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
1.5.2 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 13
1.6.1 Management Science 13
1.6.2 Systems Engineering 13
# Software Specification

## 5

### 5.1 The Uses of Specifications

### 5.2 Specification Qualities

### 5.3 Classification of Specification Styles

### 5.4 Verification of Specifications

### 5.5 Operational Specifications

#### 5.5.1 Data Flow Diagrams: Specifying Functions of Information Systems

#### 5.5.2 Finite State Machines: Describing Control Flow

#### 5.5.3 Petri Nets: Specifying Asynchronous Systems

### 5.6 Descriptive Specifications

#### 5.6.1 Entity-relationship Diagrams

#### 5.6.2 Logic Specifications

#### 5.6.3 Algebraic Specifications

### 5.7 Building and Using Specifications in Practice

#### 5.7.1 Requirements for Specification Notations

#### 5.7.2 Building Modular Specifications

#### 5.7.3 Specifications for the End User

### 5.8 Concluding Remarks

Further Exercises

# Software Verification

## 6

### 6.1 Goals and Requirements of Verification

#### 6.1.1 Everything Must Be Verified

#### 6.1.2 The Results of Verification May Not Be Binary

#### 6.1.3 Verification May Be Objective or Subjective

#### 6.1.4 Even Implicit Qualities Must Be Verified

### 6.2 Approaches to Verification

### 6.3 Testing

#### 6.3.1 Goals for Testing

#### 6.3.2 Theoretical Foundations of Testing

#### 6.3.3 Empirical Testing Principles

#### 6.3.4 Testing in the Small

#### 6.3.5 Testing in the Large

#### 6.3.6 Separate Concerns in the Testing Activity

#### 6.3.7 Testing Concurrent and Real-time Systems

### 6.4 Analysis

#### 6.4.1 Information Analysis Techniques

#### 6.4.2 Correctness Proofs

### 6.5 Symbolic Execution

#### 6.5.1 Basic Concepts of Symbolic Execution

#### 6.5.2 Programs with Arrays

#### 6.5.3 The Use of Symbolic Execution in Testing

#### 6.5.4 Symbolic Execution of Concurrent Programs

### 6.6 Debugging

### 6.7 Verifying Other Software Properties

#### 6.7.1 Verifying Performance

#### 6.7.2 Verifying Reliability

#### 6.7.3 Source-code Metrics

### 6.8 Concluding Remarks

Further Exercises

# The Software Production Process

## 7

### 7.1 Software Production Process Models

#### 7.1.1 Waterfall Model

#### 7.1.2 Evolutionary Model

#### 7.1.3 Transformation Model
8 MANAGEMENT OF SOFTWARE ENGINEERING

8.1 Management Functions
8.2 Project Planning
8.2.1 Software Productivity
8.2.2 People and Productivity
8.2.3 Cost Estimation
8.3 Project Control
8.3.1 Work Breakdown Structures
8.3.2 Gantt Chants
8.3.3 PERT Chants
8.3.4 Dealing with Deviations from the Plan
8.4 Organization
8.4.1 Centralized-control Team Organization
8.4.2 Decentralized-control Team Organization
8.4.3 Mixed-control Team Organization
8.4.4 An Assessment of Team Organizations
8.5 Risk Management
8.5.1 Typical Management Risks in Software Engineering
8.6 Concluding Remarks

Further Exercises
Hints and Sketchy Solutions
Bibliographic Notes

9 SOFTWARE ENGINEERING TOOLS AND ENVIRONMENTS

9.1 Historical Evolution of Tools and Environments
9.2 Classification of Software Tools and Environments
9.3 Representative Tools
9.3.1 Editors
9.3.2 Linkers
9.3.3 Interpreters
9.3.4 Code Generators
9.3.5 Debuggers
9.3.6 Tools Used in Software Testing
9.3.7 Static Analyzers
9.3.8 Userinterface Management Tools
9.3.9 Configuration Management Tools
9.3.10 Management Tools
9.3.11 Software Engineering Infrastructures
9.4 The Role of Programming Language in the Environment
9.4.1 Procedural Versus Nonprocedural Languages
9.4.2 Features of Programming in the Small,
9.4.3 Programming Languages and Modularity
9.4.4 Object-oriented Programming Languages
9.4.5 Programming Languages and the Handling of Anomalies
9.4.6 Programming Languages and Concurrency
9.4.7 Programming Languages and Verification
9.4.8 Programming Languages and Software Design,
9.5 Some Sample Tools and Environments
9.5.1 Teamwork
9.5.2 The UNIR Environment
9.5.3 Language-centered Environments: Smalltalk80 and KEE