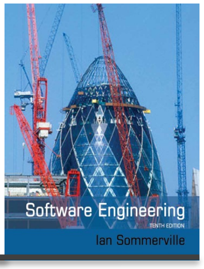
Verification, Validation, and Software Quality Management

CS413 - Software Engineering Project Management

Department of Computer Engineering, Bilkent University

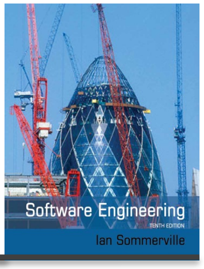
Dr. Mustafa Değerli

Bilkent University



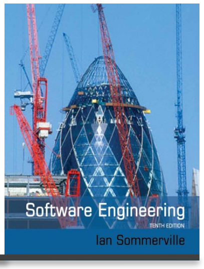
Chapter 8 – Software Testing

Program testing



- ✧ Testing is intended to show that a program does what it is intended to do and to discover program defects before it is put into use.
- ✧ When you test software, you execute a program using artificial data.
- ✧ You check the results of the test run for errors, anomalies or information about the program’s non-functional attributes.
- ✧ Can reveal the presence of errors NOT their absence.
- ✧ Testing is part of a more general verification and validation process, which also includes static validation techniques.

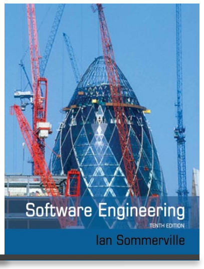
Program testing goals



✧ To demonstrate to the developer and the customer that the software meets its requirements.

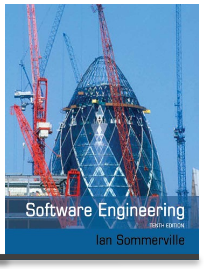
✧ To discover situations in which the behavior of the software is incorrect, undesirable or does not conform to its specification.

Testing process goals



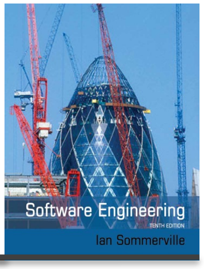
- ✧ Validation testing
  - To demonstrate to the developer and the system customer that the software meets its requirements
  - A successful test shows that the system operates as intended.
- ✧ Defect testing
  - To discover faults or defects in the software where its behaviour is incorrect or not in conformance with its specification
  - A successful test is a test that makes the system perform incorrectly and so exposes a defect in the system.

Verification vs validation

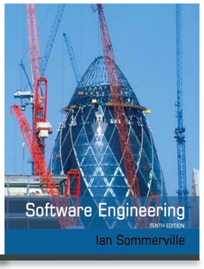


- ✧ Verification:
  "Are we building the product right”.
- ✧ The software should conform to its specification.
- ✧ Validation:
  "Are we building the right product”.
- ✧ The software should do what the user really requires.

V & V confidence

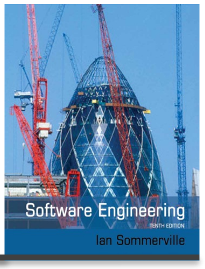


- ✧ Aim of V & V is to establish confidence that the system is 'fit for purpose'.
- ✧ Depends on system's purpose, user expectations and marketing environment
  - ▪ Software purpose
    - The level of confidence depends on how critical the software is to an organisation.
  - ▪ User expectations
    - Users may have low expectations of certain kinds of software.
  - ▪ Marketing environment
    - Getting a product to market early may be more important than finding defects in the program.

