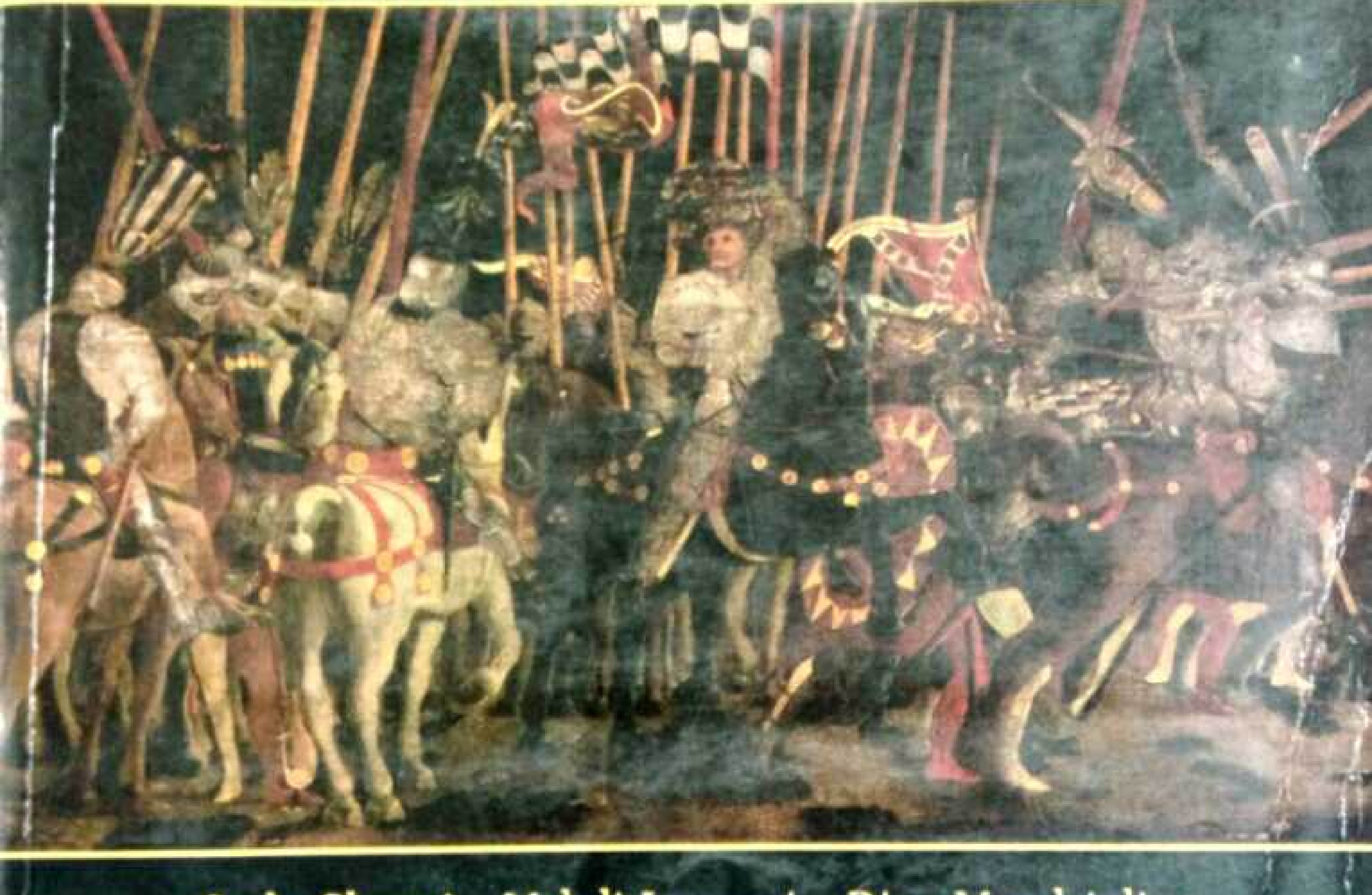
Eastern Economy Edition

FUNDAMENTALS OF

Software Engineering

SECOND EDITION



Carlo Ghezzi · Mehdi Jazayeri · Dino Mandrioli

PHI

[illegible] 22428

Fundamentals of Software Engineering

Second Edition

Carlo Ghezzi
Politecnico di Milano

Mehdi Jazayeri
Technische Universität Wien

Dino Mandrioli
Politecnico di Milano

Prentice-Hall of India Private Limited
New Delhi - 110 001

Contents

Preface to the Second Edition **xiii**
The Role of Object Orientation **xiv**
The Purpose of Case Studies **xiv**
Instructor Resources **xv**

