- Chapter 1 Software Engineering: An Overview p. 1
- 1.1 Introduction p. 1
- 1.2 A Case for a Software Engineering Approach to Building Imaging Systems p. 2
- 1.3 The Role of the Software Engineer p. 3
- 1.4 The Nature of Software for Imaging Systems p. 4
- 1.5 Case Study: A Visual Inspection System p. 5
- 1.6 Misconceptions about Software Engineering p. 7
- 1.7 Exercises p. 8
- Chapter 2 Imaging Software and Its Properties p. 9
- 2.1 Classification of Software Qualities p. 9
- 2.1.1 Reliability p. 9
- 2.1.2 Correctness p. 12
- 2.1.3 Performance p. 12
- 2.1.4 Usability p. 13
- 2.1.5 Interoperability p. 13
- 2.1.6 Maintainability p. 13
- 2.1.7 Portability p. 14
- 2.1.8 Verifiability p. 14
- 2.1.9 Summary of Software Properties and Associated Metrics p. 14
- 2.2 Basic Software Engineering Principles p. 15
- 2.2.1 Rigor and Formality p. 15
- 2.2.2 Separation of Concerns p. 15
- 2.2.3 Modularity p. 15
- 2.2.4 Anticipation of Change p. 18
- 2.2.5 Generality p. 20
- 2.2.6 Incrementality p. 20
- 2.2.7 Traceability p. 20
- 2.3 Exercises p. 22
- Chapter 3 Software Process and Life Cycle Models p. 23
- 3.1 Software Processes and Methodologies p. 23
- 3.2 Software Life Cycle Models p. 23
- 3.2.1 The Waterfall Model p. 24
- 3.2.1.1 Software Conception p. 25
- 3.2.1.2 Requirements Specification p. 25
- 3.2.1.3 Software Design p. 26
- 3.2.1.4 Software Development p. 26
- 3.2.1.5 Testing p. 27
- 3.2.1.6 Software Maintenance p. 27
- 3.2.1.7 Backtracking Transitions in the Waterfall Life Cycle p. 27
- 3.2.1.8 Waterfall Model Summary p. 28
- 3.2.2 V Model p. 28
- 3.2.3 The Spiral Model p. 28
- 3.2.4 Evolutionary Model p. 30
- 3.2.5 Incremental Model p. 31
- 3.2.6 Fountain Model p. 31
- 3.2.7 Lightweight Methodologies p. 32

- 3.2.8 Unified Process Model p. 34
- 3.2.9 Capability Maturity Model p. 34
- 3.2.9.1 CMM-1: Initial p. 34
- 3.2.9.2 CMM-2: Repeatable p. 34
- 3.2.9.3 CMM-3: Defined p. 35
- 3.2.9.4 CMM-4: Managed p. 35
- 3.2.9.5 CMM-5: Optimizing p. 35
- 3.2.9.6 CMM-I p. 35
- 3.2.10 Prototyping and Risk p. 35
- 3.3.4 ISO/IEC p. 39
- 3.3 Software Standards p. 36
- 3.3.1 DOD-STD-2167A p. 37
- 3.3.2 DOD-STD-498 p. 37
- 3.3.3 ISO 9000-3 p. 38
- 3.4 Exercises p. 40
- Chapter 4 Software Requirements p. 41
- 4.1 Requirements Engineering Process p. 41
- 4.2 Types of Requirements p. 42
- 4.3 Requirements Users p. 43
- 4.4 Formal Methods in Software Specification p. 44
- 4.4.1 Limitations of Formal Methods p. 45
- 4.4.2 Z p. 46
- 4.4.3 Finite State Machines p. 46
- 4.4.4 Statecharts p. 49
- 4.4.5 Petri Nets p. 51
- 4.5 Specification of Imaging Systems: A Survey of Current Practices p. 53
- 4.5.1 Multiresolution Block-Matching System Specification Using a Block Diagram and Flowchart p. 55
- 4.5.2 Collision Testing of Graphical Objects Using Pseudo-Code p. 56
- 4.5.3 Functional Representation of Machine Vision System Using a Structured Approach p. 56
- 4.5.4 Markov Random Fields Image Reconstruction Using Object-Oriented Design p. 57
- 4.6 Case Study p. 58
- 4.6.1 Structured Analysis and Design p. 59
- 4.6.2 Structured Analysis p. 59
- 4.7 Object-Oriented Analysis p. 61
- 4.8 Object-Oriented vs. Structured Analysis p. 62
- 4.8.1 Recommendations on Specification Approach for Imaging Systems p. 64
- 4.9 Organizing the Requirements Document p. 64
- 4.9.1 Writing Good Requirements p. 66
- 4.10 Requirements Validation and Review p. 67
- 4.11 Some Surprises about Current Software Specification Practices p. 68
- 4.11.1 Surprise 1 p. 68
- 4.11.2 Surprise 2 p. 68
- 4.11.3 Surprise 3 p. 68

