



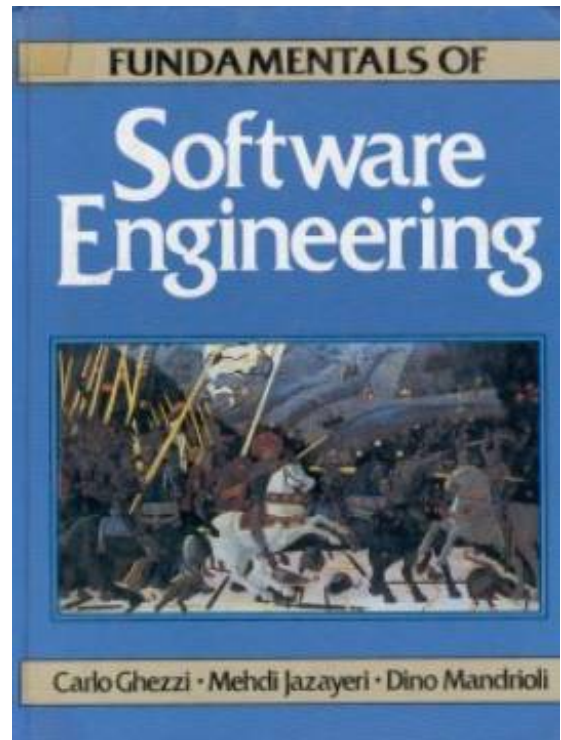
**FUNDAMENTALS OF
SOFTWARE
ENGINEERING**
CARLO GHEZZI
MEHDI JAZAYERI
DINO MANDRIOLI

SUMMARY

Here is the first book to emphasize fundamental principles rather than specific software engineering tools and techniques

Fundamentals of Software Engineering provides selective, in-depth coverage of the fundamentals of software engineering, stressing principles, methods, and rigorous formal and informal approaches. It stresses the importance of rigor in the practice of software engineering and emphasizes the important principles that can be applied independently of the life cycle model.

Numerous clear examples and exercises showing the application of principles to practical real-world problems are also included and make this an excellent self-study guide.



CONTENT HIGHLIGHTS

*uses small examples to illustrate principles and large case studies to show application and combination of principles in more realistic situations.

*emphasizes formality, design for change, and incrementality, using case studies to compare and contrast the different formalisms

*draws analogies to other engineering disciplines

*covers design, specification, verification and validation, the software process, management, and environments.

CONTENTS

PREFACE		xi
1	SOFTWARE ENGINEERING: A PREVIEW	1
1.1	The Role of Software Engineering in System Design	2
1.2	A 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	8
	1.5.1 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	13
	1.6.1 Management Science	13
	1.6.2 Systems Engineering	13

1.7	Concluding Remarks	14
	Bibliographic Notes	14
2	SOFTWARE: ITS NATURE AND QUALITIES	17
2.1	Classification of Software Qualities	18
2.1.1	External Versus Internal Qualities	18
2.1.2	Product and Process Qualities	19
2.2	Representative Qualities	19
2.2.1	Correctness, Reliability, and Robustness	19
2.2.2	Performance	22
2.2.3	User Friendliness	24
2.2.4	Verifiability	25
2.2.5	Maintainability	25
2.2.6	Reusability	28
2.2.7	Portability	30
2.2.8	Understandability	31
2.2.9	Interoperability	31
2.2.10	Productivity	32
2.2.11	Timeliness	33
2.2.12	Visibility	35
2.3	Quality Requirements in Different Application Areas	36
2.3.1	Information Systems	36
2.3.2	Real-time Systems	37
2.3.3	Distributed Systems	38
2.3.4	Embedded Systems	39
2.4	Measurement of Quality	40
2.5	Concluding Remarks	40
	Further Exercises	40
	Hints and Sketchy Solutions	41
	Bibliographic Notes	41
3	SOFTWARE ENGINEERING PRINCIPLES	43
3.1	Rigor and Formality	45
3.2	Separation of Concerns	47
3.3	Modularity	49
3.4	Abstraction	51
3.5	Anticipation of Change	53
3.6	Generality	54
3.7	Incrementality	56
3.8	Concluding Remarks	57
	Further Exercises	57
	Hints and Sketchy Solutions	58
	Bibliographic Notes	58
4	SOFTWARE DESIGN	61
4.1	Software Design Activity and Its Objectives	63
4.1.1	Design for Change	65
4.1.2	Program Families	70
4.2	Modularization Techniques	72
4.2.1	The Module Structure and Its Representation	73
4.2.2	Interface, Implementation, and Information Hiding,	81
4.2.3	Design Notations	89
4.2.4	Categories of Modules,	96
4.2.5	More Techniques for Design for Change	104
4.2.6	Stepwise Refinement	107
4.2.7	Top-down vs. Bottom-up Design	115
4.3	Object-oriented Design	115
4.4	Handling Anomalies	122
4.5	A Case Study in Design	125
4.6	Concurrent Software	128
4.6.1	Shared Data	129
4.6.2	Real-time Software	136
4.6.3	Distributed Software	138
4.7	Concluding Remarks	144

