Software Engineering Workshop*, SEL-92-004, December 1992
	1993	5.40	*Developing Interpretable Models with Optimized Set Reduction for Identifying High Risk Software Components*, L. C. Briand, V. R. Basili, and C. J. Hetmanski, TR-3048, University of Maryland, Technical Report, March 1993

Author	Year	Section	Title
Basili, V. R. (Cont'd)	1993	6.35	"Measuring and Assessing Maintainability at the End of High Level Design," L. C. Briand, S. Morasca, and V. R. Basili, *Proceedings of the 1993 IEEE Conference on Software Maintenance (CSM 93)*, November 1993
Beane, J.	1981	5.9	"Can the Parr Curve Help With Manpower Distribution and Resource Estimation Problems?" V. R. Basili and J. Beane, *Journal of Systems and Software*, February 1981, Vol. 2, No. 1
Belford, P.	1979	2.15	"Central Flow Control Software Development: A Case Study of the Effectiveness of Software Engineering Techniques," P. C. Belford, R. A. Berg, and T. L. Hannan, *Proceedings from the Fourth Summer Software Engineering Workshop*, SEL-79-005, November 1979
Berg, R. A.	1979	2.15	"Central Flow Control Software Development: A Case Study of the Effectiveness of Software Engineering Techniques," P. C. Belford, R. A. Berg, and T. L. Hannan, *Proceedings from the Fourth Summer Software Engineering Workshop*, SEL-79-005, November 1979
Booth, E.	1989	8.23	"Using Ada To Maximize Verbatim Software Reuse," M. Stark and E. Booth, *Proceedings of Tri-Ada 1989*, October 1989
	1991	8.34	*Software Engineering Laboratory (SEL) Ada Performance Study Report*, SEL-91-003, E. W. Booth and M. E. Stark, July 1991
	1991	8.35	"Designing Configurable Software: COMPASS Implementation Concepts," E. W. Booth and M. E. Stark, *Proceedings of Tri-Ada 1991*, October 1991
	1992	8.37	"Software Engineering Laboratory Ada Performance Study—Results and Implications," E. W. Booth and M. E. Stark, *Proceedings of the Fourth Annual NASA Ada User's Symposium*, April 1992

Author	Year	Section	Title
Briand, L. C.	1991	2.27	"Optimized Set Reduction for Empirically Guiding Software Development," A. Porter and L. Briand, *Proceedings of the Sixteenth Annual Software Engineering Workshop*, SEL-91-006, December 1991
	1991	5.33	*A Pattern Recognition Approach for Software Engineering Data Analysis*, L. C. Briand, V. R. Basili, and W. M. Thomas, TR-2672, University of Maryland, Technical Report, May 1991
	1992	5.38	"Providing an Empirical Basis for Optimizing the Verification and Testing Phases of Software Development," L. C. Briand, V. R. Basili, and C. J. Hetmanski, *Proceedings of the Third IEEE International Symposium on Software Reliability Engineering (ISSRE 92)*, October 1992
	1992	5.39	"A Classification Procedure for the Effective Management of Changes During the Maintenance Process," L. C. Briand and V. R. Basili, *Proceedings of the 1992 IEEE Conference on Software Maintenance (CSM 92)*, November 1992
	1993	5.40	*Developing Interpretable Models with Optimized Set Reduction for Identifying High Risk Software Components*, L. C. Briand, V. R. Basili, and C. J. Hetmanski, TR-3048, University of Maryland, Technical Report, March 1993
	1993	5.41	"Modeling and Managing Risk Early in Software Development," L. C. Briand, W. M. Thomas, and C. J. Hetmanski, *Proceedings of the Fifteenth International Conference on Software Engineering (ICSE 93)*, May 1993
	1993	6.35	"Measuring and Assessing Maintainability at the End of High Level Design," L. C. Briand, S. Morasca, and V. R. Basili, *Proceedings of the 1993 IEEE Conference on Software Maintenance (CSM 93)*, November 1993
Bristow, J.	1992	9.8	*Software Engineering Laboratory (SEL) Database Organization and User's Guide (Revision 2)*, SEL-89-201, L. Morusiewicz, J. Bristow, et al., October 1992

Author	Year	Section	Title
Brophy, C. E.	1987	2.23	"An Experiment in Ada in the Software Engineering Laboratory—Lessons Learned from the Ada Code/Unit Test Phase," S. Godfrey and C. Brophy, *Proceedings of the Twelfth Annual Software Engineering Workshop*, SEL-87-010, December 1987
	1987	8.8	"Lessons Learned in Use of Ada™- Oriented Design Methods," C. E. Brophy, W. W. Agresti, and V. R. Basili, *Proceedings of the Joint Ada Conference*, March 1987
	1987	8.10	*Assessing the Ada® Design Process and Its Implications: A Case Study*, SEL-87-004, C. Brophy and S. Godfrey, July 1987
	1988	8.15	"Lessons Learned in the Implementation Phase of a Large Ada Project," C. E. Brophy, S. Godfrey, W. W. Agresti, and V. R. Basili, *Proceedings of the Washington Ada Technical Conference*, March 1988
	1988	8.17	"Experiences in the Implementation of a Large Ada Project," S. Godfrey and C. Brophy, *Proceedings of the 1988 Washington Ada Symposium*, June 1988
	1989	8.21	*Implementation of a Production Ada Project: The GRODY Study*, SEL-89-002, S. Godfrey and C. Brophy, May 1989
	1989	8.25	*Lessons Learned in the Transition to Ada From FORTRAN at NASA/Goddard*, SEL-89-005, C. Brophy, November 1989
Buell, J. C.	1986	4.3	*Flight Dynamics System Software Development Environment Tutorial (FDF/SDE)*, SEL-86-003, J. C. Buell and P. I. Myers, July 1986
	1988	2.24	"The Software Management Environment," J. D. Valett, W. Decker, and J. Buell, *Proceedings of the Thirteenth Annual Software Engineering Workshop*, SEL-88-004, November 1988
Buhler, M.	1990	9.9	*Database Access Manager for the Software Engineering Laboratory (DAMSEL) User's Guide*, SEL-90-001, M. Buhler et al., March 1990

Author	Year	Section	Title
Caldiera, G.	1991	2.27	"Methodological and Architectural Issues in the Experience Factory," V. R. Basili and G. Caldiera, *Proceedings of the Sixteenth Annual Software Engineering Workshop*, SEL-91-006, December 1991
Caldiera, G. (Cont'd)	1991	4.5	"Towards Automated Support for Extraction of Reusable Components," S. K. Abd-El-Hafiz, V. R. Basili, and G. Caldiera, *Proceedings of the IEEE Conference on Software Maintenance—1991 (CSM 91)*, October 1991
	1992	2.34	"The Software Engineering Laboratory—An Operational Software Experience Factory," V. Basili, G. Caldiera, F. McGarry, et al., *Proceedings of the Fourteenth International Conference on Software Engineering (ICSE 92)*, May 1992
	1992	5.32	"A Reference Architecture for the Component Factory," V. R. Basili, G. Caldiera, and G. Cantone, *ACM Transactions on Software Engineering and Methodology*, January 1992
Cantone, G.	1992	5.32	"A Reference Architecture for the Component Factory," V. R. Basili, G. Caldiera, and G. Cantone, *ACM Transactions on Software Engineering and Methodology*, January 1992
Card, D. N.	1981	2.17	"Identification and Evaluation of Software Measures," D. N. Card, *Proceedings of the Sixth Annual Software Engineering Workshop*, SEL-83-007, November 1983
	1982	2.31	*The Software Engineering Laboratory*, SEL-81-104, D. N. Card, F. E. McGarry, G. Page, et al., February 1982
	1982	5.11	"Comparison of Regression Modeling Techniques for Resource Estimation," D. N. Card, Computer Sciences Corporation, Technical Memorandum, November 1982
	1982	6.7	"Early Estimation of Resource Expenditures and Program Size," D. N. Card, Computer Sciences Corporation, Technical Memorandum, June 1982

