

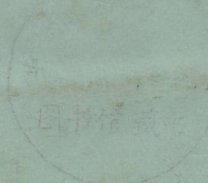
8961384

SOFTWARE ENGINEERING

A Practitioner's Approach

Second Edition

Roger S. Pressman, Ph.D.



TP31
P935
E.2

8961384

SOFTWARE ENGINEERING

A Practitioner's Approach

Second Edition



8961384

Roger S. Pressman, Ph.D.

*President, R.S. Pressman & Associates, Inc.
and
Adjunct Professor of Computer Engineering
University of Bridgeport*



McGraw-Hill Book Company

New York St. Louis San Francisco Auckland Bogotá Hamburg Johannesburg
London Madrid Mexico Milan Montreal New Delhi Panama
Paris São Paulo Singapore Sydney Tokyo Toronto

1851888

This book was set in Times Roman by Publication Services.
The editor was Gerald A. Gleason;
the production supervisor was Phil Galea;
the cover was designed by Infield + D'Astolfo.
Project supervision was done by Publication Services.
R.R. Donnelley & Sons Company was the printer and binder.

SOFTWARE ENGINEERING

A Practitioner's Approach

Copyright © 1987, 1982 by McGraw-Hill, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the United States Copyright Act of 1976, no part of this publication may be reproduced or distributed in any form or by any means, or stored in a data base or retrieval system, without the prior written permission of the publisher.

1 2 3 4 5 6 7 8 9 0 DOCDOC 8 9 4 3 2 1 0 9 8 7

ISBN 0-07-050783-X

Library of Congress Cataloging-in-Publication Data
Pressman, Roger S.

Software engineering.

Includes bibliographies and index.

I. Electronic digital computers—Programming.

J. Title.

QA76.6.P73 1987 005.1 86-18549

ISBN 0-07-050783-X

SOFTWARE ENGINEERING

A Practitioner's Approach

McGraw-Hill Series in Software Engineering and Technology

Consulting Editor

Peter Freeman, *University of California, Irvine*

Cohen: *Ada as a Second Language*

Fairley: *Software Engineering Concepts*

Howden: *Functional Program Testing and Analysis*

Jones: *Programming Productivity*

Kolence: *An Introduction to Software Physics: The Meaning of Computer Measurement*

Musa, Iannino and Okumoto: *Software Reliability: Measurement, Prediction and Application*

Pressman: *Software Engineering: A Practitioner's Approach*

ABOUT THE AUTHOR

Roger S. Pressman is a nationally recognized industry consultant in software engineering. He received a B.S.E. from the University of Connecticut, an M.S. from the University of Bridgeport, and a Ph.D. in Engineering from the University of Connecticut. He has had nearly two decades of industry experience, holding both technical and management positions with responsibility for the development of software for engineered products and systems.

In addition to his industry experience, Dr. Pressman was Bullard Associate Professor of Computer Engineering at the University of Bridgeport and Director of the University's Computer Aided Design and Manufacturing Center. He remains at the university as an adjunct Associate Professor of Computer Engineering.

Dr. Pressman is President of R.S. Pressman & Associates, Inc., a consulting firm specializing in management and technical problem solving in software engineering. Clients include many large corporations as well as smaller high technology companies. The firm specializes in software engineering training and has developed courses and other training products that have been used by over 10,000 industry professionals.

Dr. Pressman is author of many technical papers and another book, *Numerical Control and Computer Aided Manufacturing*. He is a member of the ACM, IEEE, and Tau Beta Pi, Phi Kappa Phi, Pi Tau Sigma, and Eta Kappa Nu.

TO MY PARENTS

PREFACE

In the five years since the first edition of *Software Engineering: A Practitioner's Approach*, software engineering has grown from infancy to early adolescence. Today, software engineering is recognized as a legitimate discipline and "software engineer" has replaced "programmer" as the job title of preference. Software engineering methods and procedures have been adopted successfully in a broad spectrum of industry applications. Managers and practitioners alike recognize the need for a more disciplined approach to software development.

