Quality Management

Chapter 27
Sommerville 8th Edition

Objectives

- To introduce the quality management process and key quality management activities
- To explain the role of standards in quality management
- To explain the concept of a software metric, predictor metrics and control metrics
- To explain how measurement may be used in assessing software quality and the limitations of software measurement

Topics Covered

- Process and product quality
- Quality assurance and standards
- Quality planning
- Quality control

Software Quality Management

- Concerned with ensuring that the required level of quality is achieved in a software product.
- Involves defining appropriate quality standards and procedures and ensuring that these are followed.
- Should aim to develop a ‘quality culture’ where quality is seen as everyone’s responsibility.

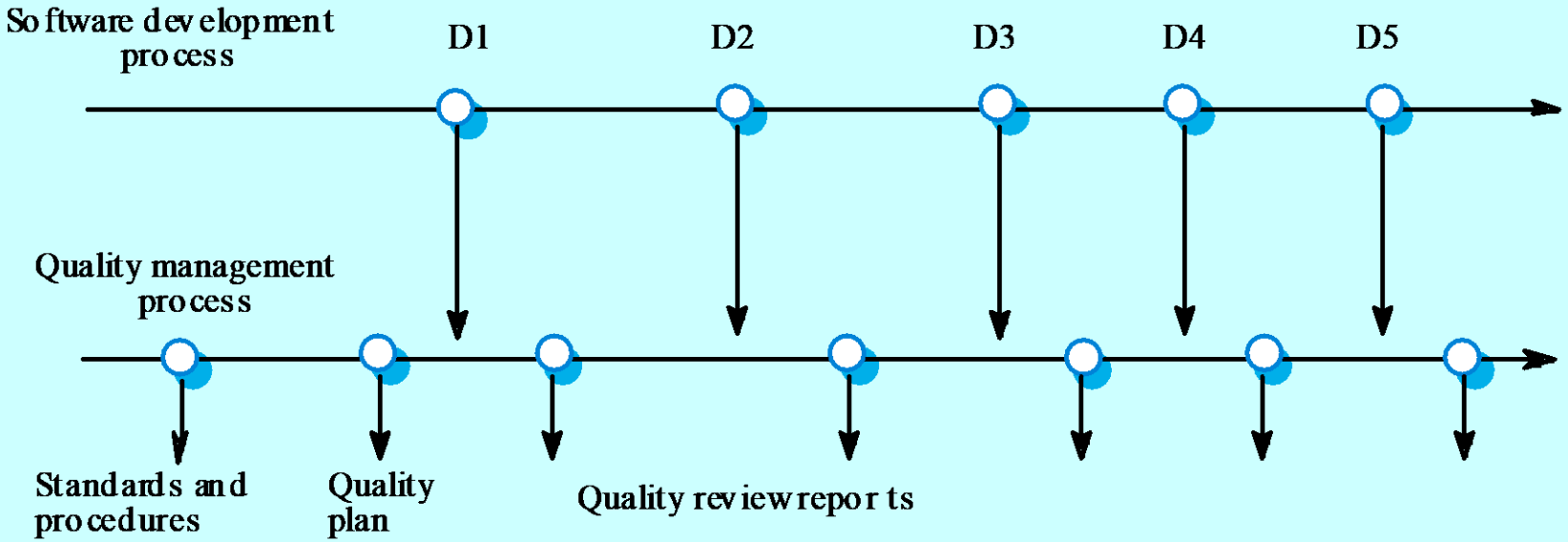
What is Quality?

- 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.

Quality Management Activities

- Quality Assurance
  - Establish organisational procedures and standards for quality.
- Quality Planning
  - Select applicable procedures and standards for a particular project and modify these as required.
- Quality Control
  - Ensure that procedures and standards are followed by the software development team.
- Quality management should be separate from project management to ensure independence.