Author	Year	Section	Title
Card, D. N. (Cont'd)	1982	6.8	*Evaluation of Management Measures of Software Development*, SEL-82-001, G. Page, D. N. Card, and F. E. McGarry, September 1982, Vol. 1 and Vol. 2
	1982	9.3	*Guide to Data Collection*, SEL-81-101, V. E. Church, D. N. Card, and F. E. McGarry, August 1982
	1983	2.19	"Evaluating Software Engineering Technologies in the SEL," D. N. Card, F. E. McGarry, and G. Page, *Proceedings of the Eighth Annual Software Engineering Workshop*, SEL-83-007, November 1983
	1984	5.14	*An Approach to Software Cost Estimation*, SEL-83-001, F. E. McGarry, G. Page, D. N. Card, et al., February 1984
	1984	6.10	"Measuring Software Technology," W. W. Agresti, F. E. McGarry, D. N. Card, et al., *Program Transformation and Programming Environments*. New York: Springer-Verlag, 1984
	1984	6.12	*Measures and Metrics for Software Development*, SEL-83-002, D. N. Card, F. E. McGarry, G. Page, et al., March 1984
	1984	6.13	Characteristics of FORTRAN Modules," D. N. Card, Q. L. Jordan, and V. E. Church, Computer Sciences Corporation, Technical Memorandum, June 1984
	1984	7.5	"A Software Engineering View of the Flight Dynamics Analysis System (FDAS): Parts I and II," D. N. Card, W. W. Agresti, V. E. Church, and Q. L. Jordan, Computer Sciences Corporation, Technical Memorandum, December 1983 (Part I) and March 1984 (Part II)
	1984	7.6	"A Practical Experience With Independent Verification and Validation," G. Page, F. E. McGarry, and D. N. Card, *Proceedings of the Eighth International Computer Software and Applications Conference*. New York: IEEE Computer Society Press, 1984

Author	Year	Section	Title
Card, D. N. (Cont'd)	1985	3.4	*Software Verification and Testing*, SEL-85-005, D. N. Card, C. Antle, and E. Edwards, December 1985
	1985	6.16	"Criteria for Software Modularization," D. N. Card, G. Page, and F. E. McGarry, *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985
	1985	7.9	*Comparison of Software Verification Techniques*, SEL-85-001, D. N. Card, R. W. Selby, F. E. McGarry, et al., April 1985
	1985	7.11	*Evaluation of an Independent Verification and Validation (IV&V) Methodology for Flight Dynamics*, SEL-81-110, G. Page, F. E. McGarry, and D. N. Card, June 1985
	1985	7.14	"A Software Technology Evaluation Program," D. N. Card, *Annais do XVIII Congresso Nacional de Informatica*, October 1985
	1986	6.19	*Measuring Software Design*, SEL-86-005, D. N. Card, W. Agresti, V. Church, et al., November 1986
	1986	7.15	"An Empirical Study of Software Design Practices," D. N. Card, V. E. Church, and W. W. Agresti, *IEEE Transactions on Software Engineering*, February 1986
	1986	7.16	"An Approach for Assessing Software Prototypes," V. E. Church, D. N. Card, W. W. Agresti, and Q. L. Jordan, *ACM Software Engineering Notes*, July 1986
	1986	8.2	"Designing With Ada for Satellite Simulation: A Case Study," W. W. Agresti, V. E. Church, D. N. Card, and P. L. Lo, *Proceedings of the First International Symposium on Ada for the NASA Space Station*, June 1986
	1987	5.18	"Resolving the Software Science Anomaly," D. N. Card and W. W. Agresti, *Journal of Systems and Software*, 1987

Author	Year	Section	Title
Card, D. N. (Cont'd)	1987	7.19	"Evaluating Software Engineering Technologies," D. N. Card, F. E. McGarry, and G. T. Page, *IEEE Transactions on Software Engineering*, July 1987
	1988	6.25	"Measuring Software Design Complexity," D. N. Card and W. W. Agresti, *Journal of Systems and Software*, June 1988
Caron, G.	1981	9.1	*A Comparison of RADC and NASA/SEL Software Development Data*, C. Turner and G. Caron, Data & Analysis Center for Software, May 1981
Chen, E.	1979	2.15	"Software Engineering Laboratory: Data Validation," M. V. Zelkowitz and E. Chen, *Proceedings from the Fourth Summer Software Engineering Workshop*, SEL-79-005, November 1979
	1981	7.4	"Use of Cluster Analysis to Evaluate Software Engineering Methodologies," E. Chen and M. V. Zelkowitz, *Proceedings of the Fifth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1981
Church, V. E.	1979	2.15	"Software Engineering Laboratory—The Data Collection Process," V. Church, *Proceedings from the Fourth Summer Software Engineering Workshop*, SEL-79-005, November 1979
	1982	9.3	*Guide to Data Collection*, SEL-81-101, V. E. Church, D. N. Card, and F. E. McGarry, August 1982
	1984	6.13	"Characteristics of FORTRAN Modules," D. N. Card, Q. L. Jordan, and V. E. Church, Computer Sciences Corporation, Technical Memorandum, June 1984
	1984	6.15	*Investigation of Specification Measures for the Software Engineering Laboratory*, SEL-84-003, W. W. Agresti, V. E. Church, and F. E. McGarry, December 1984

Author	Year	Section	Title
Church, V. E. (Cont'd)	1984	7.5	"A Software Engineering View of the Flight Dynamics Analysis System (FDAS): Parts I and II," D. N. Card, W. W. Agresti, V. E. Church, and Q. L. Jordan, Computer Sciences Corporation, Technical Memorandum, December 1983 (Part I) and March 1984 (Part II)
	1986	6.19	*Measuring Software Design*, SEL-86-005, D. N. Card, W. Agresti, V. Church, et al., November 1986
	1986	7.15	"An Empirical Study of Software Design Practices," D. N. Card, V. E. Church, and W. W. Agresti, *IEEE Transactions on Software Engineering*, February 1986
	1986	7.16	"An Approach for Assessing Software Prototypes," V. E. Church, D. N. Card, W. W. Agresti, and Q. L. Jordan, *ACM Software Engineering Notes*, July 1986
	1986	8.2	"Designing With Ada for Satellite Simulation: A Case Study," W. W. Agresti, V. E. Church, D. N. Card, and P. L. Lo, *Proceedings of the First International Symposium on Ada for the NASA Space Station*, June 1986
Cook, J. F.	1980	5.7	*An Appraisal of Selected Cost/Resource Estimation Models for Software Systems*, SEL-80-007, J. F. Cook and F. E. McGarry, December 1980
Damon, E.	1976	2.12	"DOMONIC As a Design and Management Tool," E. Damon, *Proceedings from the First Summer Software Engineering Workshop*, SEL-76-001, August 1976
Decker, W. J.	1979	7.2	*Evaluation of the Caine, Farber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Design Environment*, SEL-79-004, C. E. Goorevich, A. L. Green, and W. J. Decker, September 1979
	1980	7.3	*Multi-Level Expression Design Language—Requirement Level (MEDL-R) System Evaluation*, SEL-80-002, W. J. Decker and C. E. Goorevich, May 1980

Author	Year	Section	Title
Decker, W. J. (Cont'd)	1981	9.2	*Automated Collection of Software Engineering Data in the Software Engineering Laboratory (SEL)*, SEL-81-014, A. L. Green, W. J. Decker, and F. E. McGarry, September 1981
	1985	4.1	*FORTRAN Static Source Code Analyzer Program (SAP) System Description (Revision 1)*, SEL-82-102, W. A. Taylor and W. J. Decker, April 1985
	1986	4.2	*FORTRAN Static Source Code Analyzer Program (SAP) User's Guide (Revision 3)*, SEL-78-302, W. J. Decker and W. A. Taylor, July 1986
	1988	2.24	"The Software Management Environment," J. D. Valett, W. Decker, and J. Buell, *Proceedings of the Thirteenth Annual Software Engineering Workshop*, SEL-88-004, November 1988
	1991	6.32	*Software Engineering Laboratory (SEL) Relationships, Models, and Management Rules*, SEL-91-001, W. Decker, R. Hendrick, and J. Valett, February 1991
Doerflinger, C. W.	1983	2.19	"Monitoring Software Development Through Dynamic Variables," V. Basili and C. Doerflinger, *Proceedings of the Eighth Annual Software Engineering Workshop*, SEL-83-007, November 1983
	1983	5.12	"Monitoring Software Development Through Dynamic Variables," C. W. Doerflinger and V. R. Basili, *Proceedings of the Seventh International Computer Software and Applications Conference*. New York: IEEE Computer Society Press, 1983
	1983	5.13	*Monitoring Software Development Through Dynamic Variables*, SEL-83-006, C. W. Doerflinger, November 1983
Doubleday, D. L.	1987	8.11	"ASAP: An Ada Static Source Code Analyzer Program," D. L. Doubleday, University of Maryland, Technical Memorandum, August 1987
Edwards, E.	1985	3.4	*Software Verification and Testing*, SEL-85-005, D. N. Card, C. Antle, and E. Edwards, December 1985