	Further Exercises	145
	Hints and Sketchy Solutions	147
	Bibliographic Notes	148
5	SOFTWARE SPECIFICATION	151
5.1	The Uses of Specifications	152
5.2	Specification Qualities	154
5.3	Classification of Specification Styles	157
5.4	Verification of Specifications	159
5.5	Operational Specifications	160
5.5.1	Data Flow Diagrams: Specifying Functions of Information Systems	161
5.5.2	Finite State Machines: Describing Control Flow	167
5.5.3	Petri Nets: Specifying Asynchronous Systems	174
5.6	Descriptive Specifications	199
5.6.1	Entity-relationship Diagrams	199
5.6.2	Logic Specifications	202
5.6.3	Algebraic Specifications	219
5.7	Building and Using Specifications in Practice	226
5.7.1	Requirements for Specification Notations	227
5.7.2	Building Modular Specifications	230
5.7.3	Specifications for the End User	241
5.8	Concluding Remarks	242
	Further Exercises	243
	Hints and Sketchy Solutions	248
	Bibliographic Notes	251
6	SOFTWARE VERIFICATION	255
6.1	Goals and Requirements of Verification	256
6.1.1	Everything Must Be Verified	257
6.1.2	The Results of Verification May Not Be Binary	257
6.1.3	Verification May Be Objective or Subjective	258
6.1.4	Even Implicit Qualities Must Be Verified	259
6.2	Approaches to Verification	260
6.3	Testing	260
6.3.1	Goals for Testing	262
6.3.2	Theoretical Foundations of Testing	264
6.3.3	Empirical Testing Principles	266
6.3.4	Testing in the Small	269
6.3.5	Testing in the Large	288
6.3.6	Separate Concerns in the Testing Activity	293
6.3.7	Testing Concurrent and Real-time Systems	295
6.4	Analysis	297
6.4.1	Informal Analysis Techniques	298
6.4.2	Correctness Proofs	301
6.5	Symbolic Execution	318
6.5.1	Basic Concepts of Symbolic Execution	320
6.5.2	Programs with Arrays	323
6.5.3	The Use of Symbolic Execution in Testing	325
6.5.4	Symbolic Execution of Concurrent Programs	327
6.6	Debugging	329
6.7	Verifying Other Software Properties	334
6.7.1	Verifying Performance	334
6.7.2	Verifying Reliability	335
6.7.3	Source-code Metrics	339
6.8	Concluding Remarks	344
	Further Exercises	345
	Hints and Sketchy Solutions	350
	Bibliographic Notes	353
7	THE SOFTWARE PRODUCTION PROCESS	357
7.1	Software Production Process Models	358
7.1.1	Waterfall Model	360
7.1.2	Evolutionary Model	374
7.1.3	Transformation Model	377

71.4	Spiral Model	380
71.5	An Assessment of Process Models	382
7.2	Case Studies	384
	72.1 Telephone Switching System	384
	7.2.2 Budget Control System	389
7.3	Organizing the Process	394
	7.3.1 Software Methodologies	394
	7.3.2 Configuration Management	403
	7.3.3 Software Standards	408
7.4	Concluding Remarks	409
	Further Exercises	409
	Hints and Sketchy Solutions	411
	Bibliographic Notes	411