But the problems that were described in the preface to the first edition remain with us. Many companies still develop software haphazardly. Many professionals and students are unaware of modern methods. In addition, debate and controversy about the true nature of the software engineering approach continue. The status of software engineering is a study in contrasts. Attitudes have changed, progress has been made, but much remains to be done before the discipline reaches maturity.

The second edition of *Software Engineering: A Practitioner's Approach* is intended to provide one element of a foundation from which the bridge from adolescence to maturity can be constructed. The second edition, like the first, is intended for both students and practitioners, maintaining the same format and style of the first edition. The book retains its appeal as a guide to the industry professional and a comprehensive introduction to the student at the upper level undergraduate or first year graduate level.

Like the first edition, software engineering methods are presented in the chronological sequence that they are applied during software development. However, the second edition is more than a simple update. The book has been restructured to emphasize new and important software engineering methods and techniques. Rather than maintaining a strict life cycle view, the second edition presents generic activities that are performed regardless of the software engineering paradigm that has been chosen.

Chapters that have been retained from the first edition have been expanded and revised to reflect current trends and techniques. Major new sections have been added to chapters on computer system engineering, software project planning, analysis methodologies, programming languages and coding, testing strategies and software maintenance.

New chapters on analysis and design fundamentals have been added to provide a foundation for the methods that are introduced in later chapters. In addition, new chapters present object-oriented design, real-time design, software test case design techniques, and software quality assurance. These chapters have been added to reflect new software engineering methods that are rapidly gaining acceptance in the industry. Many new examples, problems, and points to ponder have been added and the *Further Readings* sections (one of the more popular tidbits in the first edition) have been expanded and updated for every chapter.

The software engineering literature is expanding at an explosive rate. Once again, my thanks to the many authors who have provided additional insight, ideas, and commentary in the years since the first edition. Many have been referenced within the pages of each chapter. All deserve credit for their contribution to this rapidly evolving field. I also wish to thank the reviewers of the second edition: Robert Glass, Seattle University; Ernest H. Goldman, University of Bridgeport; Medi-Harandi, University of Illinois-Urbana-Champaign; Marvin Zelkowitz, University of Maryland; and John Musa, ATT Bell Labs. Their comment and criticism have been invaluable.

The content of the second edition of *Software Engineering: A Practitioner's Approach* has been shaped by the hundreds of industry professionals, university professors and students who have taken the time to communicate their suggestions, criticisms and ideas. In addition, my personal thanks go to our many industry clients, who certainly teach me as much or more than I can teach them.

Finally, to Barbara, Mathew, and Michael, my love and thanks for tolerating my travel schedule, understanding the evenings at the office, and encouraging the second edition of "the book."

Roger S. Pressman

PREFACE TO THE FIRST EDITION

In the brief history of the electronic digital computer, the 1950s and 1960s were decades of hardware. The 1970s were a period of transition and a time of recognition of software. The decade of software is now upon us. In fact, advances in computing may become limited by our inability to produce quality software that can tap the enormous capacity of 1980-era processors.

During the past decade we have grown to recognize circumstances that are collectively called the *software crisis*. Software costs escalated dramatically, becoming the largest dollar item in many computer-based systems. Schedules and completion dates were set but rarely kept. As software systems grew larger, quality became suspect. Individuals responsible for software development projects had limited historical data to use as guides and less control over the course of a project.

A set of techniques, collectively called *software engineering*, has evolved as a response to the software crisis. These techniques deal with software as an engineered product that requires planning, analysis, design, implementation, testing, and maintenance. The goal of this text is to provide a concise presentation of each step in the software engineering process.

The contents of this book closely parallel the software life cycle. Early chapters present the planning phase, emphasizing system definition (computer systems engineering), software planning, and software requirements analysis. Specific techniques for software costs and schedule estimation should be of particular interest to project managers as well as to technical practitioners and students.

In subsequent chapters emphasis shifts to the software development phase. The fundamental principles of software design are introduced. In addition, descriptions of two important classes of software design methodology are presented in detail. A variety of software tools are discussed. Comparisons among techniques and among tools are provided to assist the practitioner and student alike. Coding style is also stressed in the context of the software engineering process.