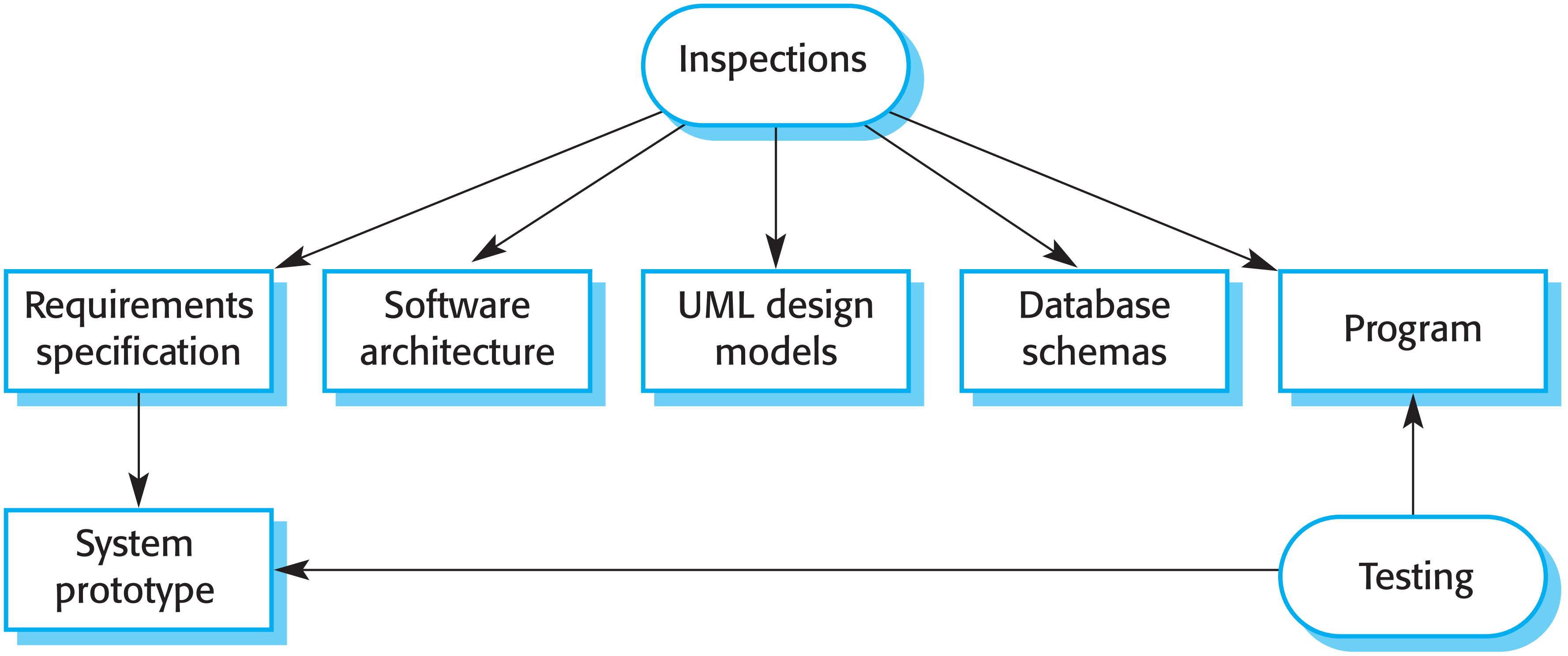


Inspections and testing

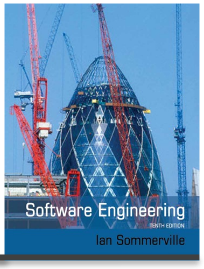
- ✧ Software inspections Concerned with analysis of the static system representation to discover problems (static verification)
  - May be supplement by tool-based document and code analysis.
- ✧ Software testing Concerned with exercising and observing product behaviour (dynamic verification)
  - The system is executed with test data and its operational behaviour is observed.



Inspections and testing

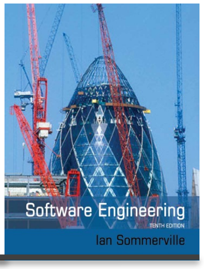


Software inspections



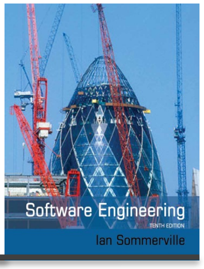
- These involve people examining the source representation with the aim of discovering anomalies and defects.
- Inspections not require execution of a system so may be used before implementation.
- They may be applied to any representation of the system (requirements, design,configuration data, test data, etc.).
- They have been shown to be an effective technique for discovering program errors.