Author	Year	Section	Title
Edwards, E. (Cont'd)	1986	3.3	*Programmer's Handbook for Flight Dynamics Software Development*, SEL-86-001, R. Wood and E. Edwards, March 1986
Esker, L.	1988	2.24	"Evolving Impact of Ada on a Production Software Environment," F. McGarry, L. Esker, and K. Quimby, *Proceedings of the Thirteenth Annual Software Engineering Workshop*, SEL-88-004, November 1988
	1988	8.19	*Evaluation of Ada Technology in the Flight Dynamics Area: Design Phase Analysis*, SEL-88-003, K. Quimby and L. Esker, December 1988
	1989	8.22	"Evolution of Ada Technology in a Production Software Environment," F. McGarry, L. Esker, and K. Quimby, *Proceedings of the Washington Ada Symposium (WADAS)*, June 1989
	1989	8.24	*Evolution of Ada Technology in the Flight Dynamics Area: Implementation/Testing Phase Analysis*, SEL-89-004, K. Quimby, L. Esker, L. Smith, M. Stark, and F. McGarry, November 1989
Freburger, K.	1979	5.3	*The Software Engineering Laboratory: Relationship Equations*, SEL-79-002, K. Freburger and V. R. Basili, May 1979
	1981	6.5	"Programming Measurement and Estimation in the Software Engineering Laboratory," V. R. Basili and K. Freburger, *Journal of Systems and Software*, February 1981, Vol. 2, No. 1
Godfrey, S.	1987	2.23	"An Experiment in Ada in the Software Engineering Laboratory—Lessons Learned from the Ada Code/Unit Test Phase," S. Godfrey and C. Brophy, *Proceedings of the Twelfth Annual Software Engineering Workshop*, SEL-87-010, December 1987
	1987	8.10	*Assessing the Ada® Design Process and Its Implications: A Case Study*, SEL-87-004, C. Brophy and S. Godfrey, July 1987

Author	Year	Section	Title
Godfrey, S. (Cont'd)	1988	8.15	"Lessons Learned in the Implementation Phase of a Large Ada Project," C. E. Brophy, S. Godfrey, W. W. Agresti, and V. R. Basili, *Proceedings of the Washington Ada Technical Conference*, March 1988
	1988	8.17	"Experiences in the Implementation of a Large Ada Project," S. Godfrey and C. Brophy, *Proceedings of the 1988 Washington Ada Symposium*, June 1988
	1989	8.21	*Implementation of a Production Ada Project: The GRODY Study*, SEL-89-002, S. Godfrey and C. Brophy, May 1989
Goorevich, C. E.	1979	7.2	*Evaluation of the Caine, Farber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Design Environment*, SEL-79-004, C. E. Goorevich, A. L. Green, and W. J. Decker, September 1979
	1980	7.3	*Multi-Level Expression Design Language—Requirement Level (MEDL-R) System Evaluation*, SEL-80-002, W. J. Decker and C. E. Goorevich, May 1980
Grantham, C.	1983	2.19	"Evaluating Multiple Coordinated Windows for Programming Workstations," B. Schneiderman, C. Grantham, K. Norman, *Proceedings of the Eighth Annual Software Engineering Workshop*, SEL-83-007, November 1983
Green, A. L.	1979	7.2	*Evaluation of the Caine, Farber, and Gordon Program Design Language (PDL) in the Goddard Space Flight Center (GSFC) Code 580 Software Design Environment*, SEL-79-004, C. E. Goorevich, A. L. Green, and W. J. Decker, September 1979
	1981	9.2	*Automated Collection of Software Engineering Data in the Software Engineering Laboratory (SEL)*, SEL-81-014, A. L. Green, W. J. Decker, and F. E. McGarry, September 1981

Author	Year	Section	Title
Green, S. E.	1989	2.25	"Evaluation of the Cleanroom Methodology in the Software Engineering Laboratory," A. Kouchakdjian, S. Green, and V. Basili, *Proceedings of the Fourteenth Annual Software Engineering Workshop*, SEL-89-007, November 1989
	1990	7.24	*The Cleanroom Case Study in the Software Engineering Laboratory: Project Description and Early Analysis*, SEL-90-002, S. Green et al., March 1990
	1991	2.27	"Cleanroom Process Evolution in the SEL," S. E. Green and R. Pajerski, *Proceedings of the Sixteenth Annual Software Engineering Workshop*, SEL-91-006, December 1991
	1991	5.34	*Software Engineering Laboratory (SEL) Cleanroom Process Model*, SEL-91-004, S. Green, November 1991
Hall, D.	1985	7.8	"Measuring the Impact of Computer Resource Quality on the Software Development Process and Product," F. E. McGarry, J. Valett, and D. Hall, *Proceedings of the Hawaii International Conference on System Sciences*, January 1985
Hall, K. R.	1992	2.28	"Development and Application of an Acceptance Testing Model," R. D. Pendley, C. H. Noonan, and K. R. Hall, *Proceedings of the Seventeenth Annual Software Engineering Workshop*, SEL-92-004, December 1992
Hannan, T. L.	1979	2.15	"Central Flow Control Software Development: A Case Study of the Effectiveness of Software Engineering Techniques," P. C. Belford, R. A. Berg, and T. L. Hannan, *Proceedings from the Fourth Summer Software Engineering Workshop*, SEL-79-005, November 1979
Heller, G.	1990	2.26	"Impact of a Process Improvement Program in a Production Software Environment: Are We Any Better?" G. H. Heller and G. T. Page, *Proceedings of the Fifteenth Annual Software Engineering Workshop*, SEL-90-006, November 1990

Author	Year	Section	Title
Heller, G. (Cont'd)	1992	9.7	*Data Collection Procedures for the Software Engineering Laboratory (SEL) Database*, SEL-92-002, G. Heller, J. Valett, and M. Wild, March 1992
Hendrick, R.	1991	6.32	*Software Engineering Laboratory (SEL) Relationships, Models, and Management Rules*, SEL-91-001, W. Decker, R. Hendrick, and J. Valett, February 1991
	1992	4.4	*Software Management Environment (SME) Concepts and Architecture (Revision 1)*, SEL-89-103, R. Hendrick, D. Kistler, and J. Valett, September 1992
Hetmanski, C. J.	1992	5.38	"Providing an Empirical Basis for Optimizing the Verification and Testing Phases of Software Development," L. C. Briand, V. R. Basili, and C. J. Hetmanski, *Proceedings of the Third IEEE International Symposium on Software Reliability Engineering (ISSRE 92)*, October 1992
	1993	5.40	*Developing Interpretable Models with Optimized Set Reduction for Identifying High Risk Software Components*, L. C. Briand, V. R. Basili, and C. J. Hetmanski, TR-3048, University of Maryland, Technical Report, March 1993
	1993	5.41	"Modeling and Managing Risk Early in Software Development," L. C. Briand, W. M. Thomas, and C. J. Hetmanski, *Proceedings of the Fifteenth International Conference on Software Engineering (ICSE 93)*, May 1993
Hutchens, D. H.	1986	5.16	"Experimentation in Software Engineering," V. R. Basili, R. W. Selby, Jr., and D. H. Hutchens, *IEEE Transactions on Software Engineering*, July 1986
Jeffery, D. R.	1987	5.24	*Characterizing Resource Data: A Model for Logical Association of Software Data*, D. R. Jeffery and V. R. Basili, TR-1848, University of Maryland, Technical Report, May 1987
	1988	6.23	"Validating the TAME Resource Data Model," D. R. Jeffery and V. R. Basili, *Proceedings of the 10th International Conference on Software Engineering*, April 1988

Author	Year	Section	Title
Jeletic, K.	1992	4.6	*Software Management Environment (SME) Installation Guide*, SEL-92-001, D. Kistler and K. Jeletic, January 1992
Jordan, Q. L.	1984	6.13	"Characteristics of FORTRAN Modules," D. N. Card, Q. L. Jordan, and V. E. Church, Computer Sciences Corporation, Technical Memorandum, June 1984
	1984	7.5	"A Software Engineering View of the Flight Dynamics Analysis System (FDAS): Parts I and II," D. N. Card, W. W. Agresti, V. E. Church, and Q. L. Jordan, Computer Sciences Corporation, Technical Memorandum, December 1983 (Part I) and March 1984 (Part II)
	1986	7.16	"An Approach for Assessing Software Prototypes," V. E. Church, D. N. Card, W. W. Agresti, and Q. L. Jordan, *ACM Software Engineering Notes*, July 1986
Jun, L.	1990	8.30	*A Study of the Portability of an Ada System in the Software Engineering Laboratory (SEL)*, SEL-90-003, L. Jun et al., June 1990
Kester, R.	1990	2.26	"SEL Ada Reuse Analysis and Representations," R. Kester, *Proceedings of the Fifteenth Annual Software Engineering Workshop*, SEL-90-006, November 1990
Kistler, D.	1992	4.4	*Software Management Environment (SME) Concepts and Architecture (Revision 1)*, SEL-89-103, R. Hendrick, D. Kistler, and J. Valett, September 1992
	1992	4.6	*Software Management Environment (SME) Installation Guide*, SEL-92-001, D. Kistler and K. Jeletic, January 1992
Kouchakdjian, A.	1989	2.25	"Evaluation of the Cleanroom Methodology in the Software Engineering Laboratory," A. Kouchakdjian, S. Green, and V. Basili, *Proceedings of the Fourteenth Annual Software Engineering Workshop*, SEL-89-007, November 1989

