

Table of contents provided by Syndetics

- **Preface** (p. v)
- **Part 1 Overview** (p. 1)
- **Chapter 1 Introduction** (p. 3)
- **1.1 FAQs about software engineering** (p. 5)
- **1.2 Professional and ethical responsibility** (p. 14)
- **Key points** (p. 17)
- **Further reading** (p. 18)
- **Exercises** (p. 18)
- **Chapter 2 Computer-based system engineering** (p. 20)
- **2.1 Emergent system properties** (p. 22)
- **2.2 Systems and their environment** (p. 24)
- **2.3 System modelling** (p. 26)
- **2.4 The system engineering process** (p. 29)
- **2.5 System procurement** (p. 37)
- **Key points** (p. 39)
- **Further reading** (p. 40)
- **Exercises** (p. 40)
- **Chapter 3 Software processes** (p. 42)
- **3.1 Software process models** (p. 44)
- **3.2 Process iteration** (p. 51)
- **3.3 Software specification** (p. 55)
- **3.4 Software design and implementation** (p. 56)
- **3.5 Software validation** (p. 60)
- **3.6 Software evolution** (p. 63)
- **3.7 Automated process support** (p. 63)
- **Key points** (p. 68)
- **Further reading** (p. 68)
- **Exercises** (p. 69)
- **Chapter 4 Project management** (p. 71)
- **4.1 Management activities** (p. 73)
- **4.2 Project planning** (p. 75)
- **4.3 Project scheduling** (p. 78)
- **4.4 Risk management** (p. 84)
- **Key points** (p. 90)
- **Further reading** (p. 91)
- **Exercises** (p. 92)
- **Part 2 Requirements** (p. 95)
- **Chapter 5 Software requirements** (p. 97)
- **5.4 The software requirements document** (p. 115)
- **5.1 Functional and non-functional requirements** (p. 100)
- **5.2 User requirements** (p. 106)
- **5.3 System requirements** (p. 109)
- **Key points** (p. 119)
- **Further reading** (p. 119)

- Exercises (p. 120)
- **Chapter 6 Requirements engineering processes** (p. 121)
- **6.1 Feasibility studies** (p. 123)
- **6.2 Requirements elicitation and analysis** (p. 124)
- **6.3 Requirements validation** (p. 137)
- **6.4 Requirements management** (p. 139)
- **Key points** (p. 145)
- **Further reading** (p. 145)
- Exercises (p. 146)
- **Chapter 7 System models** (p. 148)
- **7.1 Context models** (p. 150)
- **7.2 Behavioural models** (p. 153)
- **7.3 Data models** (p. 158)
- **7.4 Object models** (p. 160)
- **7.5 CASE workbenches** (p. 166)
- **Key points** (p. 168)
- **Further reading** (p. 169)
- Exercises (p. 169)
- **Chapter 8 Software prototyping** (p. 171)
- **8.1 Prototyping in the software process** (p. 174)
- **8.2 Rapid prototyping techniques** (p. 180)
- **8.3 User interface prototyping** (p. 188)
- **Key points** (p. 189)
- **Further reading** (p. 190)
- Exercises (p. 190)
- **Chapter 9 Formal specification** (p. 192)
- **9.1 Formal specification in the software process** (p. 194)
- **9.2 Interface specification** (p. 197)
- **9.3 Behavioural specification** (p. 204)
- **Key points** (p. 209)
- **Further reading** (p. 210)
- Exercises (p. 210)
- **Part 3 Design** (p. 213)
- **Chapter 10 Architectural design** (p. 215)
- **10.1 System structuring** (p. 219)
- **10.2 Control models** (p. 224)
- **10.3 Modular decomposition** (p. 229)
- **10.4 Domain-specific architectures** (p. 233)
- **Key points** (p. 236)
- **Further reading** (p. 237)
- Exercises (p. 237)
- **Chapter 11 Distributed systems architectures** (p. 239)
- **11.1 Multiprocessor architectures** (p. 243)
- **11.2 Client-server architectures** (p. 244)
- **11.3 Distributed object architectures** (p. 249)
- **11.4 CORBA** (p. 252)

- **Key points** (p. 257)
- **Further reading** (p. 258)
- **Exercises** (p. 258)
- **12.3 Design evolution** (p. 280)
- **Chapter 12 Object-oriented design** (p. 260)
- **12.1 Objects and object classes** (p. 262)
- **12.2 An object-oriented design process** (p. 267)
- **Key points** (p. 282)
- **Further reading** (p. 282)
- **Exercises** (p. 283)
- **Chapter 13 Real-time software design** (p. 285)
- **13.1 System design** (p. 287)
- **13.2 Real-time executives** (p. 291)
- **13.3 Monitoring and control systems** (p. 295)
- **13.4 Data acquisition systems** (p. 300)
- **Key points** (p. 303)
- **Further reading** (p. 303)
- **Exercises** (p. 304)
- **Chapter 14 Design with reuse** (p. 306)
- **14.1 Component-based development** (p. 310)
- **14.2 Application families** (p. 318)
- **14.3 Design patterns** (p. 322)
- **Key points** (p. 325)
- **Further reading** (p. 325)
- **Exercises** (p. 326)
- **Chapter 15 User interface design** (p. 327)
- **15.1 User interface design principles** (p. 330)
- **15.2 User interaction** (p. 332)
- **15.3 Information presentation** (p. 334)
- **15.4 User support** (p. 340)
- **15.5 Interface evaluation** (p. 345)
- **Key points** (p. 347)
- **Further reading** (p. 348)
- **Exercises** (p. 348)
- **Part 4 Critical Systems** (p. 351)
- **Chapter 16 Dependability** (p. 353)
- **16.1 Critical systems** (p. 356)
- **16.2 Availability and reliability** (p. 359)
- **16.3 Safety** (p. 364)
- **16.4 Security** (p. 367)
- **Key points** (p. 369)
- **Further reading** (p. 369)
- **Exercises** (p. 370)
- **Chapter 17 Critical systems specification** (p. 371)
- **17.1 Software reliability specification** (p. 373)
- **17.2 Safety specification** (p. 379)

