

<<软件工程>>

图书基本信息

书名：<<软件工程>>

13位ISBN编号：9787111348252

10位ISBN编号：7111348257

出版时间：2011-7

出版时间：机械工业出版社

作者：（英）Ian Sommerville

页数：773

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<软件工程>>

内容概要

本书是系统介绍软件工程理论的经典教材，自1982年初版以来，随着软件工程学科的发展不断更新，影响了一代又一代软件工程人才，对学科本身也产生了积极影响。全书共四个部分，完整讨论了软件工程各个阶段的内容，是软件工程和系统工程专业本科生和研究生的优秀教材，也是软件工程师必备的参考书籍。

本书特点

- 涵盖了对所有开发过程都很基础的重要主题，包括软件工程理论与实践的最新进展。
- 将第8版中的八篇内容重构为四个部分，使教师讲授软件工程课程更加容易。
- 每一章都有30%~40%的更新，增加了敏捷软件开发和嵌入式系统等新篇章，补充了模型驱动工程、开源开发、测试驱动开发、可依赖系统体系结构、静态分析和模型检查、cots复用、服务作为软件以及敏捷规划等新内容。
- 着重讨论了开发可靠的分布式系统的相关主题以及敏捷方法和软件复用。
- 反映敏捷方法先进性的同时，不忘强调传统的计划驱动软件工程的作用，阐述了两相结合构建优秀软件系统的重要性。
- 以一个新的病人记录系统案例研究贯穿始终，系统、完整地讲解软件工程的各个方面。
- 设计为“印刷/web”相结合的方式，核心信息采用印刷版本，教辅材料及先前版本中的一些章节放在web上，为读者提供丰富翔实的信息。

<<软件工程>>

作者简介

作者：（英国）萨默维尔（Ian Sommerville）萨默维尔（Lan Sommerville），英国著名软件工程专家，曾任教于兰卡斯特大学，现为圣安德鲁斯大学软件工程教授、他是IEEE CS组织编撰“软件工程知识体系”（SWEBOK）的专家委员会成员之一。

他在软件工程的教学和科研方面有20多年的经验，其研究领域包括计算机系统工程、需求工程、系统可靠性以及软件进化。

<<软件工程>>

书籍目录

- preface v
- part 1 introduction to software engineering 1
- chapter 1 introduction 3
 - 1.1 professional software development 5
 - 1.2 software engineering ethics 14
 - 1.3 case studies 17
- chapter 2 software processes 27
 - 2.1 software process models 29
 - 2.2 process activities 36
 - 2.3 coping with change 43
 - 2.4 the rational unified process 50
- chapter 3 agile software development 56
 - 3.1 agile methods 58
 - 3.2 plan-driven and agile development 62
 - 3.3 extreme programming 64
 - 3.4 agile project management 72
 - 3.5 scaling agile methods 74
- chapter 4 requirements engineering 82
 - 4.1 functional and non-functional requirements 84
 - 4.2 the software requirements document 91
 - 4.3 requirements specification 94
 - 4.4 requirements engineering processes 99
 - 4.5 requirements elicitation and analysis 100
 - 4.6 requirements validation 110
 - 4.7 requirements management 111
- chapter 5 system modeling 118
 - 5.1 context models 121
 - 5.2 interaction models 124
 - 5.3 structural models 129
 - 5.4 behavioral models 133
 - 5.5 model-driven engineering 138
- chapter 6 architectural design 147
 - 6.1 architectural design decisions 151
 - 6.2 architectural views 153
 - 6.3 architectural patterns 155
 - 6.4 application architectures 164
- chapter 7 design and implementation 176
 - 7.1 object-oriented design using the uml 178
 - 7.2 design patterns 189
 - 7.3 implementation issues 193
 - 7.4 open source development 198
- chapter 8 software testing 205
 - 8.1 development testing 210
 - 8.2 test-driven development 221
 - 8.3 release testing 224