The concluding chapters deal with software testing techniques, reliability, and software maintenance. Software engineering steps associated with testing are de-

scribed and specific techniques for software testing are presented. The current status of software reliability prediction is discussed and an overview of reliability models and program correctness approaches is presented. The concluding chapter considers both management and technical aspects of software maintenance.

This book is an outgrowth of a senior-level/first-year-graduate course in software engineering offered at the University of Bridgeport. The course and this text cover both management and technical aspects of the software development process. The chapters of the text correspond roughly to major lecture topics. In fact, the text is derived in part from edited versions of transcribed notes of these lectures. Writing style is therefore purposely casual and figures are derived from viewgraphs used during the course.

Software Engineering: A Practitioner's Approach may be used in a number of ways for various audiences. The text can serve as a concise guide to software engineering for the practicing manager, analyst, or programmer. It can also serve as the basic text for an upper-level undergraduate or graduate course in software engineering. Lastly, the text can be used as a supplementary guide for software development early in computer science or computer engineering undergraduate curricula.

The software engineering literature has expanded rapidly during the past decade. I gratefully acknowledge the many authors who have helped this new discipline evolve. Their work has had an important influence on this book and my method of presentation. I also wish to acknowledge Pat Duran, Leo Lambert, Kyu Lee, John Musa, Claude Walston, Anthony Wasserman, Marvin Zeligowitz, and Nicholas Zvegintzov, the reviewers of this book, and Peter Freeman, the series editor. Their thoughtful insights and suggestions have been invaluable during the final stages of preparation. Special thanks go to Leo Lambert and his colleagues from the Computer Management Operation, General Electric Company, who have allowed me to tap their broad collective experience during my long association with them. In addition, to the students at the University of Bridgeport and the hundreds of software professionals and their managers who have attended short courses that I have taught, my thanks for the arguments, the ideas, and the challenges that are essential in a field such as ours.

Finally, to Barbara, Mathew, and Michael, my love and thanks for tolerating the genesis of book number two.

Roger S. Pressman

CONTENTS

Preface	xvii
---------	------

Preface to the First Edition	xix
------------------------------	-----

1 Software and Software Engineering

1.1 The Importance of Software	1
1.1.1 The Evolving Role of Software	2
1.1.2 An Industry Perspective	4
1.2 Software	5
1.2.1 Software Characteristics	5
1.2.2 Software Components	8
1.2.3 Software Applications	11
1.3 The Software Crisis	13
1.3.1 Problems	13
1.3.2 Causes	14
1.4 Software Myths	15
1.4.1 Management Myths	15
1.4.2 Customer Myths	16
1.4.3 Practitioner's Myths	17
1.5 Software Engineering Paradigms	19
1.5.1 Software Engineering: A Definition	19
1.5.2 The Classic Life Cycle	20
1.5.3 Prototyping	22
1.5.4 Fourth Generation Techniques	24
1.5.5 Combining Paradigms	26
1.6 A Generic View of Software Engineering	27
1.7 Summary	29

2 Computer System Engineering

2.1 Computer-Based Systems	33
2.2 Computer System Engineering	35
2.3 Hardware Considerations	38
2.3.1 The Hardware Element	38
2.3.2 Hardware Applications	41
2.3.3 Hardware Engineering	42

2.4	Software Considerations	44
2.4.1	The Software Element	44
2.4.2	Software Applications	44
2.4.3	Software Engineering	45
2.5	Human Considerations	50
2.5.1	The Human Element	51
2.5.2	HMI Applications	52
2.5.3	Human Engineering	53
2.6	Data Base Considerations	53
2.7	System Analysis	53
2.7.1	Identification of Need	54
2.7.2	Feasibility Study	55
2.7.3	Economic Analysis	57
2.7.4	Technical Analysis	61
2.7.5	Allocation and Trade-Offs	63
2.8	A System Analysis Checklist	67
2.9	The System Specification	75
2.10	System Definition Review	76
2.11	Summary	77
3	Software Project Planning	81
3.1	Observations on Estimating	82
3.2	Project Planning Objectives	82
3.3	Software Scope	84
3.4	Resources	85
3.4.1	Human Resources	86
3.4.2	Hardware Resources	86
3.4.3	Software Resources	88
3.5	Metrics for Software Productivity and Quality	89
3.5.1	Measuring Software	90
3.5.2	Size-Oriented Metrics	91
3.5.3	Function-Oriented Metrics	94
3.5.4	Reconciling Different Metrics Approaches	95
3.5.5	Metrics Data Collection	98
3.6	Software Project Estimation	101
3.7	Decomposition Techniques	101
3.7.1	LOC and FP Estimation	103
3.7.2	An Example	106
3.7.3	Effort Estimation	106
3.7.4	An Example	109
3.8	Empirical Estimation Models	110
3.8.1	The COCOMO Model	112
3.8.2	Putnam's Estimation Model	114
3.8.3	Function Point Models	114
3.8.4	A Time-Study Model	117
3.9	Automated Estimation Tools	118
3.10	Software Project Scheduling	118
3.10.1	People-Work Relationships	118
3.10.2	Task Definition and Parallelism	120