- 5.1 The Design Activity p. 71
- 4.11.4 Surprise 4 p. 69
- 4.12 Exercises p. 69
- Chapter 5 Software System Design p. 71
- 5.2 Procedural-Oriented Design p. 72
- 5.2.1 Parnas Partitioning p. 72
- 5.2.2 Structured Design p. 74
- 5.2.2.1 Transitioning from Structured Analysis to Structured Design p. 74
- 5.2.2.2 Data Dictionaries p. 76
- 5.2.2.3 Problems with SASD in Imaging Applications p. 77
- 5.2.2.4 Real-Time Extensions of SASD p. 78
- 5.2.3 Design in Procedural Form Using Finite State Machines p. 78
- 5.3 Object-Oriented Design p. 80
- 5.3.1 Benefits of Object Orientation p. 81
- 5.3.1.1 Open-Closed Principle p. 81
- 5.3.1.2 Once and Only Once p. 82
- 5.3.1.3 Dependency Inversion Principle p. 82
- 5.3.1.4 Liskov Substitution Principle p. 82
- 5.3.2 Design Patterns p. 83
- 5.3.3 Object-Oriented Design Using Unified Modeling Language p. 84
- 5.3.4 Modeling Time Explicitly p. 84
- 5.4.2 Non-von Neumann Architectures p. 95
- 5.3.5 Visual Inspection System Case Study p. 88
- 5.4 Hardware Considerations in Imaging System Design p. 93
- 5.4.1 Processors p. 94
- 5.4.2.1 Single Instruction Single Data p. 95
- 5.4.2.2 Single Instruction Multiple Data p. 95
- 5.4.2.3 Multiple Instruction Single Data p. 96
- 5.4.2.4 Multiple Instruction Multiple Data p. 96
- 5.4.3 Interrupt Handling p. 96
- 5.4.4 Memory p. 97
- 5.4.5 Input and Output p. 97
- 5.5 Fault-Tolerant Design p. 99
- 5.5.1 Spatial Fault Tolerance p. 99
- 5.5.2 Using a Kalman Filter in the Case Study System p. 99
- 5.5.3 Checkpoints p. 102
- 5.5.4 Recovery Blocks p. 102
- 5.5.5 Software Black Boxes p. 104
- 5.5.6 N-Version Programming p. 105
- 5.5.6.1 Built-In Test Software p. 105
- 5.5.6.2 CPU Testing p. 106
- 5.5.6.3 Memory Testing p. 106
- 5.5.6.4 Other Devices p. 107
- 5.6 Exercises p. 107
- Chapter 6 The Software Production Process p. 109
- 6.1 Programming Languages p. 109

- 6.1.1 Parameter Passing Techniques p. 110
- 6.1.5 Dynamic Memory Allocation p. 111
- 6.1.2 Call-by-Value and Call-by-Reference p. 110
- 6.1.3 Global Variables p. 110
- 6.1.4 Recursion p. 111
- 6.1.6 Typing p. 112
- 6.1.7 Exception Handling p. 112
- 6.1.8 Modularity p. 113
- 6.1.9 Brief Survey of Languages p. 114
- 6.1.9.1 Ada 95 p. 114
- 6.1.9.2 Assembly Language p. 115
- 6.1.9.3 C p. 115
- 6.1.9.4 C++ p. 116
- 6.1.9.5 Fortran p. 116
- 6.1.9.6 Java p. 117
- 6.2 Writing and Testing Code p. 118
- 6.2.1 Example: The Unix/Linux C Compiler p. 119
- 6.2.2 Handling Compiler Errors p. 120
- 6.2.3 Some Debugging Tips: Unit-Level Testing p. 120
- 6.2.4 Extended Syntax and Semantic Checking p. 120
- 6.2.5 Symbolic Debugging p. 121
- 6.2.6 Test-First Coding p. 122
- 6.2.7 Know the Compiler p. 122
- 6.3 Coding Standards p. 123
- 6.4 Reviews and Audits p. 124
- 6.5 Documentation p. 126
- 6.6 Exercises p. 127
- Chapter 7 Software Measurement and Testing p. 129
- 7.1 The Role of Metrics p. 129
- 7.1.1 Lines of Code p. 129
- 7.1.2 McCabe's Metric p. 130
- 7.1.2.1 Measuring Software Complexity p. 130
- 7.1.2.2 Determining the Limit on Number of Test Cases p. 132
- 7.1.3 Halstead's Metrics p. 132
- 7.1.4 Function Points p. 133
- 7.1.5 Feature Points p. 137
- 7.1.6 Metrics for Object-Oriented Software p. 137
- 7.1.7 Objections to Metrics p. 138
- 7.2 Faults, Failures, and Bugs p. 138
- 7.3 The Role of Testing p. 139
- 7.4 Testing Techniques p. 139
- 7.4.1 Unit-Level Testing p. 139
- 7.4.1.1 Black Box Testing p. 139
- 7.4.1.2 White Box Testing p. 141
- 7.4.2 Testing Object-Oriented Software p. 142
- 7.4.3 System-Level Testing p. 142