- **17.3 Security specification** (p. 387)
- **Key points** (p. 389)
- **Further reading** (p. 389)
- **Exercises** (p. 390)
- **Chapter 18 Critical systems development** (p. 392)
- **18.1 Fault minimisation** (p. 393)
- **18.2 Fault tolerance** (p. 400)
- **18.3 Fault-tolerant architectures** (p. 410)
- **18.4 Safe system design** (p. 413)
- **Key points** (p. 414)
- **Further reading** (p. 415)
- **Exercises** (p. 415)
- **19.2 Software inspections** (p. 425)
- **Part 5 Verification and Validation** (p. 417)
- **Chapter 19 Verification and validation** (p. 419)
- **19.1 Verification and validation planning** (p. 423)
- **19.3 Automated static analysis** (p. 431)
- **19.4 Cleanroom software development** (p. 434)
- **Key points** (p. 437)
- **Further reading** (p. 438)
- **Exercises** (p. 438)
- **Chapter 20 Software testing** (p. 440)
- **20.1 Defect testing** (p. 442)
- **20.2 Integration testing** (p. 452)
- **20.3 Object-oriented testing** (p. 458)
- **20.4 Testing workbenches** (p. 462)
- **Key points** (p. 464)
- **Further reading** (p. 465)
- **Exercises** (p. 466)
- **Chapter 21 Critical systems validation** (p. 467)
- **21.1 Formal methods and critical systems** (p. 469)
- **21.2 Reliability validation** (p. 470)
- **21.3 Safety assurance** (p. 476)
- **21.4 Security assessment** (p. 483)
- **Key points** (p. 484)
- **Further reading** (p. 484)
- **Exercises** (p. 485)
- **Part 6 Management** (p. 487)
- **Chapter 22 Managing people** (p. 489)
- **22.1 Limits to thinking** (p. 490)
- **22.2 Group working** (p. 497)
- **22.3 Choosing and keeping people** (p. 503)
- **22.4 The People Capability Maturity Model** (p. 506)
- **Key points** (p. 508)
- **Further reading** (p. 509)
- **Exercises** (p. 509)

- **Chapter 23 Software cost estimation** (p. 511)
- **23.1 Productivity** (p. 513)
- **23.2 Estimation techniques** (p. 518)
- **23.3 Algorithmic cost modelling** (p. 520)
- **23.4 Project duration and staffing** (p. 531)
- **Key points** (p. 533)
- **Further reading** (p. 533)
- **Exercises** (p. 534)
- **Chapter 24 Quality management** (p. 535)
- **24.1 Quality assurance and standards** (p. 539)
- **24.2 Quality planning** (p. 544)
- **24.3 Quality control** (p. 546)
- **24.4 Software measurement and metrics** (p. 547)
- **Key points** (p. 555)
- **Further reading** (p. 555)
- **Exercises** (p. 556)
- **Chapter 25 Process improvement** (p. 557)
- **25.1 Process and product quality** (p. 560)
- **25.2 Process analysis and modelling** (p. 562)
- **25.3 Process measurement** (p. 566)
- **Further reading** (p. 576)
- **25.4 The SEI Process Capability Maturity Model** (p. 568)
- **25.5 Process classification** (p. 573)
- **Key points** (p. 576)
- **Exercises** (p. 577)
- **Part 7 Evolution** (p. 579)
- **Chapter 26 Legacy systems** (p. 581)
- **26.1 Legacy system structures** (p. 583)
- **26.2 Legacy system design** (p. 587)
- **26.3 Legacy system assessment** (p. 592)
- **Key points** (p. 598)
- **Further reading** (p. 599)
- **Exercises** (p. 599)
- **Chapter 27 Software change** (p. 601)
- **27.1 Program evolution dynamics** (p. 603)
- **27.2 Software maintenance** (p. 605)
- **27.3 Architectural evolution** (p. 614)
- **Key points** (p. 620)
- **Further reading** (p. 620)
- **Exercises** (p. 621)
- **Chapter 28 Software re-engineering** (p. 622)
- **28.1 Source code translation** (p. 626)
- **28.2 Reverse engineering** (p. 628)
- **28.3 Program structure improvement** (p. 629)
- **28.4 Program modularisation** (p. 632)
- **28.5 Data re-engineering** (p. 634)

- **Key points** (p. 638)
- **Further reading** (p. 639)
- **Exercises** (p. 639)
- **Chapter 29 Configuration management** (p. 641)
- **29.1 Configuration management planning** (p. 644)
- **29.2 Change management** (p. 647)
- **29.3 Version and release management** (p. 650)
- **29.4 System building** (p. 655)
- **29.5 CASE tools for configuration management** (p. 656)
- **Key points** (p. 660)
- **Further reading** (p. 661)
- **Exercises** (p. 661)
- **References** (p. 663)
- **Index** (p. 679)