Author	Year	Section	Title
Landis, L.	1990	3.2	*Manager's Handbook for Software Development (Revision 1)*, SEL-84-101, L. Landis, F. E. McGarry, S. Waligora, et al., November 1990
	1992	3.1	*Recommended Approach to Software Development (Revision 3)*, SEL-81-305, L. Landis, S. Waligora, F. McGarry, et al., June 1992
Langston, J.	1992	2.28	"Maximizing Reuse: Applying Common Sense and Discipline," S. Waligora and J. Langston, *Proceedings of the Seventeenth Annual Software Engineering Workshop*, SEL-92-004, December 1992
Li, N. R.	1993	5.42	"An Information Model for Use in Software Management Estimation and Prediction," N. R. Li and M. V. Zelkowitz, *Proceedings of the Second International Conference on Information and Knowledge Management*, November 1993
Lo, P. L.	1986	8.2	"Designing With Ada for Satellite Simulation: A Case Study," W. W. Agresti, V. E. Church, D. N. Card, and P. L. Lo, *Proceedings of the First International Symposium on Ada for the NASA Space Station*, June 1986
Mapp, T. E.	1978	5.1	*Applicability of the Rayleigh Curve to the SEL Environment*, SEL-78-007, T. E. Mapp, December 1978
Mark, L.	1987	5.21	*A Meta Information Base for Software Engineering*, L. Mark and H. D. Rombach, TR-1765, University of Maryland, Technical Report, July 1987
	1989	5.22	"Generating Customized Software Engineering Information Bases from Software Process and Product Specifications," L. Mark and H. D. Rombach, *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, January 1989
	1989	5.23	"Software Process and Product Specifications: A Basis for Generating Customized SE Information Bases," H. D. Rombach and L. Mark, *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, January 1989

Author	Year	Section	Title
McDermott, T.	1989	2.25	"Experiences in the Software Engineering Laboratory (SEL) Applying Software Measurement," F. McGarry, S. Waligora, and T. McDemott, *Proceedings of the Fourteenth Annual Software Engineering Workshop*, SEL-89-007, November 1989
	1990	8.31	*Gamma Ray Observatory Dynamics Simulator in Ada (GRODY) Experiment Summary*, SEL-90-004, T. McDermott et al., September 1990
McGarry, F. E.	1979	2.15	"Overview of the Software Engineering Laboratory," F. McGarry, *Proceedings from the Fourth Summer Software Engineering Workshop*, SEL-79-005, November 1979
	1980	2.16	"An Approach to Measuring Software Technology," F. McGarry, *Proceedings from the Fifth Annual Software Engineering Workshop*, SEL-80-006, November 1980
	1980	5.7	*An Appraisal of Selected Cost/Resource Estimation Models for Software Systems*, SEL-80-007, J. F. Cook and F. E. McGarry, December 1980
	1981	9.2	*Automated Collection of Software Engineering Data in the Software Engineering Laboratory (SEL)*, SEL-81-014, A. L. Green, W. J. Decker, and F. E. McGarry, September 1981
	1982	2.31	*The Software Engineering Laboratory*, SEL-81-104, D. N. Card, F. E. McGarry, G. Page, et al., February 1982
	1982	6.8	*Evaluation of Management Measures of Software Development*, SEL-82-001, G. Page, D. N. Card, and F. E. McGarry, September 1982, Vol. 1 and Vol. 2
	1982	9.3	*Guide to Data Collection*, SEL-81-101, V. E. Church, D. N. Card, and F. E. McGarry, August 1982
	1983	2.19	"Evaluating Software Engineering Technologies in the SEL," D. N. Card, F. E. McGarry, and G. Page, *Proceedings of the Eighth Annual Software Engineering Workshop*, SEL-83-007, November 1983

Author	Year	Section	Title
McGarry, F. E. (Cont'd)	1983	2.32	*Glossary of Software Engineering Laboratory Terms*, SEL-82-105, T. A. Babst, M. G. Rohleder, and F. E. McGarry, November 1983
	1984	2.20	"Evaluating Software Testing Strategies," R. W. Selby, Jr., V. R. Basili, J. Page, and F. E. McGarry, *Proceedings of the Ninth Annual Software Engineering Workshop*, SEL-84-004, November 1984
	1984	5.14	*An Approach to Software Cost Estimation*, SEL-83-001, F. E. McGarry, G. Page, D. N. Card, et al., February 1984
	1984	6.10	"Measuring Software Technology," W. W. Agresti, F. E. McGarry, D. N. Card, et al., *Program Transformation and Programming Environments*. New York: Springer-Verlag, 1984
	1984	6.12	*Measures and Metrics for Software Development*, SEL-83-002, D. N. Card, F. E. McGarry, G. Page, et al., March 1984
	1984	6.15	*Investigation of Specification Measures for the Software Engineering Laboratory*, SEL-84-003, W. W. Agresti, V. E. Church, and F. E. McGarry, December 1984
	1984	7.6	"A Practical Experience With Independent Verification and Validation," G. Page, F. E. McGarry, and D. N. Card, *Proceedings of the Eighth International Computer Software and Applications Conference*. New York: IEEE Computer Society Press, 1984
	1985	2.21	"Recent SEL Studies," F. E. McGarry, *Proceedings from the Tenth Annual Software Engineering Workshop*, SEL-85-006, December 1985
	1985	6.16	"Criteria for Software Modularization," D. N. Card, G. Page, and F. E. McGarry, *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985

Author	Year	Section	Title
McGarry, F. E. (Cont'd)	1985	7.8	"Measuring the Impact of Computer Resource Quality on the Software Development Process and Product," F. E. McGarry, J. Valett, and D. Hall, *Proceedings of the Hawaii International Conference on System Sciences*, January 1985
	1985	7.9	*Comparison of Software Verification Techniques*, SEL-85-001, D. N. Card, R. W. Selby, F. E. McGarry, et al., April 1985
	1985	7.11	*Evaluation of an Independent Verification and Validation (IV&V) Methodology for Flight Dynamics*, SEL-81-110, G. Page, F. E. McGarry, and D. N. Card, June 1985
	1986	2.22	"Determining Software Productivity Leverage Factors in the SEL," F. McGarry, S. Voltz, and J. Valett, *Proceedings of the Eleventh Annual Software Engineering Workshop*, SEL-86-006, December 1986
	1987	7.19	"Evaluating Software Engineering Technologies," D. N. Card, F. E. McGarry, and G. T. Page, *IEEE Transactions on Software Engineering*, July 1987
	1988	2.24	"Evolving Impact of Ada on a Production Software Environment," F. McGarry, L. Esker, and K. Quimby, *Proceedings of the Thirteenth Annual Software Engineering Workshop*, SEL-88-004, November 1988
	1988	6.28	"A Summary of Software Measurement Experiences in the Software Engineering Laboratory," J. D. Valett and F. E. McGarry, *Proceedings of the 21st Annual Hawaii International Conference on System Sciences*, January 1988
	1988	8.13	"Measuring Ada for Software Development in the Software Engineering Laboratory (SEL)," F. E. McGarry and W. W. Agresti, *Proceedings of the 21st Annual Hawaii International Conference on System Sciences*, January 1988

Author	Year	Section	Title
McGarry, F. E. (Cont'd)	1989	2.25	"Experiences in the Software Engineering Laboratory (SEL) Applying Software Measurement," F. McGarry, S. Waligora, and T. McDermott, *Proceedings of the Fourteenth Annual Software Engineering Workshop*, SEL-89-007, November 1989
	1989	8.22	"Evolution of Ada Technology in a Production Software Environment," F. McGarry, L. Esker, and K.Quimby, *Proceedings of the Washington Ada Symposium (WADAS)*, June 1989
	1989	8.24	*Evolution of Ada Technology in the Flight Dynamics Area: Implementation/Testing Phase Analysis*, SEL-89-004, K. Quimby, L. Esker, L. Smith, M. Stark, and F. McGarry, November 1989
	1990	2.26	"Towards Understanding Software—15 Years in the SEL," F. McGarry and R. Pajerski, *Proceedings of the Fifteenth Annual Software Engineering Workshop*, SEL-90-006, November 1990
	1990	3.2	*Manager's Handbook for Software Development (Revision 1)*, SEL-84-101, L. Landis, F. E. McGarry, S. Waligora, et al., November 1990
	1991	2.27	"Experiments in Software Engineering Technology," F. McGarry and S. Waligora, *Proceedings of the Sixteenth Annual Software Engineering Workshop*, SEL-91-006, December 1991
	1991	2.33	*Software Engineering Laboratory (SEL) Data and Information Policy (Revision 1)*, SEL-91-102, F. McGarry, August 1991
	1992	2.34	"The Software Engineering Laboratory—An Operational Software Experience Factory," V. Basili, G. Caldiera, F. McGarry, et al., *Proceedings of the Fourteenth International Conference on Software Engineering (ICSE 92)*, May 1992
	1992	3.1	*Recommended Approach to Software Development (Revision 3)*, SEL-81-305, L. Landis, S. Waligora, F. McGarry, et al., June 1992