<<软件工程>>

- 8.4 user testing 228
- chapter 9 software evolution 234
 - 9.1 evolution processes 237
 - 9.2 program evolution dynamics 240
 - 9.3 software maintenance 242
 - 9.4 legacy system management 252
 - part 2 dependability and security 261
- chapter 10 sociotechnical systems 263
 - 10.1 complex systems 266
 - 10.2 systems engineering 273
 - 10.3 system procurement 275
 - 10.4 system development 278
 - 10.5 system operation 281
- chapter 11 dependability and security 289
 - 11.1 dependability properties 291
 - 11.2 availability and reliability 295
 - 11.3 safety 299
 - 11.4 security 302
- chapter 12 dependability and security specification 309
 - 12.1 risk-driven requirements specification 311
 - 12.2 safety specification 313
 - 12.3 reliability specification 320
 - 12.4 security specification 329
 - 12.5 formal specification 333
- chapter 13 dependability engineering 341
 - 13.1 redundancy and diversity 343
 - 13.2 dependable processes 345
 - 13.3 dependable system architectures 348
 - 13.4 dependable programming 355
- chapter 14 security engineering 366
 - 14.1 security risk management 369
 - 14.2 design for security 375
 - 14.3 system survivability 386
- chapter 15 dependability and security assurance 393
 - 15.1 static analysis 395
 - 15.2 reliability testing 401
 - 15.3 security testing 404
 - 15.4 process assurance 406
 - 15.5 safety and dependability cases 410
 - part 3 advanced software engineering 423
- chapter 16 software reuse 425
 - 16.1 the reuse landscape 428
 - 16.2 application frameworks 431
 - 16.3 software product lines 434
 - 16.4 cots product reuse 440
- chapter 17 component-based software engineering 452
 - 17.1 components and component models 455

<<软件工程>>

- 17.2 cbse processes 461
- 17.3 component composition 468
- chapter 18 distributed software engineering 479
 - 18.1 distributed systems issues 481
 - 18.2 client – server computing 488
 - 18.3 architectural patterns for distributed systems 490
 - 18.4 software as a service 501
- chapter 19 service-oriented architecture 508
 - 19.1 services as reusable components 514
 - 19.2 service engineering 518
 - 19.3 software development with services 527
- chapter 20 embedded software 537
 - 20.1 embedded systems design 540
 - 20.2 architectural patterns 547
 - 20.3 timing analysis 554
 - 20.4 real-time operating systems 558
- chapter 21 aspect-oriented software engineering 565
 - 21.1 the separation of concerns 567
 - 21.2 aspects, join points and pointcuts 571
 - 21.3 software engineering with aspects 576
- part 4 software management 591
- chapter 22 project management 593
 - 22.1 risk management 595
 - 22.2 managing people 602
 - 22.3 teamwork 607
- chapter 23 project planning 618
 - 23.1 software pricing 621
 - 23.2 plan-driven development 623
 - 23.3 project scheduling 626
 - 23.4 agile planning 631
 - 23.5 estimation techniques 633
- chapter 24 quality management 651
 - 24.1 software quality 655
 - 24.2 software standards 657
 - 24.3 reviews and inspections 663
 - 24.4 software measurement and metrics 668
- chapter 25 configuration management 681
 - 25.1 change management 685
 - 25.2 version management 690
 - 25.3 system building 693
 - 25.4 release management 699
- chapter 26 process improvement 705
 - 26.1 the process improvement process 708
 - 26.2 process measurement 711
 - 26.3 process analysis 715
 - 26.4 process change 718
 - 26.5 the cmmi process improvement framework 721

<<软件工程>>

glossary 733

subject index 749

author index 767

章节摘录

版权页：插图：The development of the World Wide Web has had a profound effect on all of our lives. Initially, the Web was primarily a universally accessible information store and it had little effect on software systems. These systems ran on local computers and were only accessible from within an organization. Around 2000, the Web started to evolve and more and more functionality was added to browsers. This meant that web-based systems could be developed where, instead of a special-purpose user interface, these systems could be accessed using a web browser. This led to the development of a vast range of new system products that delivered innovative services, accessed over the Web. These are often funded by adverts that are displayed on the user's screen and do not involve direct payment from users. As well as these system products, the development of web browsers that could run small programs and do some local processing led to an evolution in business and organizational software. Instead of writing software and deploying it on users' PCs, the software was deployed on a web server. This made it much cheaper to change and upgrade the software, as there was no need to install the software on every PC. It also reduced costs, as user interface development is particularly expensive. Consequently, wherever it has been possible to do so, many businesses have moved to web-based interaction with company software systems. The next stage in the development of web-based systems was the notion of web services. Web services are software components that deliver specific, useful functionality and which are accessed over the Web. Applications are constructed by integrating these web services, which may be provided by different companies. In principle, this linking can be dynamic so that an application may use different web services each time that it is executed. I cover this approach to software development in Chapter 19. In the last few years, the notion of 'software as a service' has been developed. It has been proposed that software will not normally run on local computers but will run on 'computing clouds' that are accessed over the Internet. If you use a service such as web-based mail, you are using a cloud-based system. A computing cloud is a huge number of linked computer systems that is shared by many users. Users do not buy software but pay according to how much the software is used or are given free access in return for watching adverts that are displayed on their screen. The advent of the web, therefore, has led to a significant change in the way that business software is organized. Before the web, business applications were mostly monolithic, single programs running on single computers or computer clusters. Communications were local, within an organization. Now, software is highly distributed, sometimes across the world. Business applications are not programmed from scratch but involve extensive reuse of components and programs.

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>