Preface to the First Edition **xvii**
Audience **xviii**
Prerequisites **xviii**
Organization and Content **xviii**
Exercises **xix**
Case Studies **xix**
Laboratory Course **xix**
Reading Graph **xx**
Acknowledgments **xx**

Chapter 1 Software Engineering: A Preview 1
1.1 The Role of Software Engineering in System Design 2
1.2 A Shortened History of Software Engineering 3
1.3 The Role of The Software Engineer 5
1.4 The Software Life Cycle 6
1.5 The Relationship of Software Engineering to Other Areas of Computer Science 8
1.5.1 Programming Languages 9
1.5.2 Operating Systems 10
1.5.3 Data Bases 10
1.5.4 Artificial Intelligence 11
1.5.5 Theoretical Models 12
1.6 The Relationship of Software Engineering to Other Disciplines 12
1.6.1 Management Science 13
1.6.2 Systems Engineering 13
1.7 Concluding Remarks 13
Bibliographic Notes 14

Chapter 2 Software: Its Nature and Qualities

2.1 Classification of Software Qualities [illegible]
2.1.1 External Versus Internal Qualities [illegible]
2.1.2 Product and Process Qualities [illegible]
2.2 Representative Qualities [illegible]
2.2.1 Correctness, Reliability, and Robustness [illegible]
2.2.2 Performance [illegible]
2.2.3 Usability [illegible]
2.2.4 Verifiability 22
2.2.5 Maintainability 23
2.2.6 Reusability 23
2.2.7 Portability [illegible]
2.2.8 Understandability [illegible]
2.2.9 Interoperability [illegible]
2.2.10 Productivity [illegible]
2.2.11 Timeliness [illegible]
2.2.12 Visibility 32
2.3 Quality Requirements in Different Application Areas 33
2.3.1 Information Systems 33
2.3.2 Real-Time Systems 34
2.3.3 Distributed Systems 36
2.3.4 Embedded Systems 36
2.4 Measurement of Quality 37
2.5 Concluding Remarks 38
Further Exercises 38
Hints and Sketchy Solutions 39
Bibliographic Notes 39

Chapter 3 Software Engineering Principles. 41

3.1 Rigor and Formality 42
3.2 Separation of Concerns 44
3.3 Modularity 47
3.4 Abstraction 49
3.5 Anticipation of Change 50
3.6 Generality 52
3.7 Incrementality 53
3.8 Two Case Studies Illustrating Software Engineering Principles 54
3.8.1 Application of Software Engineering Principles to Compiler Construction 54
3.8.2 A Case Study in System Engineering 59
3.9 Concluding Remarks 64
Further Exercises 65

Chapter 4 Design and Software Architecture 67

4.1 The Software Design Activity and its Objectives 70
4.1.1 Design for Change 72
4.1.2 Product Families 76
4.2 Modularization Techniques 78
4.2.1 The Module Structure and its Representation 79
4.2.2 Interface, Implementation, and Information Hiding 86
4.2.3 Design Notations 93
4.2.4 Categories of Modules 100
4.2.5 Some Specific Techniques for Design for Change 108
4.2.6 Stepwise Refinement 111
4.2.7 Top-Down Versus Bottom-Up Design 117
4.3 Handling Anomalies 118
4.4 A Case Study in Design 121
4.5 Concurrent Software 124
4.5.1 Shared Data 124
4.5.2 Real-Time Software 132
4.5.3 Distributed Software 134
4.6 Object-Oriented Design 139
4.6.1 Generalization and Specialization 140
4.6.2 Associations 143
4.6.3 Aggregation 145
4.6.4 More on UML Class Diagrams 146
4.7 Architecture and Components 146
4.7.1 Standard Architectures 147
4.7.2 Software Components 149
4.7.3 Architecture as Framework for Component Integration 152
4.7.4 Architectures for Distributed Systems 153
FSM, 4.8 Concluding Remarks 154
Further Exercises 156
Hints and Sketchy Solutions 158
ERD Bibliographic Notes 159

pter 5 Specification 161

5.1 The Uses of Specifications 162
5.2 Specification Qualities 165
5.3 Classification of Specification Styles 167
5.4 Verification of Specifications 170
5.5 Operational Specifications 171
Flow Diagrams: Specifying Functions of Information Systems 171