Advantages of inspections

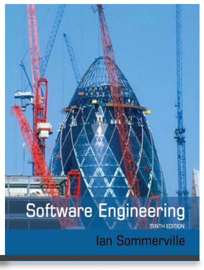


- ✧ During testing, errors can mask (hide) other errors. Because inspection is a static process, you don't have to be concerned with interactions between errors.
- ✧ Incomplete versions of a system can be inspected without additional costs. If a program is incomplete, then you need to develop specialized test harnesses to test the parts that are available.
- ✧ As well as searching for program defects, an inspection can also consider broader quality attributes of a program, such as compliance with standards, portability and maintainability.

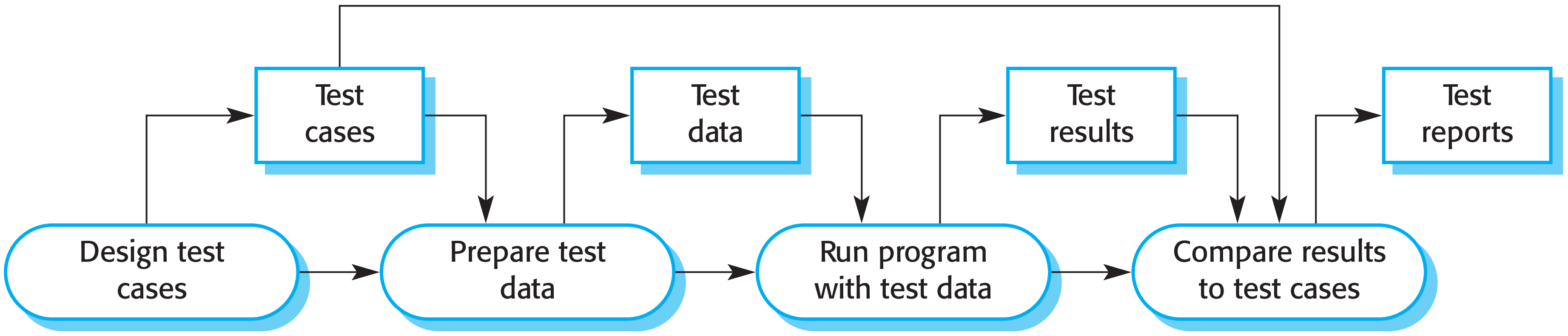
Inspections and testing



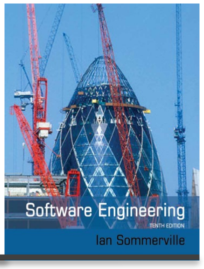
- ✧ Inspections and testing are complementary and not opposing verification techniques.
- ✧ Both should be used during the V & V process.
- ✧ Inspections can check conformance with a specification but not conformance with the customer’s real requirements.
- ✧ Inspections cannot check non-functional characteristics such as performance, usability, etc.



A model of the software testing process

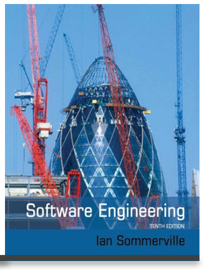


Stages of testing



- ✧ Development testing, where the system is tested during development to discover bugs and defects.
- ✧ Release testing, where a separate testing team test a complete version of the system before it is released to users.
- ✧ User testing, where users or potential users of a system test the system in their own environment.

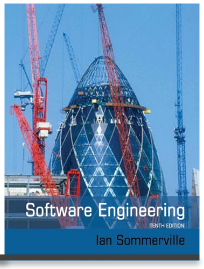
Development testing



✧ Development testing includes all testing activities that are carried out by the team developing the system.

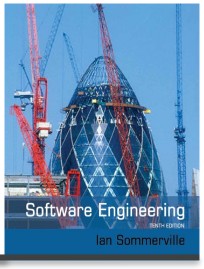
- Unit testing, where individual program units or object classes are tested. Unit testing should focus on testing the functionality of objects or methods.
- Component testing, where several individual units are integrated to create composite components. Component testing should focus on testing component interfaces.
- System testing, where some or all of the components in a system are integrated and the system is tested as a whole. System testing should focus on testing component interactions.