- 7.4.3.1 Cleanroom Testing p. 143
- 7.4.3.2 Stress Testing p. 143
- 7.5 Design of Testing Plans p. 144
- 7.6 Exercises p. 144
- Chapter 8 Hardware-Software Integration and Maintenance p. 145
- 8.1 Goals of System Integration p. 145
- 8.2 System Unification p. 145
- 8.3 System Verification p. 146
- 8.4 System Integration Tools p. 146
- 8.4.1 Multimeter p. 147
- 8.4.2 Oscilloscope p. 147
- 8.4.3 Logic Analyzer p. 147
- 8.4.3.1 Timing Instructions p. 148
- 8.4.3.2 Timing Code p. 148
- 8.4.6 Hardware Prototypes p. 149
- 8.5 Software Integration p. 150
- 8.5.1 A Simple Integration Strategy p. 150
- 8.5.2 Patching p. 150
- 8.5.3 The Probe Effect p. 152
- 8.6 Postintegration Software Optimization p. 153
- 8.6.1 CPU Utilization Estimation p. 153
- 8.6.2 Execution Time Estimation p. 154
- 8.6.3 Scaled Numbers p. 154
- 8.6.4 Binary Angular Measure p. 155
- 8.6.5 Look-Up Tables p. 155
- 8.6.6 Imprecise Computation p. 157
- 8.6.7 Optimizing Memory Usage p. 157
- 8.7 A Software Reengineering Process Model p. 157
- 8.8 A Maintenance Process Model p. 158
- 8.9 Software Reuse p. 159
- 8.9.2.2 In Object-Oriented Languages p. 162
- 8.4.4 In-Circuit Emulator p. 149
- 8.4.5 Software Simulators p. 149
- 8.9.1 When Not to Reuse p. 160
- 8.9.2 Achieving Reuse p. 160
- 8.9.2.1 In Procedural Languages p. 161
- 8.9.2.3 Pareto's Principle p. 162
- 8.10 The Second System Effect p. 162
- 8.11 Code and Program Maintenance p. 163
- 8.12 Exercises p. 163
- Chapter 9 Management of Software Projects p. 165
- 9.1 Why Software Project Management? p. 165
- 9.2 Software Project Management Themes p. 166
- 9.3 General Project Management Basics p. 166
- 9.3.1 What Does the Project Manager Control? p. 166
- 9.4 Software Project Management p. 167

- 9.5 Managing and Mitigating Risks p. 168
- 9.6 Personnel Management p. 169
- 9.6.1 The n-Body Problem p. 170
- 9.6.2 Some Approaches to Leading Teams p. 171
- 9.6.2.1 Theory X p. 171
- 9.6.2.2 Theory Y p. 171
- 9.6.2.3 Theory Z p. 171
- 9.6.2.4 Theory W p. 172
- 9.6.4 Dealing with Difficult People p. 174
- 9.6.3 Principle-Centered Leadership p. 173
- 9.6.3.1 Management by Sight p. 173
- 9.6.3.2 Management by Objectives p. 173
- 9.7 Assessment of Project Personnel p. 174
- 9.7.1 Skills Testing p. 174
- 9.7.2 Recommended Practices p. 175
- 9.8 Tracking and Reporting Progress p. 177
- 9.8.1 Gantt Chart p. 177
- 9.8.2 Critical Path Method p. 178
- 9.8.3 Program Evaluation and Review Technique p. 179
- 9.9 Cost Estimation Using COCOMO p. 180
- 9.9.1 Basic COCOMO p. 180
- 9.9.2 Intermediate and Detailed COCOMO p. 182
- 9.9.3 COCOMO II p. 183
- 9.10 Exercises p. 183
- Glossary p. 185
- References p. 203
- Index p. 209