Author	Year	Section	Title
McGarry, F. E. (Cont'd)	1992	2.28	"Experimental Software Engineering: Seventeen Years of Lessons in the SEL," F. McGarry, *Proceedings of the Seventeenth Annual Software Engineering Workshop*, SEL-92-004, December 1992
Miller, A. M.	1980	5.6	*A Study of the Musa Reliability Model*, SEL-80-005, A. M. Miller, November 1980
Miyamoto, I.	1982	2.18	"User Interface Design of a Software Tool System as a Technology Transfer Vehicle," I. Miyamoto, *Proceedings of the Seventh Annual Software Engineering Workshop*, SEL-82-007, December 1982
Morasca, S.	1993	6.35	"Measuring and Assessing Maintainability at the End of High Level Design," L. C. Briand, S. Morasca, and V. R. Basili, *Proceedings of the 1993 IEEE Conference on Software Maintenance (CSM 93)*, November 1993
Morusiewicz, L.	1992	9.8	*Software Engineering Laboratory (SEL) Database Organization and User's Guide (Revision 2)*, SEL-89-201, L. Morusiewicz, J. Bristow, et al., October 1992
Murphy, R.	1985	8.1	*Ada Training Evaluation and Recommendations From the Gamma Ray Observatory Ada Development Team*, SEL-85-002, R. Murphy and M. Stark, October 1985
Myers, P. I.	1986	4.3	*Flight Dynamics System Software Development Environment Tutorial, (FDF/SDE)*, SEL-86-003, J. C. Buell and P. I. Myers, July 1986
Noonan, C. H.	1992	2.28	"Development and Application of an Acceptance Testing Model," R. D. Pendley, C. H. Noonan, and K. R. Hall, *Proceedings of the Seventeenth Annual Software Engineering Workshop*, SEL-92-004, December 1992
Norman, K.	1983	2.19	"Evaluating Multiple Coordinated Windows for Programmer Workstations," B. Schneiderman, C. Grantham, K. Norman, et al., *Proceedings of the Eighth Annual Software Engineering Workshop*, SEL-83-007, November 1983

Author	Year	Section	Title
Page, G. T.	1980	2.16	"Impacts of Experiment and Software Technology Changes in a Production Environment," J. Page, *Proceedings from the Fifth Annual Software Engineering Workshop*, SEL-80-006, November 1980
	1981	2.17	"Methodology Evaluation: Effects of Independent Verification and Integration on One Class of Application," J. Page, *Proceedings of the Sixth Annual Software Engineering Workshop*, SEL-81-013, December 1981
	1982	2.31	*The Software Engineering Laboratory*, SEL-81-104, D. N. Card, F. E. McGarry, G. Page, et al., February 1982
	1982	6.8	*Evaluation of Management Measures of Software Development*, SEL-82-001, G. Page, D. N. Card, and F. E. McGarry, September 1982, Vol. 1 and Vol. 2
	1983	2.19	"Evaluating Software Engineering Technologies in the SEL," D. N. Card, F. E. McGarry, and G. Page, *Proceedings of the Eighth Annual Software Engineering Workshop*, SEL-83-007, November 1983
	1984	2.20	"Evaluating Software Testing Strategies," R. W. Selby, Jr., V. R. Basili, J. Page, and F. E. McGarry, *Proceedings of the Ninth annual Software Engineering Workshop*, SEL-84-004, November 1984
	1984	5.14	*An Approach to Software Cost Estimation*, SEL-83-001, F. E. McGarry, G. Page, D. N. Card, et al., February 1984
	1984	6.12	*Measures and Metrics for Software Development*, SEL-83-002, D. N. Card, F. E. McGarry, G. Page, et al., March 1984
	1984	7.6	"A Practical Experience With Independent Verification and Validation," G. Page, F. E. McGarry, and D. N. Card, *Proceedings of the Eighth International Computer Software and Applications Conference*. New York: IEEE Computer Society Press, 1984

Author	Year	Section	Title
Page, G. T. (Cont'd)	1985	6.16	"Criteria for Software Modularization," D. N. Card, G. Page, and F. E. McGarry, *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985
	1985	7.11	*Evaluation of an Independent Verification and Validation (IV&V) Methodology for Flight Dynamics*, SEL-81-110, G. Page, F. E. McGarry, and D. N. Card, June 1985
	1987	7.19	"Evaluating Software Engineering Technologies," D. N. Card, F. E. McGarry, and G. T. Page, *IEEE Transactions on Software Engineering*, July 1987
	1990	2.26	"Impact of a Process Improvement Program in a Production Software Environment: Are We Any Better?" G. H. Heller and G. T. Page, *Proceedings of the Fifteenth Annual Software Engineering Workshop*, SEL-90-006, November 1990
Pajerski, R.	1990	2.26	"Towards Understanding Software—15 Years in the SEL," F. McGarry and R. Pajerski, *Proceedings of the Fifteenth Annual Software Engineering Workshop*, SEL-90-006, November 1990
	1991	2.27	"Cleanroom Process Evolution in the SEL," S. E. Green and R. Pajerski, *Proceedings of the Sixteenth Annual Software Engineering Workshop*, SEL-91-006, December 1991
Panlilio-Yap, N. M.	1985	5.15	"Finding Relationships Between Effort and Other Variables in the SEL," V. R. Basili and N. M. Panlilio-Yap, *Proceedings of the Ninth International Computer Software and Applications Conference*. New York: IEEE Computer Society Press, 1985
Patnaik, D.	1986	5.17	*A Study on Fault Prediction and Reliability Assessment in the SEL Environment*, TR-1699, V. R. Basili and D. Patnaik, University of Maryland, Technical Report, August 1986

Author	Year	Section	Title
Pendley, R. D.	1992	2.28	"Development and Application of an Acceptance Testing Model," R. D. Pendley, C. H. Noonan, and K. R. Hall, *Proceedings of the Seventeenth Annual Software Engineering Workshop*, SEL-92-004, December 1992
Perricone, B. T.	1982	2.18	"Software Errors and Complexity: An Empirical Investigation," V. R. Basili and B. T. Perricone, *Proceedings of the Seventh Annual Software Engineering Workshop*, SEL-82-007, December 1982
	1984	6.11	"Software Errors and Complexity: An Empirical Investigation," V. R. Basili and B. T. Perricone, *Communications of the ACM*, January 1984
Perry, S.	1987	3.5	*Product Assurance Policies and Procedures for Flight Dynamics Software Development*, SEL-87-001, S. Perry et al., March 1987
Philips, T.	1981	6.6	"Evaluating and Comparing Software Metrics in the Software Engineering Laboratory," V. R. Basili and T. Phillips, *Proceedings of the ACM Sigmetrics Symposium/Workshop: Quality Metrics*, March 1981
	1983	6.9	"Metric Analysis and Data Validation Across FORTRAN Projects," V. R. Basili, R. W. Selby, and T. Phillips, *IEEE Transactions on Software Engineering*, November 1983
Picasso, G. O.	1981	5.10	*The Rayleigh Curve as a Model for Effort Distribution Over the Life of Medium Scale Software Systems*, SEL-81-012, G. O. Picasso, December 1981
Porter, A.	1991	2.27	"Optimized Set Reduction for Empirically Guiding Software Development," A. Porter and L. Briand, *Proceedings of the Sixteenth Annual Software Engineering Workshop*, SEL-91-006, December 1991
	1992	5.37	"An Improved Classification Tree Analysis of High Cost Modules Based Upon an Axiomatic Definition of Complexity," J. Tian, A. Porter, and M. V. Zelkowitz, *Proceedings of the Third IEEE International Symposium on Software Reliability Engineering (ISSRE 92)*, October 1992