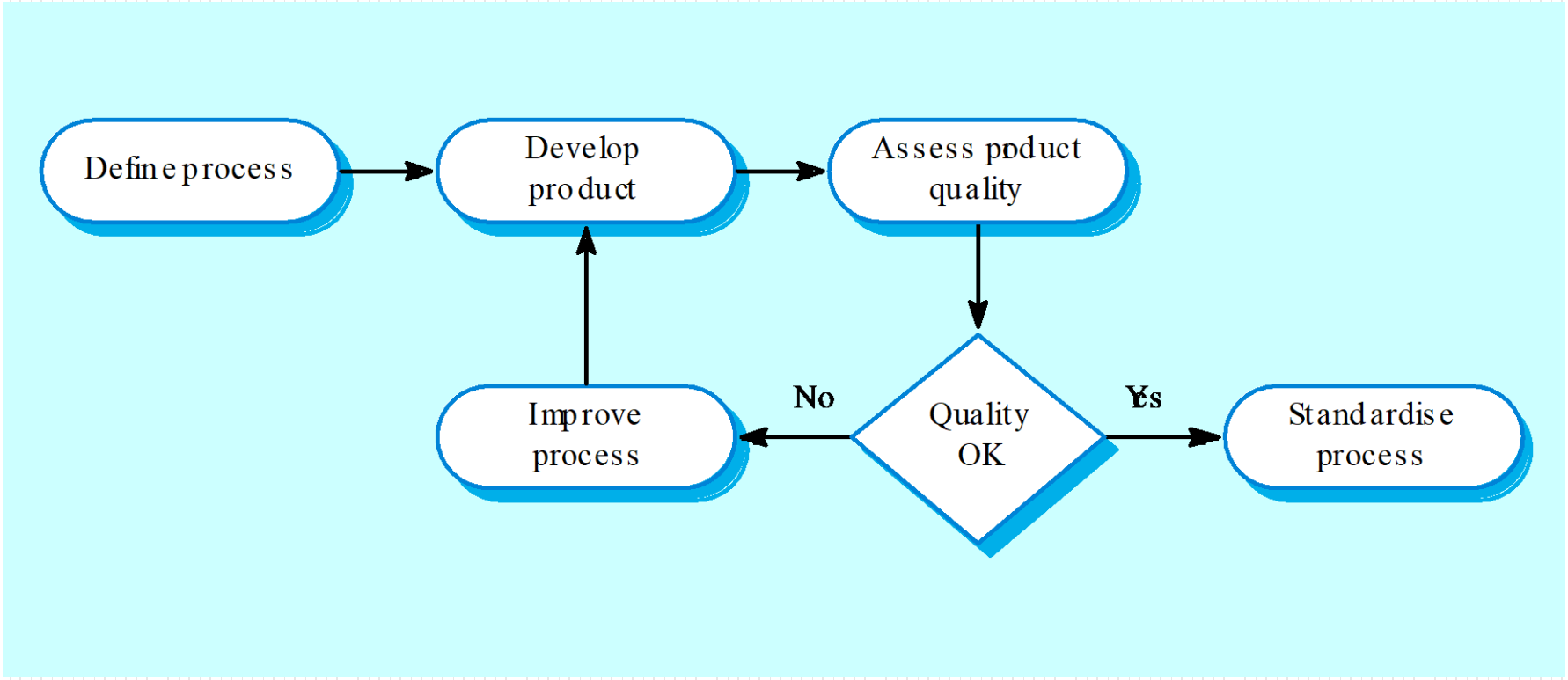
Quality Management and Software Development



Process-based quality

- There is a straightforward link between process and product in manufactured goods.
- More complex for software because:
  - 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.
- Care must be taken not to impose inappropriate process standards - these could reduce rather than improve the product quality.

Process-based quality



Practical process quality

- Define process standards such as how reviews should be conducted, configuration management, etc.
- Monitor the development process to ensure that standards are being followed.
- Report on the process to project management and software procurer.
- Don’t use inappropriate practices simply because standards have been established.

Quality assurance and standards

- Standards are the key to effective quality management.
- They may be international, national, organizational or project standards.
- Product standards define characteristics that all components should exhibit e.g. a common programming style.
- Process standards define how the software process should be enacted.

Product and process standards

Product standards	Process standards
Design review form	Design review conduct
Requirements document structure	Submission of documents to CM
Method header format	Version release process
Java programming style	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

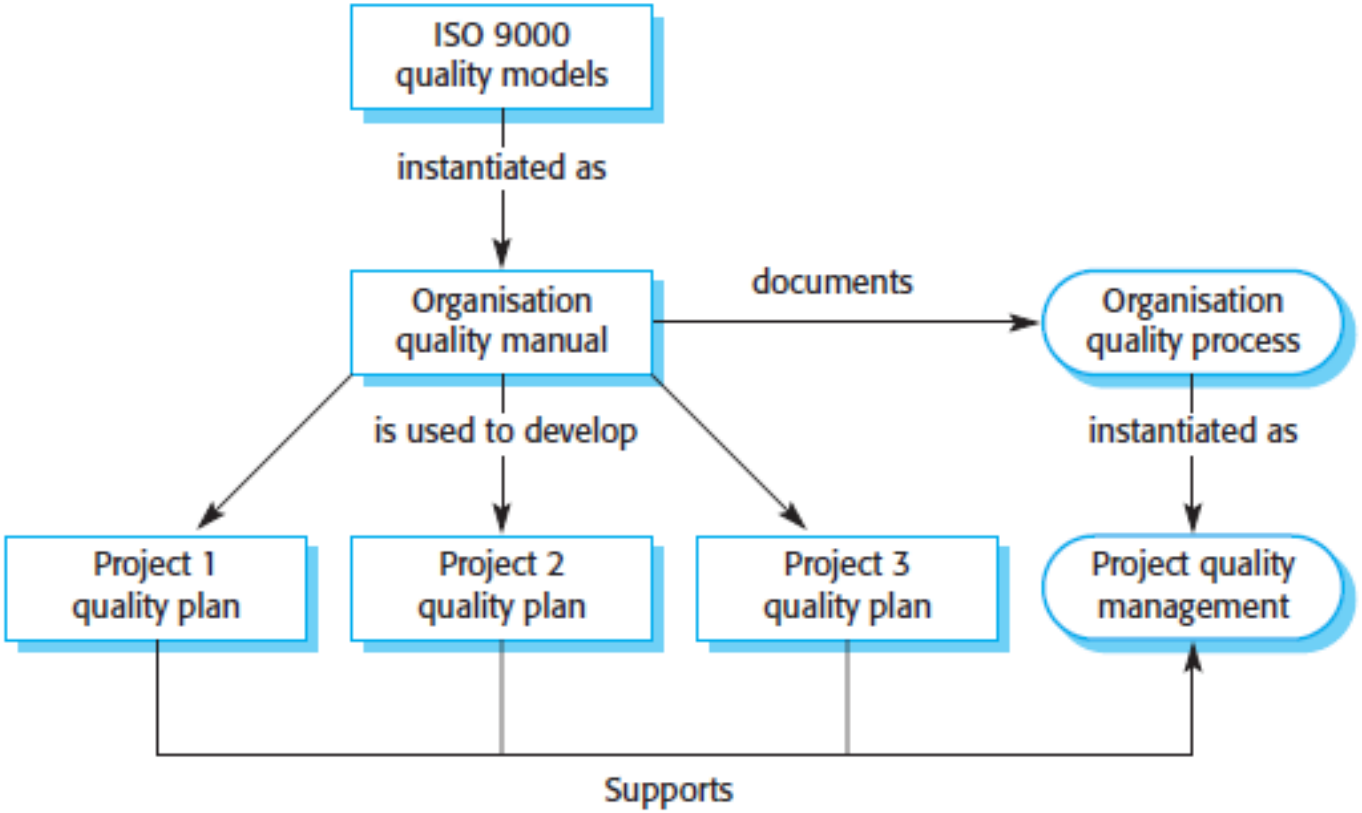
ISO 9000

- An international set of standards for quality management.
- Applicable to a range of organisations from manufacturing to service industries.
- ISO 9000 standards can be applied to a range of organisations from manufacturing to service industries.
- ISO 9001 is the most general of these standards and applies to organisations concerned with the quality process in organisations that design, develop and maintain products.

ISO 9000 certification

- Quality standards and procedures should be documented in an organisational quality manual.
- An external body may certify that an organisation's quality manual conforms to ISO 9000 standards.
- Some customers require suppliers to be ISO 9000 certified although the need for flexibility here is increasingly recognised.

ISO 9000 and quality management



ISO 9001

Management responsibility	Quality system
Control of nonconforming products	Design control
Handling, storage, packaging and delivery	Purchasing
Purchaser-supplied products	Product identification and traceability
Process control	Inspection and testing
Inspection and test equipment	Inspection and test status
Contract review	Corrective action
Document control	Quality records
Internal quality audits	Training
Servicing	Statistical techniques

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.

Software quality attributes

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

Quality control

- This involves checking the software development process to ensure that procedures and standards are being followed.
- There are two approaches to quality control
  - Quality reviews;
  - Automated software assessment and software measurement.

Quality reviews

- This is the principal method of validating the quality of a process or of a product.
- A group examines part or all of a process or system and its documentation to find potential problems.
- 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).

Types of review

Review type	Principal purpose
Design or program inspections	To detect detailed errors in the requirements, design or code. A checklist of possible errors should drive the review.
Progress reviews	To provide information for management about the overall progress of the project. This is both a process and a product review and is concerned with costs, plans and schedules.
Quality reviews	To carry out a technical analysis of product components or documentation to find mismatches between the specification and the component design, code or documentation and to ensure that defined quality standards have been followed.

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.

Quality reviews

- The objective is the discovery of system defects and inconsistencies.
- Any documents produced in the process may be reviewed.
- Review teams should be relatively small and reviews should be fairly short.
- Records should always be maintained of quality reviews.

Review results

- Comments made during the review should be classified
  - No action. No change to the software or documentation is required;
  - Refer for repair. Designer or programmer should correct an identified fault;
  - Reconsider overall design. The problem identified in the review impacts other parts of the design. Some overall judgement must be made about the most cost-effective way of solving the problem;
- Requirements and specification errors may have to be referred to the client.

Software measurement and metrics

- 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.

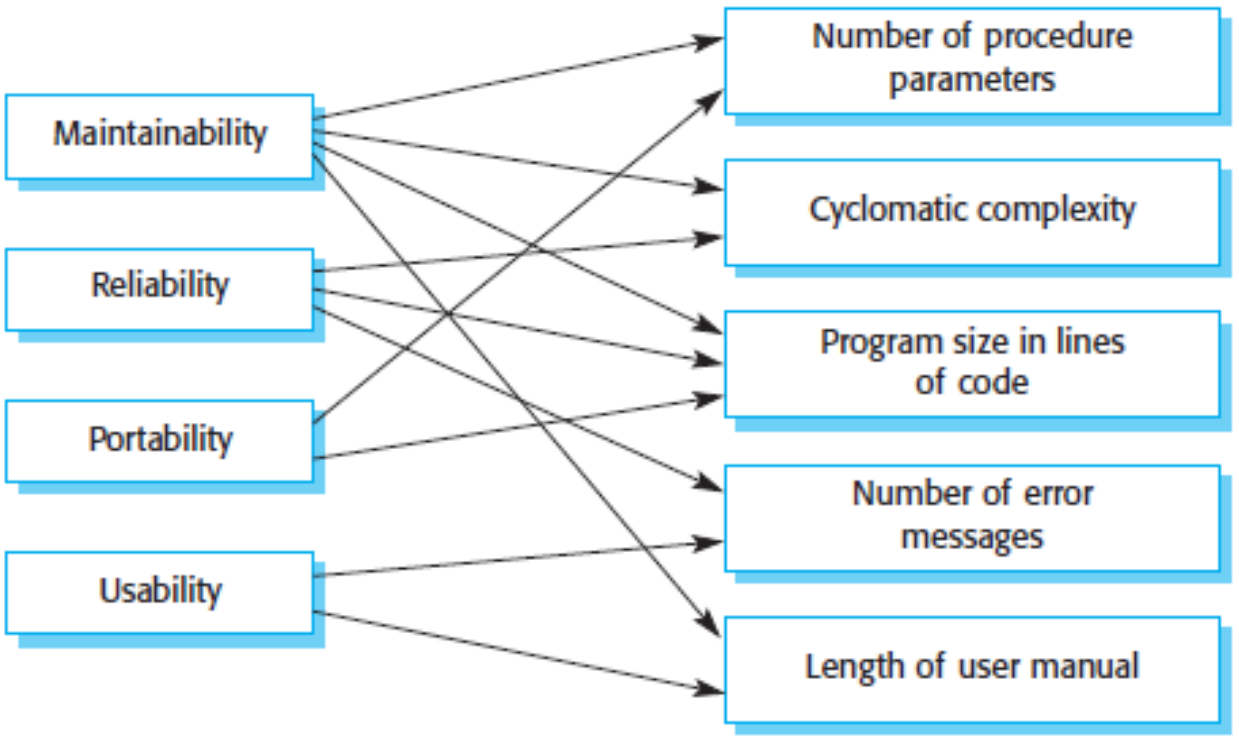
Software metric

- 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.

Metrics assumptions

- A software property can be measured.
- The relationship exists between what we can measure and what we want to know. We can only measure internal attributes but are often more interested in external software attributes.
- This relationship has been formalised and validated.
- It may be difficult to relate what can be measured to desirable external quality attributes.

Internal and external attributes



Product metrics

- A quality metric should be a predictor of product quality.
- Classes of product metric
  - Dynamic metrics which are collected by measurements made of a program in execution;
  - Static metrics which are collected by measurements made of the system representations;
  - Dynamic metrics help assess efficiency and reliability; static metrics help assess complexity, understandability and maintainability.

Dynamic and static metrics

- Dynamic metrics are closely related to software quality attributes
  - It is relatively easy to measure the response time of a system (performance attribute) or the number of failures (reliability attribute).
- Static metrics have an indirect relationship with quality attributes
  - You need to try and derive a relationship between these metrics and properties such as complexity, understandability and maintainability.

Static Software Product Metrics

Software metric	Description
Fan-in/Fan-out	*Fan-in* is a measure of the number of functions or methods that call some other function or method (say X). *Fan-out* is the number of functions that are called by function X. A high value for fan-in means that X is tightly coupled to the rest of the design and changes to X will have extensive knock-on effects. A high value for fan-out suggests that the overall complexity of X may be high because of the complexity of the control logic needed to coordinate the called components.
Length of code	This is a measure of the size of a program. Generally, the larger the size of the code of a component, the more complex and error-prone that component is likely to be. Length of code has been shown to be one of the most reliable metrics for predicting error-proneness in components.
Cyclomatic complexity	This is a measure of the control complexity of a program. This control complexity may be related to program understandability. I discuss how to compute cyclomatic complexity in Chapter 22.
Length of identifiers	This is a measure of the average length of distinct identifiers in a program. The longer the identifiers, the more likely they are to be meaningful and hence the more understandable the program.
Depth of conditional nesting	This is a measure of the depth of nesting of if-statements in a program. Deeply nested if statements are hard to understand and are potentially error-prone.
Fog index	This is a measure of the average length of words and sentences in documents. The higher the value for the Fog index, the more difficult the document is to understand.

Object-oriented metrics

Object-oriented metric	Description
Depth of inheritance tree	This represents the number of discrete levels in the inheritance tree where sub-classes inherit attributes and operations (methods) from super-classes. The deeper the inheritance tree, the more complex the design. Many object classes may have to be understood to understand the object classes at the leaves of the tree.
Method fan-in/fan-out	This is directly related to fan-in and fan-out, as described in Figure 27.12, and means essentially the same thing. However, it may be appropriate to make a distinction between calls from other methods within the object and calls from external methods.
Weighted methods per class	This is the number of methods included in a class, weighted by the complexity of each method. Therefore, a simple method may have a complexity of 1 and a large and complex method may have a much higher value. The larger the value for this metric, the more complex the object class. Complex objects are more likely to be more difficult to understand. They may not be logically cohesive so cannot be reused effectively as super-classes in an inheritance tree.
Number of overriding operations	This is the number of operations in a super-class that are overridden in a sub-class. A high value for this metric indicates that the super-class used may not be an appropriate parent for the sub-class.

Thank You!!

ANY QUESTIONS??