3.10.3	Effort Distribution	121
3.10.4	Scheduling Methods	122
3.10.5	A Scheduling Example	123
3.11	Software Acquisition	123
3.12	Organizational Planning	130
3.13	The Software Project Plan	131
3.14	Summary	132

4 Requirements Analysis Fundamentals

4.1	Requirements Analysis	136
4.1.1	Analysis Tasks	137
4.1.2	The Analyst	139
4.2	Problem Areas	140
4.3	Analysis Principles	141
4.3.1	The Information Domain	142
4.3.2	Partitioning	143
4.3.3	Logical and Physical Views	145
4.4	Object-Oriented Analysis	146
4.5	Software Prototyping	148
4.5.1	A Prototyping Scenario	149
4.5.2	Prototyping Methods and Tools	150
4.6	Specification	151
4.6.1	Specification Principles	152
4.6.2	Representation	155
4.6.3	Software Requirements Specification Outline	156
4.7	Specification Review	158
4.8	Summary	159

5 Requirements Analysis Methods

5.1	Requirements Analysis Methodologies	164
5.1.1	Common Characteristics	164
5.1.2	Representative Methods and Tools	164
5.2	Data Flow-Oriented Analysis Methods	165
5.2.1	Data Flow Diagrams	166
5.2.2	Data Dictionary	172
5.2.3	Functional Descriptions	174
5.3	Data Structure-Oriented Methods	175
5.4	Data Structured Systems Development	176
5.4.1	Warnier Diagrams	176
5.4.2	The DSSD Approach	178
5.4.3	Application Context	179
5.4.4	Application Functions	181
5.4.5	Application Results	182
5.4.6	Physical Requirements	183
5.5	Jackson System Development	185
5.5.1	The Entity Action Step	186
5.5.2	The Entity Structure Step	187
5.5.3	The Initial Model Step	188

5.6	Automated Tools for Requirements Analysis	191
5.6.1	SADT	192
5.6.2	SREM	196
5.6.3	PSL/PSA	198
5.6.4	TAGS	200
5.6.5	Analysis Tools—A Summary	200
5.7	Data Base Requirements	201
5.7.1	Data Base Characteristics	201
5.7.2	Analysis Steps	201
5.7.3	Normalization	203
5.7.4	Capacity Analysis	205
5.7.5	Data Diagramming	206
5.8	Summary	208
6	Software Design Fundamentals	
6.1	The Development Phase and Software Design	214
6.2	The Design Process	215
6.2.1	Design and Software Quality	216
6.2.2	The Evolution of Software Design	216
6.3	Design Fundamentals	217
6.3.1	Refinement	217
6.3.2	Software Architecture	218
6.3.3	Program Structure	219
6.3.4	Data Structure	219
6.3.5	Software Procedure	222
6.3.6	Modularity	222
6.3.7	Abstraction	225
6.3.8	Information Hiding	228
6.4	Effective Modular Design	229
6.4.1	Module Types	229
6.4.2	Functional Independence	230
6.4.3	Cohesion	230
6.4.4	Coupling	232
6.5	Data Design	235
6.6	Architectural Design	237
6.7	Procedural Design	238
6.7.1	Structured Programming	238
6.7.2	Graphical Design Tools	239
6.7.3	Tabular Design Tools	243
6.7.4	Program Design Language	246
6.7.5	A PDL Example	251
6.7.6	Comparison of Design Notations	253
6.8	Design Documentation	254
6.9	Summary	257
7	Data Flow—Oriented Design	
7.1	Design and Information Flow	263
7.1.1	Contributors	263
7.1.2	Areas of Application	263