Release testing



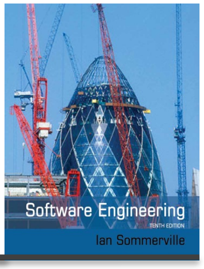
- ✧ Release testing is the process of testing a particular release of a system that is intended for use outside of the development team.
- ✧ The primary goal of the release testing process is to convince the supplier of the system that it is good enough for use.
  - Release testing, therefore, has to show that the system delivers its specified functionality, performance and dependability, and that it does not fail during normal use.
- ✧ Release testing is usually a black-box testing process where tests are only derived from the system specification.

Release testing and system testing



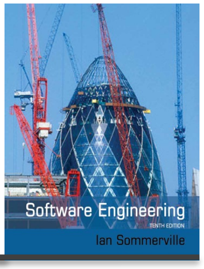
- ✧ Release testing is a form of system testing.
- ✧ Important differences:
  - A separate team that has not been involved in the system development, should be responsible for release testing.
  - System testing by the development team should focus on discovering bugs in the system (defect testing). The objective of release testing is to check that the system meets its requirements and is good enough for external use (validation testing).

User testing



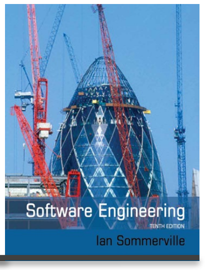
- ✧ User or customer testing is a stage in the testing process in which users or customers provide input and advice on system testing.
- ✧ User testing is essential, even when comprehensive system and release testing have been carried out.
  - ▪ The reason for this is that influences from the user’s working environment have a major effect on the reliability, performance, usability and robustness of a system. These cannot be replicated in a testing environment.

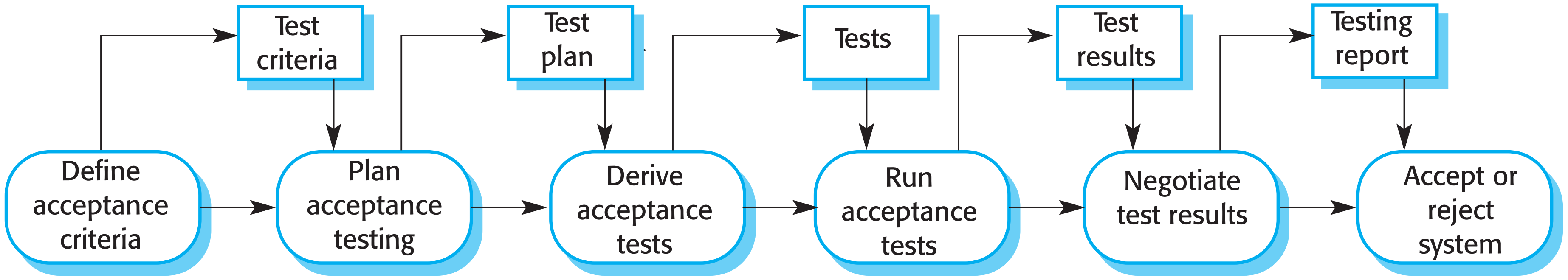
Types of user testing



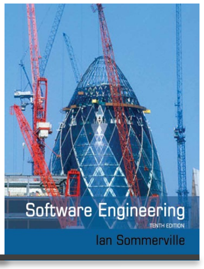
- Alpha testing
  - Users of the software work with the development team to test the software at the developer's site.
- Beta testing
  - A release of the software is made available to users to allow them to experiment and to raise problems that they discover with the system developers.
- Acceptance testing
  - Customers test a system to decide whether or not it is ready to be accepted from the system developers and deployed in the customer environment. Primarily for custom systems.

The acceptance testing process





Stages in the acceptance testing process



✧ Define acceptance criteria

✧ Plan acceptance testing

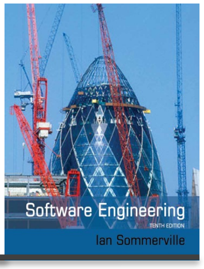
✧ Derive acceptance tests

✧ Run acceptance tests

✧ Negotiate test results

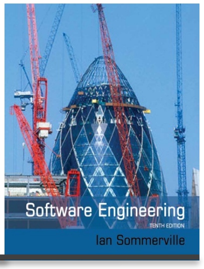
✧ Reject/accept system

Agile methods and acceptance testing



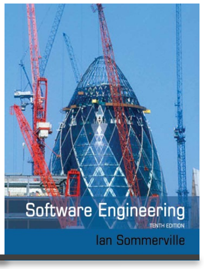
- ✧ In agile methods, the user/customer is part of the development team and is responsible for making decisions on the acceptability of the system.
- ✧ Tests are defined by the user/customer and are integrated with other tests in that they are run automatically when changes are made.
- ✧ There is no separate acceptance testing process.
- ✧ Main problem here is whether or not the embedded user is ‘typical’ and can represent the interests of all system stakeholders.

