Software Engineering presents a broad perspective on software systems engineering, concentrating on widely-used techniques for developing large-scale software systems. In seven parts, this best-selling book covers a wide spectrum of software processes from initial requirements elicitation through design and development to system evolution. It supports students taking undergraduate and graduate courses in software engineering and software engineers in industry who need to update their knowledge on new techniques such as requirements engineering, distributed systems architectures and system dependability.

Extensive market research has ensured that this new edition is useful and relevant for both students and practising software engineers. The sixth edition has been restructured and updated, important new topics have been added and obsolete material has been cut. The end result is an even more focused book that is about 10% shorter than the previous edition.

Changes from the fifth edition

- There are new chapters covering software processes, distributed systems architectures, dependability and legacy systems.
- Program examples are now in Java and graphical system models are described in the standard UML.
- All chapters have been updated and several have been extensively rewritten. Reuse now focuses on component-based development and patterns; object-oriented design has a process focus and uses the UML; the chapters on requirements have been split to cover the requirements themselves and requirements engineering process; cost estimation has been updated to include the COCOMO 2 model.
- The chapters on critical systems has been restructured so that reliability, safety, availability and security are integrated in chapters on critical systems specification, development and validation.
- The section on formal specification has been cut to a single chapter and material on CASE has been integrated with the chapters covering the processes supported. Functional design has been incorporated in the new chapter on legacy systems.

The book's web site (www.software-engin.com) includes links to material to support the use of the book in teaching and personal study. It includes an instructor's manual, overhead transparencies, source code of the program examples and additional material on CASE and formal specification.

Ian Sommerville is Professor of Software Engineering at Lancaster University, England. He has more than 20 years of experience in software engineering education and research. His current areas of interest include computer-based systems engineering, requirements engineering, system dependability and software evolution.

Synopsis

Software Engineering presents a broad perspective on software systems engineering, concentrating on...
widely used techniques for developing large-scale systems. The objectives of this seventh edition are to include new material on iterative software development, component-based software engineering and system architectures, to emphasize that system dependability is not an add-on but should be considered at all stages of the software process, and not to increase the size of the book significantly. To this end the book has been restructured into 6 parts, removing the separate section on evolution as the distinction between development and evolution can be seen as artificial. New chapters have been added on:

Socio-technical Systems discussing the context of software in a broader system composed of other hardware and software, people, organisations, policies, procedures and laws.

Application System Architectures to teach students the general structure of application systems such as transaction systems, information systems and embedded control systems. The chapter covers 6 common system architectures with an architectural overview and discussion of the characteristics of these types of system.

Iterative Software Development looking at prototyping and adding new material on agile methods and extreme programming.

Component-based Software Engineering introducing the notion of a component, component composition and component frameworks and covering design with reuse.

Software Evolution revising the presentation of the 6th edition to cover re-engineering and software change in a single chapter.

The book supports students taking undergraduate or graduate courses in software engineering, and software engineers in industry needing to update their knowledge.

Description
Includes bibliographical references (p. [663]-677) and index.

Table of Contents
(NOTE: Each chapter concludes with Key points, Further reading, and Exercises.)

I. OVERVIEW. 1. Introduction.

FAQs About software engineering.

Professional and ethical responsibility. 2. Computer-based system engineering.

Emergent system properties.

Systems and their environment.

System modelling.

The system engineering process.

System procurement. 3. Software Processes.

Software process models.

Process iteration.

Software Specification.

Software design and implementation.

Software validation.
Software evolution.

Automated process support. 4. Project management.

Management activities.

Project planning.

Project scheduling.

Risk management.

II. REQUIREMENTS. 5. Software requirements.

Functional and non-functional requirements.

User requirements.

System requirements.

The software requirements document. 6. Requirements engineering processes.

Feasibility studies.

Requirements elicitation and analysis.

Requirements validation.

Requirements management. 7. System Models.

Context models.

Behavioural models.

Data models.

Object models.

CASE workbenches. 8. Software prototyping.

Prototyping in the software process.

Rapid prototyping techniques.


Formal specification in the software process.

Interface specification.

Behavioural specification.

III. DESIGN. 10. Architectural design.

System structuring.

Control models.

Modular decomposition.

Domain-specific architectures. 11. Distributed systems design.
Multiprocessor architectures.

Client-server architectures.

Distributed object architectures.

CORBA. **12. Object-oriented design.**

Objects and object classes.

An object-oriented design process.

Design evolution. **13. Real-time software design.**

System design.

Real-time executives.

Monitoring and control systems.

Data acquisition systems. **14. Design with Reuse.**

Component-based development. **Application families.**

Design patterns. **15. User interface design.**

User interface design principles.

User interaction.

Information presentation.

User support.

Interface evaluation.

**IV. CRITICAL SYSTEMS. 16. Dependability.**

Critical systems.

Availability and reliability.

Safety.

Security. **17. Critical systems specification.**

Software reliability specification.

Safety specification.

Security specification. **18. Critical systems development.**

Fault minimisation.

Fault tolerance.

Fault-tolerant architectures.

Safe system design.

**V. VERIFICATION AND VALIDATION. 19. Verification and validation.**
Verification and validation planning.

Software inspections.

Automated static analysis.

Cleanroom software development. **20. Software testing.**

Defect testing.

Integration testing.

Object-oriented testing. **21. Critical systems validation.**

Formal methods and critical systems.

Reliability validation.

Safety assurance.

**Security assessment.**

**VI. MANAGEMENT. 22. Managing people.**

Limits to thinking.

Group working.

Choosing and keeping people.

The People Capability Maturity Model. **23. Software cost estimation.**

Productivity.

Estimation techniques.

Algorithmic cost modelling.

Project duration and staffing. **24. Quality management.**

Quality assurance and standards.

Quality planning.

Quality control.

Software measurement and metrics. **25. Process Improvement.**

Process and product quality.

Process analysis and modelling.

Process measurement.

The SEI Process Capability Maturity Model.

Process classification.

**VII. EVOLUTION. 26. Legacy Systems.**

Legacy system structures.
Legacy system design.

Legacy system assessment. 27. Software change.

Program evolution dynamics.

Software maintenance.

Architectural evolution. 28. Software re-engineering.

Source code translation.

Reverse engineering.

Program structure improvement.

Program modularisation.

Data re-engineering. 29. Configuration management.

Configuration management planning.

Change management.

Version and release management.

System building.

CASE tools for configuration management. References.

Index.

What Our Readers Are Saying
Add a comment for a chance to win!

Average customer rating 4.5 (2 comments)