Author	Year	Section	Title
Quimby, K.	1988	2.24	"Evolving Impact of Ada on a Production Software Environment," F. McGarry, L. Esker, and K. Quimby, *Proceedings of the Thirteenth Annual Software Engineering Workshop*, SEL-88-004, November 1988
	1988	8.19	*Evaluation of Ada Technology in the Flight Dynamics Area: Design Phase Analysis*, SEL-88-003, K. Quimby and L. Esker, December 1988
	1989	8.22	"Evolution of Ada Technology in a Production Software Environment," F. McGarry, L. Esker, and K.Quimby, *Proceedings of the Washington Ada Symposium (WADAS)*, June 1989
	1989	8.24	*Evolution of Ada Technology in the Flight Dynamics Area: Implementation/Testing Phase Analysis*, SEL-89-004, K. Quimby, L. Esker, L. Smith, M. Stark, and F. McGarry, November 1989
Ramsey, C. L.	1985	7.22	"ARROWSMITH-P—A Prototype Expert System for Software Engineering Management," V. R. Basili and C. L. Ramsey, *Proceedings of the IEEE/MITRE Expert Systems in Government Symposium*, October 1985
	1989	7.17	"An Evaluation of Expert Systems for Software Engineering Management," C. L. Ramsey and V. R. Basili, *IEEE Transactions on Software Engineering*, June 1989
Ramsey, J.	1983	2.19	"Structure Coverage of Functional Testing," J. Ramsey, *Proceedings of the Eighth Annual Software Engineering Workshop*, SEL-83-007, November 1983
	1984	6.14	*Structural Coverage of Functional Testing*, TR-1442, V. R. Basili and J. Ramsey, University of Maryland, Technical Report, September 1984
	1985	7.7	"Analyzing the Test Process Using Structural Coverage," J. Ramsey and V. R. Basili, *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985

Author	Year	Section	Title
Raskin, A.	1985	2.20	"DEASEL: An Expert System for Software Engineering," J. Valett and A. Raskin, *Proceedings of the Tenth Annual Software Engineering Workshop*, SEL-85-006, December 1985
Reed, K.	1987	8.6	"A Structure Coverage Tool for Ada™ Software Systems," L. Wu, V. R. Basili, and K. Reed, *Proceedings of the Joint Ada Conference*, March 1987
Reiter, R. W., Jr.	1978	2.14	"Investigating Software Development Approaches: A Synopsis," R. W. Reiter, Jr., *Proceedings from the Third Summer Software Engineering Workshop*, SEL-78-005, September 1978
	1979	6.4	"Evaluating Automatable Measures for Software Development," V. R. Basili and R. Reiter, *Proceedings of the Workshop on Quantitative Software Models for Reliability, Complexity, and Cost.* New York: IEEE Computer Society Press, 1979
Rohleder, M. G.	1983	2.32	*Glossary of Software Engineering Laboratory Terms*, SEL-82-105, T. A. Babst, M. G. Rohleder, and F. E. McGarry, November 1983
Rombach, H. D.	1984	2.20	"Design Metrics for Maintenance," H. D. Rombach, *Proceedings of the Ninth Annual Software Engineering Workshop*, SEL-84-004, November 1984
	1986	2.22	"TAME: Tailoring A Measurement Environment," V. R. Basili and H. D. Rombach, *Proceedings of the Eleventh Annual Software Engineering Workshop*, SEL-86-006, December 1986
	1987	5.19	"Tailoring the Software Process to Project Goals and Environments," V. R. Basili and H. D. Rombach, *Proceedings of the 9th International Conference on Software Engineering*, March 1987
	1987	5.21	*A Meta Information Base for Software Engineering*, L. Mark and H. D. Rombach, TR-1765, University of Maryland, Technical Report, July 1987

Author	Year	Section	Title
Rombach, H. D. (Cont'd)	1987	6.20	"A Controlled Experiment on the Impact of Software Structure on Maintainability," H. D. Rombach, *IEEE Transactions on Software Engineering*, March 1987
	1987	6.21	*TAME: Integrating Measurement Into Software Environments*, V. R. Basili and H. D. Rombach, TR-1764, University of Maryland, Technical Report, June 1987
	1987	7.20	"Quantitative Assessment of Maintenance: An Industrial Case Study," H. D. Rombach and V. R. Basili, *Proceedings from the Conference on Software Maintenance*, September 1987
	1987	8.7	"TAME: Tailoring an Ada™ Measurement Environment," V. R. Basili and H. D. Rombach, *Proceedings of the Joint Ada Conference*, March 1987
	1988	2.24	"Towards a Comprehensive Framework for Reuse: A Reuse-Enabling Software Evolution Environment," V. R. Basili and H. D. Rombach, *Proceedings of the Thirteenth Annual Software Engineering Workshop*, SEL-88-004, November 1988
	1988	5.25	*Towards a Comprehensive Framework for Reuse: A Reuse-Enabling Software Evolution Environment*, V. Basili and H. Rombach, TR-2158, University of Maryland, Technical Report, December 1988
	1988	6.24	"The TAME Project: Towards Improvement-Oriented Software Environments," V. R. Basili and H. D. Rombach, *IEEE Transactions on Software Engineering*, June 1988
	1989	2.25	"Measurement Based Improvement of Maintenance in the SEL," H. D. Rombach and B. T. Ulery, *Proceedings of the Fourteenth Annual Software Engineering Workshop*, SEL-89-007, November 1989
	1989	6.29	*Establishing a Measurement Based Maintenance Improvement Program: Lessons Learned in the SEL*, H. Rombach and B. Ulery, TR-2252, University of Maryland Technical Report, May 1989

Author	Year	Section	Title
Rombach, H. D. (Cont'd)	1989	5.22	"Generating Customized Software Engineering Information Bases from Software Process and Product Specifications," L. Mark and H. D. Rombach, *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, January 1989
	1989	5.23	"Software Process and Product Specifications: A Basis for Generating Customized SE Information Bases," H. D. Rombach and L. Mark, *Proceedings of the 22nd Annual Hawaii International Conference on System Sciences*, January 1989
	1990	5.28	*Towards a Comprehensive Framework for Reuse: Model-Based Reuse Characterization Schemes*, V. Basili and H. Rombach, TR-2446, University of Maryland Technical Report, April 1990
	1990	6.31	"Design Measurement: Some Lessons Learned," H. Rombach, *IEEE Software*, March 1990
	1991	5.30	"Software Reuse: A Key to the Maintenance Problem," H. D. Rombach, *Butterworth Journal of Information and Software Technology*, January/ February 1991
	1991	5.31	*Support for Comprehensive Reuse*, V. R. Basili and H. D. Rombach, *Software Engineering Journal*, September 1991
	1992	6.34	"Toward Full Life Cycle Control: Adding Maintenance Measurement to the SEL," H. D. Rombach, B. T. Ulery, and J. D. Valett, *Journal of Systems and Software*, May 1992
Scheffer, P. A.	1978	7.1	*GSFC Software Engineering Research Requirements Analysis Study*, SEL-78-006, P. A. Scheffer and C. E. Velez, November 1978
Schneiderman, B.	1983	2.19	"Evaluating Multiple Coordinated Windows for Programming Workstations," B. Schneiderman, C. Grantham, K. Norman, et al., *Proceedings of the Eighth Annual Software Engineering Workshop*, SEL-83-007, November 1983

Author	Year	Section	Title
Seidewitz, E.	1986	8.3	"Towards a General Object-Oriented Software Development Methodology," E. Seidewitz and M. Stark, *Proceedings of the First International Symposium on Ada for the NASA Space Station*, June 1986
	1986	8.4	*General Object-Oriented Software Development*, SEL-86-002, E. Seidewitz and M. Stark, August 1986
	1987	8.5	"Towards a General Object-Oriented Ada Lifecycle," M. Stark and E. Seidewitz, *Proceedings of the Joint Ada Conference*, March 1987
	1987	8.9	*Ada® Style Guide (Version 1.1)*, SEL-87-002, E. Seidewitz, May 1987
	1987	8.12	"Object-Oriented Programming in Smalltalk and Ada," E. Seidewitz, *Proceedings of the 1987 Conference on Object-Oriented Programming Systems, Languages and Applications*, October 1987
	1988	8.14	"General Object-Oriented Software Development: Background and Experience," E. Seidewitz, *Proceedings of the 21st Hawaii International Conference on System Sciences*, January 1988
	1988	8.16	"General Object-Oriented Software Development with Ada: A Life Cycle Approach," E. Seidewitz, *Proceedings of the CASE Technology Conference*, April 1988
	1991	8.32	"Object-Oriented Programming Through Type Extensions in Ada 9X," E. Seidewitz, *Ada Letters*, March/April 1991
	1991	8.33	"An Object-Oriented Approach to Parameterized Software in Ada," E. Seidewitz and M. Stark, *Proceedings of the Eighth Washington Ada Symposium*, June 1991
	1992	8.36	"Object-Oriented Programming with Mixins in Ada," E. Seidewitz, *Ada Letters*, March/April 1992
Seigle, J.	1988	8.18	*System Testing of a Production Ada Project—The GRODY Study*, SEL-88-001, J. Seigle and Y. Shi, November 1988