5.5.2 UML Diagrams for Specifying Behaviors [illegible]
5.5.3 Finite State Machines: Describing Control Flow [illegible]
5.5.4 Petri Nets: Specifying Asynchronous Systems [illegible]
5.6 Descriptive Specifications [illegible]
5.6.1 Entity-Relationship Diagrams [illegible]
5.6.2 Logic Specifications [illegible]
5.6.3 Algebraic Specifications [illegible]
5.7 Building and Using Specifications in Practice [illegible]
5.7.1 Requirements for Specification Notations [illegible]
5.7.2 Building Modular Specifications [illegible]
5.7.3 Specifications for the End User [illegible]
Concluding Remarks [illegible]
Further Exercises [illegible]
Hints and Sketchy Solutions [illegible]
Bibliographic Notes [illegible]

Chapter 6 Verification 269

6.1 Goals and Requirements of Verification 270
6.1.1 Everything Must Be Verified 271
6.1.2 The Results of Verification May Not Be Binary 271
6.1.3 Verification May Be Objective or Subjective 272
6.1.4 Even Implicit Qualities Must Be Verified 273
6.2 Approaches to Verification 274
6.3 Testing 274
6.3.1 Goals for Testing 275
6.3.2 Theoretical Foundations of Testing 277
6.3.3 Empirical Testing Principles 280
6.3.4 Testing in the Small 282
6.3.5 Testing in the Large 302
6.3.6 Separate Concerns in the Testing Activity 312
6.3.7 Testing Concurrent and Real-Time Systems 313
6.4 Analysis 316
6.4.1 Informal Analysis Techniques 317
6.4.2 Correctness Proofs 320
6.5 Symbolic Execution 337
6.5.1 Basic Concepts of Symbolic Execution 339
6.5.2 Programs with Arrays 342
6.5.3 The Use of Symbolic Execution in Testing 345
6.6 Model Checking 347
6.7 Putting it All Together 349
6.8 Debugging 351
6.9 Verifying Other Software Properties 355
6.9.1 Verifying Performan 356

6.9.2 Verifying Reliability 356
6.9.3 Verifying Subjective Qualities 360
Concluding Remarks 371
Further Exercises 372
Hints and Sketchy Solutions 378
Bibliographic Notes 381

Chapter 7 The Software Production Process 385

7.1 What is a Software Process Model? 386
7.2 Why Are Software Process Models Important? 388
7.3 The Main Activities of Software production 391
7.3.1 Feasibility Study 391
7.3.2 Eliciting, Understanding, and Specifying Requirements 392
7.3.3 Definition of the Software Architecture and Detailed Design 399
7.3.4 Coding and Module Testing 399
7.3.5 Integration and System Testing 400
7.3.6 Delivery, Deployment, and Maintenance 400
7.3.7 Other Activities 401
7.4 An Overview of Software Process Models 403
7.4.1 Waterfall Models 403
7.4.2 Evolutionary Models 410
7.4.3 Transformation Model 413
7.4.4 Spiral Model 416
7.4.5 An Assessment of Process Models 417
7.5 Dealing with Legacy Software 420
7.6 Case Studies 421
7.6.1 Case Study: A Telephone Switching System 421
7.6.2 Case Study: A Budget Control System 426
7.6.3 Case study: The Microsoft Synchronize and Stabilize Process 430
7.6.4 Case study: The Open Source Approach 431
7.7 Organizing the Process 433
7.7.1 Structured Analysis/Structured Design 434
7.7.2 Jackson's System Development and Structured Programming 439
7.7.3 The Unified Software Development Process 444
7.8 Organizing Artifacts: Configuration Management 447
7.9 Software Standards 451
7.10 Concluding Remarks 451
Further Exercises 452
Hints and Sketchy Solutions 453
Bibliographic Notes 454

Chapter 8 Management of Software Engineering 457

8.1 Management Functions 459

8.2 Project Planning [illegible]
8.2.1 Software Productivity [illegible]
8.2.2 People and Productivity [illegible]
8.2.3 Cost Estimation [illegible]
8.3 Project Control [illegible]
8.3.1 Work Breakdown Structures [illegible]
8.3.2 Gantt Charts [illegible]
8.3.3 PERT Charts [illegible]
8.3.4 Dealing with Deviations from the Plan [illegible]
8.4 Organization [illegible]
8.4.1 Centralized-Control Team Organization [illegible]
8.4.2 Decentralized-Control Team Organization [illegible]
8.4.3 Mixed-Control Team Organization [illegible]
8.4.4 An Assessment of Team Organizations [illegible]
8.5 Risk Management [illegible]
8.5.1 Typical Management Risks in Software Engineering [illegible]
8.6 Capability Maturity Model [illegible]
8.7 Concluding Remarks [illegible]
Further Exercises [illegible]
Hints and Sketchy Solutions [illegible]
Bibliographic Notes [illegible]