7.2	Design Process Considerations	263
7.2.1	Transform Flow	264
7.2.2	Transaction Flow	264
7.2.3	A Process Abstract	265
7.3	Transform Analysis	265
7.3.1	An Example	266
7.3.2	Design Steps	267
7.4	Transaction Analysis	275
7.4.1	An Example	275
7.4.2	Design Steps	277
7.5	Design Heuristics	283
7.6	Design Postprocessing	287
7.7	Design Optimization	288
7.8	Summary	289
8	Data Structure–Oriented Design	
8.1	Design and Data Structure	293
8.1.1	Contributors	294
8.1.2	Areas of Application	294
8.1.3	Data Structure versus Data Flow Techniques	295
8.2	Design Process Considerations	295
8.3	Jackson System Development	296
8.3.1	JSD Design Steps	297
8.3.2	The Function Step	300
8.3.3	System Timing Step	305
8.3.4	The Implementation Step	306
8.3.5	Procedural Representation	308
8.4	Logical Construction of Programs and Systems	310
8.4.1	The Warnier Diagram	310
8.4.2	LCP Design Approach	311
8.4.3	Detailed Organization	312
8.4.4	Complex Structures	317
8.5	Data Structured Systems Development	322
8.5.1	A Simplified Design Approach	324
8.5.2	Derivation of Logical Output Structure	324
8.5.3	Derivation of Logical Process Structure	326
8.5.4	Complex Process Logic	327
8.6	Summary	330
9	Object-Oriented Design	
9.1	Origins of Object-Oriented Design	335
9.2	Object-Oriented Design Concepts	336
9.2.1	Objects, Operations, and Messages	336
9.2.2	Classes, Instances, and Inheritance	336
9.2.3	Object Descriptions	339
9.3	Object-Oriented Design Methods	340
9.4	Problem Definition	341
9.5	An Informal Strategy	342

9.6	Formalizing the Strategy	343
9.6.1	Objects and Their Attributes	343
9.6.2	Operations Applied to Objects	345
9.6.3	Program Components and Interfaces	348
9.6.4	Graphical Representations for OOD	350
9.6.5	Implementation Detail	353
9.7	An Alternative Approach	356
9.7.1	Design Steps	357
9.7.2	A Design Example	358
9.8	Summary	363
10	Real-Time Design	368
10.1	System Considerations	368
10.2	Real-Time Systems	369
10.2.1	Integration and Performance Issues	370
10.2.2	Interrupt Handling	371
10.2.3	Real-Time Data Bases	372
10.2.4	Real-Time Operating Systems	373
10.2.5	Real-Time Languages	374
10.2.6	Task Synchronization and Communication	375
10.3	Analysis of Real-Time Systems	375
10.3.1	Mathematical Tools for Real-Time System Analysis	379
10.3.2	Formal Analysis Methods for Real-Time Systems	382
10.4	Software Design Methods	382
10.5	A Data Flow-Oriented Design Method	383
10.5.1	Requirements of a Real-Time Systems Design Method	384
10.5.2	DARTS	385
10.5.3	Task Design	387
10.5.4	Example of the DARTS Design Method	390
10.6	Other Real-Time Design Methods	390
10.6.1	Extensions for Data Flow Modeling	398
10.6.2	Alternative Approaches	399
10.7	Summary	
11	Programming Languages and Coding	404
11.1	The Translation Process	404
11.2	Programming Language Characteristics	404
11.2.1	A Psychological View	407
11.2.2	A Syntactic/Semantic Model	407
11.2.3	An Engineering View	409
11.2.4	Choosing a Language	410
11.2.5	Programming Languages and Software Engineering	411
11.3	Programming Language Fundamentals	412
11.3.1	Data Types and Data Typing	413
11.3.2	Subprograms	413
11.3.3	Control Structures	413