Author	Year	Section	Title
Selby, R.W., Jr.	1983	6.9	"Metric Analysis and Data Validation Across FORTRAN Projects," V. R. Basili, R. W. Selby, and T. Phillips, *IEEE Transactions on Software Engineering*, November 1983
	1984	2.20	"Evaluating Software Testing Strategies," R. W. Selby, Jr., and V. R. Basili, *Proceedings of the Ninth Annual Software Engineering Workshop*, SEL-84-004, November 1984
	1985	6.17	"Calculation and Use of an Environment's Characteristic Software Metric Set," V. R. Basili and R. W. Selby, Jr., *Proceedings of the Eighth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1985
	1985	7.9	*Comparison of Software Verification Techniques*, SEL-85-001, D. N. Card, R. W. Selby, F. E. McGarry, et al., April 1985
	1985	7.10	*Evaluations of Software Technologies: Testing, Cleanroom, and Metrics*, SEL-85-004, R. W. Selby, Jr., May 1985
	1985	7.12	"Four Applications of a Software Data Collection and Analysis Methodology," V. R. Basili and R. W. Selby, Jr., *Proceedings of the NATO Advanced Study Institute*, August 1985
	1986	5.16	"Experimentation in Software Engineering," V. R. Basili, R. W. Selby, Jr., and D. H. Hutchens, *IEEE Transactions on Software Engineering*, July 1986
	1987	7.21	"Comparing the Effectiveness of Software Testing Strategies," V. R. Basili and R. W. Selby, *IEEE Transactions on Software Engineering*, December 1987
	1991	6.33	"Paradigms for Experimentation and Empirical Studies in Software Engineering," V. R. Basili and R. W. Selby, *Reliability Engineering and System Safety*, January 1991
Shi, Y.	1988	8.18	*System Testing of a Production Ada Project—The GRODY Study*, SEL-88-001, J. Seigle and Y. Shi, November 1988

Author	Year	Section	Title
Smith, L.	1989	8.24	*Evolution of Ada Technology in the Flight Dynamics Area: Implementation/Testing Phase Analysis*, SEL-89-004, K. Quimby, L. Esker, L. Smith, M. Stark, and F. McGarry, November 1989
Stark, M. E.	1985	8.1	*Ada Training Evaluation and Recommendations From the Gamma Ray Observatory Ada Development Team*, SEL-85-002, R. Murphy and M. Stark, October 1985
	1986	8.3	"Towards a General Object-Oriented Software Development Methodology," E. Seidewitz and M. Stark, *Proceedings of the First International Symposium on Ada for the NASA Space Station*, June 1986
	1986	8.4	*General Object-Oriented Software Development*, SEL-86-002, E. Seidewitz and M. Stark, August 1986
	1987	8.5	"Towards a General Object-Oriented Ada Lifecycle," M. Stark and E. Seidewitz, *Proceedings of the Joint Ada Conference*, March 1987
	1989	8.23	"Using Ada to Maximize Verbatim Software Reuse," M. Stark and E. Booth, *Proceedings of Tri-Ada 1989*, October 1989
	1989	8.24	*Evolution of Ada Technology in the Flight Dynamics Area: Implementation/Testing Phase Analysis*, SEL-89-004, K. Quimby, L. Esker, L. Smith, M. Stark, and F. McGarry, November 1989
	1990	8.27	"On Designing Parametrized Systems Using Ada," M. Stark, *Proceedings of the Seventh Washington Ada Symposium*, June 1990
	1991	8.33	"An Object-Oriented Approach to Parameterized Software in Ada," E. Seidewitz and M. Stark, *Proceedings of the Eighth Washington Ada Symposium*, June 1991
	1991	8.34	*Software Engineering Laboratory (SEL) Ada Performance Study Report*, SEL-91-003, E. W. Booth and M. E. Stark, July 1991

Author	Year	Section	Title
Stark, M. E. (Cont'd)	1991	8.35	"Designing Configurable Software: COMPASS Implementation Concepts," E. W. Booth and M. E. Stark, *Proceedings of Tri-Ada 1991*, October 1991
	1992	8.37	"Software Engineering Laboratory Ada Performance Study—Results and Implications," E. W. Booth and M. E. Stark, *Proceedings of the Fourth Annual NASA Ada User's Symposium*, April 1992
	1992	2.28	"Impacts of Object-Oriented Technologies: Seven Years of SEL Studies," M. Stark, *Proceedings of the Seventeenth Annual Software Engineering Workshop*, SEL-92-004, December 1992
	1993	7.25	"Impacts of Object-Oriented Technologies: Seven Years of SEL Studies," M. Stark, *Proceedings of the Conference on Object-Oriented Programming Systems, Languages, and Applications*, September 1993
Straub, P. A.	1990	2.26	"Bias and Design in Software Specifications," P. A. Straub and M. V. Zelkowitz, *Proceedings of the Fifteenth Annual Software Engineering Workshop*, SEL-90-006, November 1990
	1990	8.28	"PUC: A Functional Specification Language for Ada," P. Straub and M. Zelkowitz, *Proceedings of the Tenth International Conference of the Chilean Computer Science Society*, July 1990
	1992	5.36	"On the Nature of Bias and Defects in the Software Specification Process," P. A. Straub and M. V. Zelkowitz, *Proceedings of the Sixteenth International Computer Software and Applications Conference (COMPSAC 92)*, September 1992
Sunazuka, T	1989	6.30	*Integrating Automated Support for a Software Management Cycle Into the TAME System*, V. Basili and T. Sunazuka, TR-2289, University of Maryland, Technical Report, July 1989
Sukri, J.	1983	2.19	"Characteristics of a Prototyping Experimenting," J. Sukri and M. V. Zelkowitz, *Proceedings of the Eighth Annual Software Engineering Workshop*, SEL-83-007, November 1983

Author	Year	Section	Title
Taylor, W. A.	1985	4.1	*FORTRAN Static Source Code Analyzer Program (SAP) System Description (Revision 1)*, SEL-82-102, W. A. Taylor and W. J. Decker, April 1985
	1986	4.2	*FORTRAN Static Source Code Analyzer Program (SAP). User's Guide (Revision 3)*, SEL-78-302, W. J. Decker and W. A. Taylor, July 1986
Thomas, W. M.	1991	5.33	*A Pattern Recognition Approach for Software Engineering Data Analysis*, L. C. Briand, V. R. Basili, and W. M. Thomas, TR-2672, University of Maryland Technical Report, May 1991
	1993	5.41	"Modeling and Managing Risk Early in Software Development," L. C. Briand, W. M. Thomas, and C. J. Hetmanski, *Proceedings of the Fifteenth International Conference on Software Engineering (ICSE 93)*, May 1993
Tian, J.	1992	5.37	"An Improved Classification Tree Analysis of High Cost Modules Based Upon an Axiomatic Definition of Complexity," J. Tian, A. Porter, and M. V. Zelkowitz, *Proceedings of the Third IEEE International Symposium on Software Reliability Engineering (ISSRE 92)*, October 1992
Turner, C.	1981	9.1	*A Comparison of RADC and NASA/SEL Software Development Data*, C. Turner and G. Caron, Data & Analysis Center for Software, May 1981
Ulery, B. T.	1989	2.25	"Measurement Based Improvement of Maintenance in the SEL," H. D. Rombach and B. T. Ulery, *Proceedings of the Fourteenth Annual Software Engineering Workshop*, SEL-89-007, November 1989
	1989	6.29	*Establishing a Measurement Based Maintenance Improvement Program: Lessons Learned in the SEL*, H. Rombach and B. Ulery, TR-2252, University of Maryland Technical Report, May 1989
	1992	6.34	"Toward Full Life Cycle Control: Adding Maintenance Measurement to the SEL," H. D. Rombach, B. T. Ulery, and J. D. Valett, *Journal of Systems and Software*, May 1992

Author	Year	Section	Title
Valett, J. D.	1985	2.21	"DEASEL: An Expert System for Software Engineering," J. Valett and A. Raskin, *Proceedings of the Tenth Annual Software Engineering Workshop*, SEL-85, 006, December 1985
	1985	7.8	"Measuring the Impact of Computer Resource Quality on the Software Development Process and Product," F. E. McGarry, J. Valett, and D. Hall, *Proceedings of the Hawaii International Conference on System Sciences*, January 1985
	1986	2.22	"Determining Software Productivity Leverage Factors in the SEL," F. McGarry, S. Voltz, and J. Valett, *Proceedings of the Eleventh Annual Software Engineering Workshop*, SEL-86-006, December 1986
	1988	2.24	"The Software Management Environment," J. D. Valett, W. Decker, and J. Buell, *Proceedings of the Thirteenth Annual Software Engineering Workshop*, SEL-88-004, November 1988
	1988	6.28	"A Summary of Software Measurement Experiences in the Software Engineering Laboratory," J. D. Valett and F. E. McGarry, *Proceedings of the 21st Annual Hawaii International Conference on System Sciences*, January 1988
	1989	4.4	*Software Management Environment (SME) Concepts and Architecture (Revision 1)*, SEL-89-103, R. Hendrick, D. Kistler, and J. Valett, September 1992
	1991	6.32	*Software Engineering Laboratory (SEL) Relationships, Models, and Management Rules*, SEL-91-001, W. Decker, R. Hendrick, and J. Valett, February 1991
	1992	4.7	"Automated Support for Experience-Based Software Management," J. D. Valett, *Proceedings of the Second Irvine Software Symposium (ISS '92)*, March 1992
	1992	6.34	"Toward Full Life Cycle Control: Adding Maintenance Measurement to the SEL," H. D. Rombach, B. T. Ulery, and J. D. Valett, *Journal of Systems and Software*, May 1992