8 MANAGEMENT OF SOFTWARE ENGINEERING 415

8.1	Management Functions	417
8.2	Project Planning	418
	8.2.1 Software Productivity	420
	8.2.2 People and Productivity	426
	8.2.3 Cost Estimation	427
8.3	Project Control	434
	8.3.1 Work Breakdown Structures	434
	8.3.2 Gantt Charts	435
	8.3.3 PERT Charts	437
	8.3.4 Dealing with Deviations from the Plan	440
8.4	Organization	441
	8.4.1 Centralized-control Team Organization	443
	8.4.2 Decentralized-control Team Organization	444
	8.4.3 Mixed-control Team Organization	445
	8.4.4 An Assessment of Team Organizations	446
8.5	Risk Management	446
	8.5.1 Typical Management Risks in Software Engineering	447
8.6	Concluding Remarks	449
	Further Exercises	451
	Hints and Sketchy Solutions	452
	Bibliographic Notes	452

9 SOFTWARE ENGINEERING TOOLS AND ENVIRONMENTS 455

9.1	Historical Evolution of Tools and Environments	456
9.2	Classification of Software Tools and Environments	458
9.3	Representative Tools	462
	9.3.1 Editors	463
	9.3.2 Linkers	464
	9.3.3 Interpreters	464
	9.3.4 Code Generators	465
	9.3.5 Debuggers	466
	9.3.6 Tools Used in Software Testing	467
	9.3.7 Static Analyzers	468
	9.3.8 Userinterface Management Tools	470
	9.3.9 Configuration Management Tools	474
	9.3.10 Management Tools	477
	9.3.11 Software Engineering Infrastructures	478
9.4	The Role of Programming Language in the Environment	479
	9.4.1 Procedural versus Nonprocedural Languages	480
	9.4.2 Features of Programming in the Small,	483
	9.4.3 Programming Languages and Modularity	484
	9.4.4 Object-oriented Programming Languages	487
	9.4.5 Programming Languages and the Handling of Anomalies	491
	9.4.6 Programming Languages and Concurrency	493
	9.4.7 Programming Languages and Verification	496
	9.4.8 Programming Languages and Software Design,	497
9.5	Some Sample Tools and Environments	498
	9.5.1 Teamwork	498
	9.5.2 The UNIR Environment	500
	9.5.3 Language-centered Environments: Smalltalk80 and KEE	504

	9.5.4 PCTE	505
9.6	An Ideal Scenario for the Future	506
9.7	Concluding Remarks	511
	Further Exercises	511
	Hints and Sketchy Solutions	512
	Bibliographic Notes	513
10	EPILOGUE	517
	10.1 The Future	518
	10.2 Ethics and Social Responsibility	520
	10.3 Concluding Remarks	521
	Bibliographic Notes	521
	CASE STUDIES	523
A.	Case Study 1: Automating a Law Office	524
	A.1 Economic and Financial Planning	525
	A.2 Technical Planning and Management	526
	A.3 Project Monitoring	527
	A.4 Initial Delivery	528
	A.5 Partial Recovery	528
B.	Case Study 2: Building a Family of Compilers	529
	B.1 Initial Product Planning	529
	B.2 Economic and Financial Planning	529
	B.3 Technical Planning and Management	530
	B.4 Early Development Phases	530
	B.5 Project Monitoring	531
	B. 6 Project Reexamination, Revival, and Goal Setting	531
	B.7 Assignment of Responsibility	533
	B.8 Steady Progress and Release of the Product	533
	B. 9 Product Distribution	534
	B.10 Remarks	534
C.	Case Study 3: Incremental Delivery	537
	Concluding Remarks	538
	REFERENCES	539
	INDEX	567

[TOP](#)

Contribute to awsaavedra/fundamentals-of-software-engineering development by creating an account on GitHub.Â Objective: This repository aims to give beginners a strong and comprehensive background in fundamental concepts and practices for software engineering. â”œâ”€â”€ language # Language, Object Oriented Programming, OOD, etc. â”œâ”€â”€ data-structures # Numerous useful data structures â”œâ”€â”€ algorithms # sorting, shortest path, etc. â”œâ”€â”€ design-patterns # General ways to design code for a reoccurring problem â”œâ”€â”€ version-control # Just learn git, plenty of free resources â”œâ”€â”€ terminal # Learn Bash (cd, ls, etc.