Key points

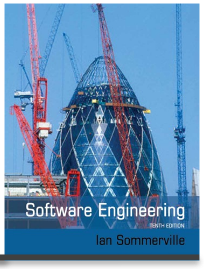


- ✧ Testing can only show the presence of errors in a program. It cannot demonstrate that there are no remaining faults.
- ✧ Development testing is the responsibility of the software development team. A separate team should be responsible for testing a system before it is released to customers.
- ✧ Development testing includes unit testing, in which you test individual objects and methods component testing in which you test related groups of objects and system testing, in which you test partial or complete systems.

Key points

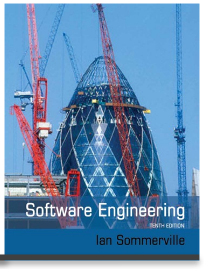


- When testing software, you should try to 'break' the software by using experience and guidelines to choose types of test case that have been effective in discovering defects in other systems.
- Wherever possible, you should write automated tests. The tests are embedded in a program that can be run every time a change is made to a system.
- Test-first development is an approach to development where tests are written before the code to be tested.
- Scenario testing involves inventing a typical usage scenario and using this to derive test cases.
- Acceptance testing is a user testing process where the aim is to decide if the software is good enough to be deployed and used in its operational environment.



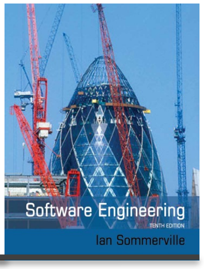
Chapter 24 - Quality Management

Software quality management



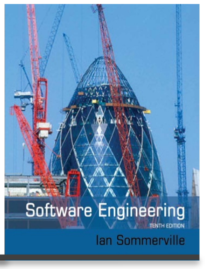
- ✧ Concerned with ensuring that the required level of quality is achieved in a software product.
- ✧ Three principal concerns:
  - At the organizational level, quality management is concerned with establishing a framework of organizational processes and standards that will lead to high-quality software.
  - At the project level, quality management involves the application of specific quality processes and checking that these planned processes have been followed.
  - At the project level, quality management is also concerned with establishing a quality plan for a project. The quality plan should set out the quality goals for the project and define what processes and standards are to be used.

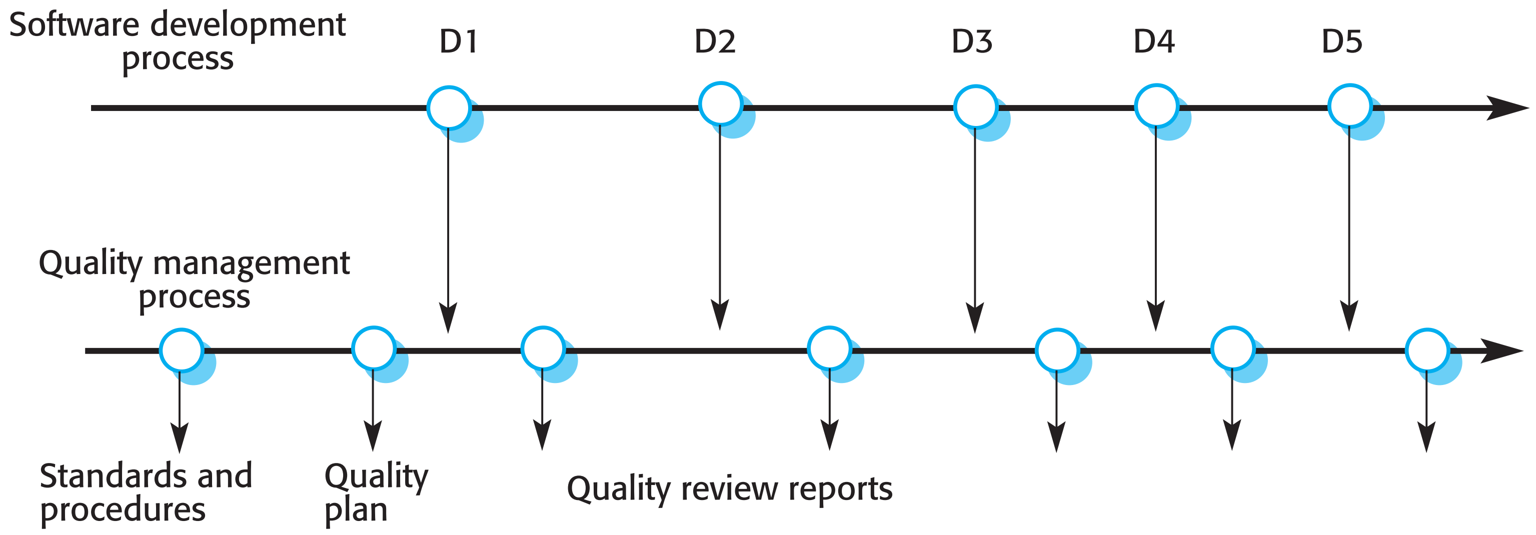
Quality management activities



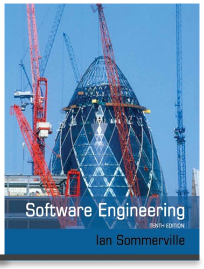
- ✧ Quality management provides an independent check on the software development process.
- ✧ The quality management process checks the project deliverables to ensure that they are consistent with organizational standards and goals
- ✧ The quality team should be independent from the development team so that they can take an objective view of the software. This allows them to report on software quality without being influenced by software development issues.

Quality management and software development





Quality planning

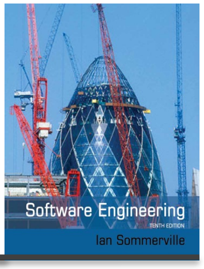


✧ A quality plan sets out the desired product qualities and how these are assessed and defines the most significant quality attributes.

✧ The quality plan should define the quality assessment process.

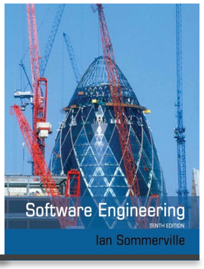
✧ It should set out which organisational standards should be applied and, where necessary, define new standards to be used.

Quality plans



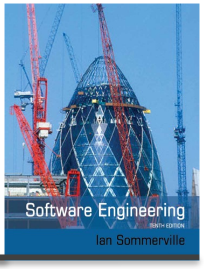
- ✧ Quality plan structure
  - Product introduction;
  - Product plans;
  - Process descriptions;
  - Quality goals;
  - Risks and risk management.
- ✧ Quality plans should be short, succinct documents
  - If they are too long, no-one will read them.

Scope of quality management



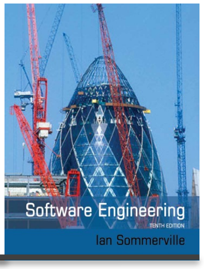
- ✧ Quality management is particularly important for large, complex systems. The quality documentation is a record of progress and supports continuity of development as the development team changes.
- ✧ For smaller systems, quality management needs less documentation and should focus on establishing a quality culture.
- ✧ Techniques have to evolve when agile development is used.

Software quality



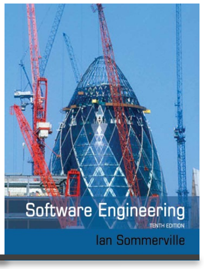
- Quality, simplistically, means that a product should meet its specification.
- This is problematical for software systems
  - There is a tension between customer quality requirements (efficiency, reliability, etc.) and developer quality requirements (maintainability, reusability, etc.);
  - Some quality requirements are difficult to specify in an unambiguous way;
  - Software specifications are usually incomplete and often inconsistent.
- The focus may be ‘fitness for purpose’ rather than specification conformance.

Software fitness for purpose



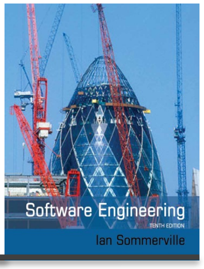
- ✧ Has the software been properly tested?
- ✧ Is the software sufficiently dependable to be put into use?
- ✧ Is the performance of the software acceptable for normal use?
- ✧ Is the software usable?
- ✧ Is the software well-structured and understandable?
- ✧ Have programming and documentation standards been followed in the development process?

Non-functional characteristics



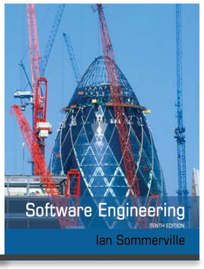
- ✧ The subjective quality of a software system is largely based on its non-functional characteristics.
- ✧ This reflects practical user experience – if the software's functionality is not what is expected, then users will often just work around this and find other ways to do what they want to do.
- ✧ However, if the software is unreliable or too slow, then it is practically impossible for them to achieve their goals.

Software quality attributes



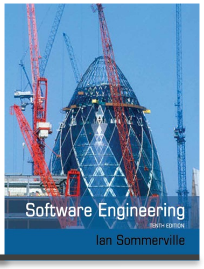
Safety	Understandability	Portability
Security	Testability	Usability
Reliability	Adaptability	Reusability
Resilience	Modularity	Efficiency
Robustness	Complexity	Learnability

Process and product quality



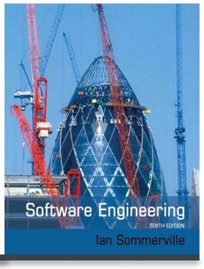
- ✧ The quality of a developed product is influenced by the quality of the production process.
- ✧ This is important in software development as some product quality attributes are hard to assess.
- ✧ However, there is a very complex and poorly understood relationship between software processes and product quality.
  - ▪ The application of individual skills and experience is particularly important in software development;
  - ▪ External factors such as the novelty of an application or the need for an accelerated development schedule may impair product quality.

Quality culture



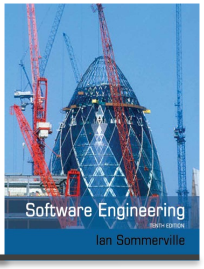
- ✧ Quality managers should aim to develop a ‘quality culture’ where everyone responsible for software development is committed to achieving a high level of product quality.
- ✧ They should encourage teams to take responsibility for the quality of their work and to develop new approaches to quality improvement.
- ✧ They should support people who are interested in the intangible aspects of quality and encourage professional behavior in all team members.

Reviews and inspections



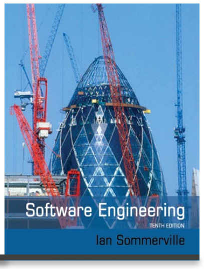
- ✧ A group examines part or all of a process or system and its documentation to find potential problems.
- ✧ Software or documents may be 'signed off' at a review which signifies that progress to the next development stage has been approved by management.
- ✧ There are different types of review with different objectives
  - Inspections for defect removal (product);
  - Reviews for progress assessment (product and process);
  - Quality reviews (product and standards).

Quality reviews



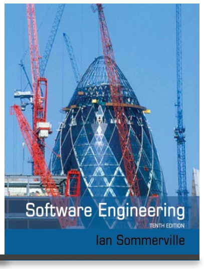
- ✧ A group of people carefully examine part or all of a software system and its associated documentation.
- ✧ Code, designs, specifications, test plans, standards, etc. can all be reviewed.
- ✧ Software or documents may be 'signed off' at a review which signifies that progress to the next development stage has been approved by management.

Phases in the review process



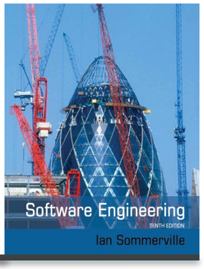
- ✧ Pre-review activities
  - Pre-review activities are concerned with review planning and review preparation
- ✧ The review meeting
  - During the review meeting, an author of the document or program being reviewed should 'walk through' the document with the review team.
- ✧ Post-review activities
  - These address the problems and issues that have been raised during the review meeting.

Quality management and agile development



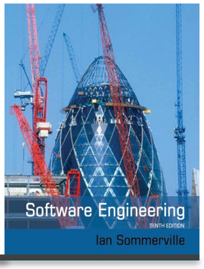
- ✧ Quality management in agile development is informal rather than document-based.
- ✧ It relies on establishing a quality culture, where all team members feel responsible for software quality and take actions to ensure that quality is maintained.

Shared good practice



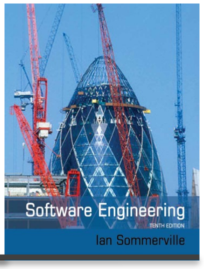
- *Check before check-in*
  - Programmers are responsible for organizing their own code reviews with other team members before the code is checked in to the build system.
- *Never break the build*
  - Team members should not check in code that causes the system to fail. Developers have to test their code changes against the whole system and be confident that these work as expected.
- *Fix problems when you see them*
  - If a programmer discovers problems or obscurities in code developed by someone else, they can fix these directly rather than referring them back to the original developer.

Reviews and agile methods



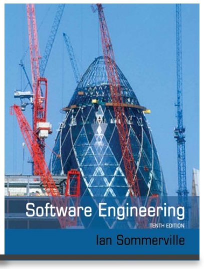
- ✧ The review process in agile software development is usually informal.
- ✧ In Scrum,, there is a review meeting after each iteration of the software has been completed (a sprint review), where quality issues and problems may be discussed.
- ✧ In Extreme Programming, pair programming ensures that code is constantly being examined and reviewed by another team member.

Software measurement



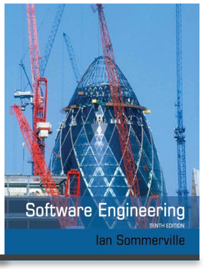
- ✧ Software measurement is concerned with deriving a numeric value for an attribute of a software product or process.
- ✧ This allows for objective comparisons between techniques and processes.
- ✧ Although some companies have introduced measurement programmes, most organisations still don’t make systematic use of software measurement.
- ✧ There are few established standards in this area.

Software metric



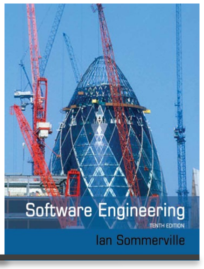
- ✧ Any type of measurement which relates to a software system, process or related documentation
  - Lines of code in a program, the Fog index, number of person-days required to develop a component.
- ✧ Allow the software and the software process to be quantified.
- ✧ May be used to predict product attributes or to control the software process.
- ✧ Product metrics can be used for general predictions or to identify anomalous components.

Key points



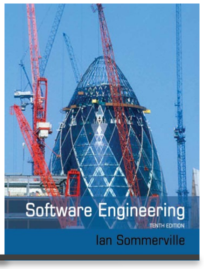
- ✧ Software quality management is concerned with ensuring that software has a low number of defects and that it reaches the required standards of maintainability, reliability, portability etc. Software standards are important for quality assurance as they represent an identification of 'best practice'. When developing software, standards provide a solid foundation for building good quality software.
- ✧ Reviews of the software process deliverables involve a team of people who check that quality standards are being followed. Reviews are the most widely used technique for assessing quality.

Key points



- ✧ In a program inspection or peer review, a small team systematically checks the code. They read the code in detail and look for possible errors and omissions. The problems detected are discussed at a code review meeting.
- ✧ Agile quality management relies on establishing a quality culture where the development team works together to improve software quality.
- ✧ Software measurement can be used to gather quantitative data about software and the software process.

Key points



- ✧ You may be able to use the values of the software metrics that are collected to make inferences about product and process quality.
- ✧ Product quality metrics are particularly useful for highlighting anomalous components that may have quality problems. These components should then be analyzed in more detail.
- ✧ Software analytics is the automated analysis of large volumes of software product and process data to discover relationships that may provide insights for project managers and developers.

Reference

- Software Engineering, 10th Edition, Ian Sommerville.

Verification, Validation, and Software Quality Management

CS413 - Software Engineering Project Management

Department of Computer Engineering, Bilkent University

Dr. Mustafa Değerli

Bilkent University