Author	Year	Section	Title
Valett, J. D. (Cont'd)	1992	9.7	*Data Collection Procedures for the Software Engineering Laboratory (SEL) Database*, SEL-92-002, G. Heller, J. Valett, and M. Wild, March 1992
Velez, C. E.	1978	7.1	*GSFC Software Engineering Research Requirements Analysis Study*, SEL-78-006, P. A. Scheffer and C. E. Velez, November 1978
Voltz, S.	1986	2.22	"Determining Software Productivity Leverage Factors in the SEL," F. McGarry, S. Voltz, and J. Valett, *Proceedings of the Eleventh Annual Software Engineering Workshop*, SEL-86-006, December 1986
Waligora, S. R.	1989	2.25	"Experiences in the Software Engineering Laboratory (SEL) Applying Software Measurement," F. McGarry, S. Waligora, and T. McDermott, *Proceedings of the Fourteenth Annual Software Engineering Workshop*, SEL-89-007, November 1989
	1990	3.2	*Manager's Handbook for Software Development (Revision 1)*, SEL-84-101, L. Landis, F. E. McGarry, S. Waligora, et al., November 1990
	1991	2.27	"Experiments in Software Engineering Technology," F. McGarry and S. Waligora, *Proceedings of the Sixteenth Annual Software Engineering Workshop*, SEL-91-006, December 1991
	1992	3.1	*Recommended Approach to Software Development (Revision 3)*, SEL-81-305, L. Landis, S. Waligora, F. McGarry, et al., June 1992
	1992	2.28	"Maximizing Reuse: Applying Common Sense and Discipline," S. Waligora and J. Langston, *Proceedings of the Seventeenth Annual Software Engineering Workshop*, SEL-92-004, December 1992
Weiss, D. M.	1981	6.26	*Evaluating Software Development by Analysis of Change Data*, SEL-81-011, D. M. Weiss, November 1981
	1982	6.27	*Evaluating Software Development by Analysis of Changes: The Data From the Software Engineering Laboratory*, SEL-82-008, V. R. Basili and D. M. Weiss, December 1982

Author	Year	Section	Title
Weiss, D. M. (Cont'd)	1982	9.5	*A Methodology for Collecting Valid Software Engineering Data*, V. R. Basili and D. M. Weiss, University of Maryland, Technical Report, December 1982
	1984	9.6	"A Methodology for Collecting Valid Software Engineering Data," V. R. Basili and D. M. Weiss, *IEEE Transactions on Software Engineering*, November 1984
	1985	6.18	"Evaluating Software Development by Analysis of Changes: Some Data From the Software Engineering Laboratory," D. M. Weiss and V. R. Basili, *IEEE Transactions on Software Engineering*, February 1985
Wild, M.	1992	9.7	*Data Collection Procedures for the Software Engineering Laboratory (SEL) Database*, SEL-92-002, G. Heller, J. Valett, and M. Wild, March 1992
Wood, R.	1986	3.3	*Programmer's Handbook for Flight Dynamics Software Development*, SEL-86-001, R. Wood and E. Edwards, March 1986
Wu, L.	1987	8.6	"A Structure Coverage Tool for Ada™ Software Systems," L. Wu, V. R. Basili, and K. Reed, *Proceedings of the Joint Ada Conference*, March 1987
Zelkowitz, M. V.	1976	2.12	"Automated Tools," M. V. Zelkowitz, *Proceedings From the First Summer Software Engineering Workshop*, SEL-76-001, August 1976
	1977	2.13	"Overview of the Software Engineering Laboratory," V. R. Basili and M. V. Zelkowitz, *Proceedings from the Second Summer Software Engineering Workshop*, SEL-77-002, September 1977
	1977	2.29	"The Software Engineering Laboratory: Objectives," V. R. Basili and M. V. Zelkowitz, *Proceedings of the Fifteenth Annual Conference on Computer Personnel Research*, August 1977
	1977	6.1	"Designing a Software Measurement Experiment," V. R. Basili and M. V. Zelkowitz, *Proceedings of the Software Life Cycle Management Workshop*, September 1977

Author	Year	Section	Title
Zelkowitz, M. V. (Cont'd)	1978	2.14	"The Software Engineering Laboratory—1978," V. R. Basili and M. V. Zelkowitz, *Proceedings from the Third Summer Software Engineering Workshop*, SEL-78-005, September 1978
	1978	2.30	"Operation of the Software Engineering Laboratory," V. R. Basili and M. V. Zelkowitz, *Proceedings of the Second Software Life Cycle Management Workshop*, August 1978
	1978	6.2	"Analyzing Medium Scale Software Development," V. R. Basili and M. V. Zelkowitz, *Proceedings of the Third International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1978
	1978	6.3	"Measuring Software Development Characteristics in the Local Environment," V. R. Basili and M. V. Zelkowitz, *Computers and Structures*, August 1978, Vol. 10
	1979	2.15	"Software Engineering Laboratory: Data Validation," M. V. Zelkowitz and E. Chen, *Proceedings from the Fourth Summer Software Engineering Workshop*, SEL-79-005, November 1979
	1979	5.2	"Resource Estimation for Medium-Scale Software Projects," M. V. Zelkowitz, *Proceedings of the Twelfth Conference on the Interface of Statistics and Computer Science*. New York: IEEE Computer Society Press, 1979
	1981	7.4	"Use of Cluster Analysis to Evaluate Software Engineering Methodologies," E. Chen and M. V. Zelkowitz, *Proceedings of the Fifth International Conference on Software Engineering*. New York: IEEE Computer Society Press, 1981
	1982	2.18	"Software Prototyping in the Software Engineering Laboratory," M. V. Zelkowitz, *Proceedings of the Seventh Annual Software Engineering Workshop*, SEL-82-007, December 1982
	1982	9.4	"Data Collection and Evaluation for Experimental Computer Science Research," M. V. Zelkowitz, *Empirical Foundations for Computer and Information Science* (Proceedings), November 1982

Author	Year	Section	Title
Zelkowitz, M. V. (Cont'd)	1983	2.19	"Characteristics of a Prototyping Experiment," J. Sukri and M. V. Zelkowitz, *Proceedings of the Eighth Annual Software Engineering Workshop*, SEL-83-007, November 1983
	1986	2.23	"Automating the Design Process with Syntactic-Based Tools," M. V. Zelkowitz, *Proceedings of the Eleventh Annual Software Engineering Workshop*, SEL-86-006, December 1986
	1987	7.18	"The Effectiveness of Software Prototyping: A Case Study," M. V. Zelkowitz, *Proceedings of the Washington, D.C., Chapter of the ACM*, June 1987
	1988	6.22	"Resource Utilization During Software Development," M. V. Zelkowitz, *Journal of Systems and Software*, 1988
	1990	2.26	"Bias and Design in Software Specifications," P. A. Straub and M. V. Zelkowitz, *Proceedings of the Fifteenth Annual Software Engineering Workshop*, SEL-90-006, November 1990
	1990	7.23	"Evolution Towards Specifications Environment: Experiences With Syntax Editors," M. Zelkowitz, *Information and Software Technology*, April 1990
	1990	8.28	"PUC: A Functional Specification Language for Ada," P. Straub and M. Zelkowitz, *Proceedings of the Tenth International Conference of the Chilean Computer Science Society*, July 1990
	1992	5.36	"On the Nature of Bias and Defects in the Software Specification Process," P. A. Straub and M. V. Zelkowitz, *Proceedings of the Sixteenth International Computer Software and Applications Conference (COMPSAC 92)*, September 1992
	1992	5.37	"An Improved Classification Tree Analysis of High Cost Modules Based Upon an Axiomatic Definition of Complexity," J. Tian, A. Porter, and M. V. Zelkowitz, *Proceedings of the Third IEEE International Symposium on Software Reliability Engineering (ISSRE 92)*, October 1992

Author	Year	Section	Title
Zelkowitz, M. V. (Cont'd)	1993	5.42	"An Information Model for Use in Software Management Estimation and Prediction," N. R. Li and M. V. Zelkowitz, *Proceedings of the Second International Conference on Information and Knowledge Management*, November 1993