Chapter 9 Software Engineering Tools and Environments 501

9.1 Historical Evolution of Tools and Environments 502
9.2 Dimensions for Comparing SoftwareTools 503
9.3 Representative Tools 507
9.3.1 Editors 507
9.3.2 Linkers 508
9.3.3 Interpreters 508
9.3.4 Code Generators 509
9.3.5 Debuggers 510
9.3.6 Tools Used in Software Testing 511
9.3.7 Static Analyzers 512
9.3.8 Graphical User Interface Tools 514
9.3.9 Configuration Management Tools 516
9.3.10 Tracking tools 520
9.3.11 Reverse and reengineering tools 521
9.3.12 Process support tools 522
9.3.13 Management Tools 522
9.4 Tool Integration 523
9.5 Forces Influencing the Evolution of Tools 524
9.6 Concluding Remarks 525
Further Exercises 525
Hints and Sketchy Solutions 526
Bibliographic Notes 526

Chapter 10 Epilogue 529

10.1 The Future 529
10.2 Ethics and Social Responsibility 532
10.3 Software Engineering Code of Ethics 533
10.4 Concluding Remarks 534
Bibliographic notes 534

Appendix 536

Case Studies

Case Study A: Automating a Law Office 536
A.1 Economic and Financial Planning 538
A.2 Technical Planning and Manag:ment 538
A.3 Project Monitoring 540
A.4 Initial Delivery 540
A.5 Partial Recovery 541
Case Study B: Building a Family of Compliers 541
B.1 Initial Product Planning 541
B.2 Economic and Financial Planning 542
B.3 Technical Planning and Management 542
B.4 Early Development Phases 543
B.5 Project Monitoring 543
B.6 Project Reexamination, Revival, And Goal Setting 544
B.7 Assignment of Responsibility 545
B.8 Steady Progress and Release of the Product 546
B.9 Product Distribution 546
B.10 Remarks 546
Case Study C: Incremental Delivery 549
Case Study D. Applying Formal Methods in Industry 550
D.1 Education and Training 551
D.2 Requirements Specification 552
D.3 Requirements Validation and Verification Planning 553
D.4 Design, Implementation, and Verification 555
D.5 Overall Evaluation and Assessment 555
D.6 The Impact of the Project on Company's Strategy 557
Concluding Remarks 558
Bibliographic Notes 559

Bibliography 560

Index 589

Preface to the Second Edition

The first edition of this book was published in 1991. Since then, there has been a lot of progress in computing technology and also in software engineering. Certainly the proliferation of the Internet has had a profound influence on education, research, development, business, and commerce. We decided to produce this second edition in order to bring the book up to date with respect to the advances in software engineering in the last 10 years.

We were pleased to find that the basic premise of the book—the durability and importance of principles—has been borne out by the passage of time: Even though the technology has improved, principles of software engineering have remained the same. We have therefore been able to update every chapter without changing the original structure of the book. The following is still the structure:

Introduction: Chapters 1 –3;
The product: Chapters 4 –6;
Process and management: Chapters 7 –8;
Tools and environments: Chapter 9.

The product-related chapters follow the sequence consisting of design (4), specification (5), and verification (6). This is different from the approach taken by other books, which cover specification before design. The reason for our choice follows from the principles-based approach of the book. All of these activities—design, specification, and verification—are basic activities that must be learned and applied throughout the software life cycle. For example, design is something we do not only with software architecture, but also with software specifications. The modular design approach helps us structure software and also the specification documents. Other books present specification first and then design, allegedly because—according to the traditional software processes—first we specify a software and then we design it. By contrast, we believe that learning about the design activity and approaches first, creates the needed motivation for the study of specification and provides the skills and techniques for structuring those specifications.

While all areas of software engineering have evolved since the first edition of the book was written, the area of tools and environments has changed substantially. Chapter 9, therefore, is revised considerably. Our approach in this chapter also is to present primarily principles rather than specific tools. We have seen over the years that tools change as technology evolves, and the choice of what particular tools to study depends on the student's environment and focus. We therefore cover a framework for studying and evaluating software tools without a detailed look at any particular tools.

Besides many minor improvements and changes, we have made the following major additions: