Chapter 24

Quality Management

Quality Management

- Managing the quality of the software process and products

Salesdata - sales_data.xlsx

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

Topics covered

- Quality assurance and standards
- Quality planning
- Quality control
- Software measurement and metrics

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
  - 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

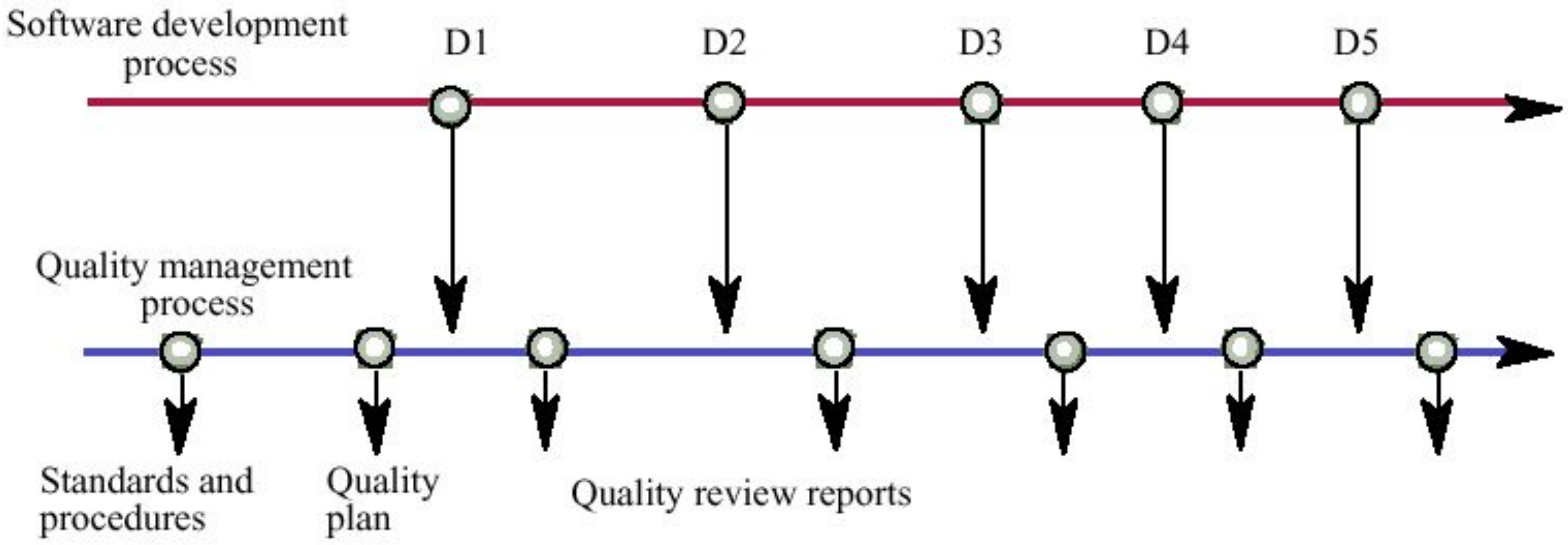
The quality compromise

- We cannot wait for specifications to improve before paying attention to quality management
- Must put procedures into place to improve quality in spite of imperfect specification
- Quality management is therefore not just concerned with reducing defects but also with other product qualities

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

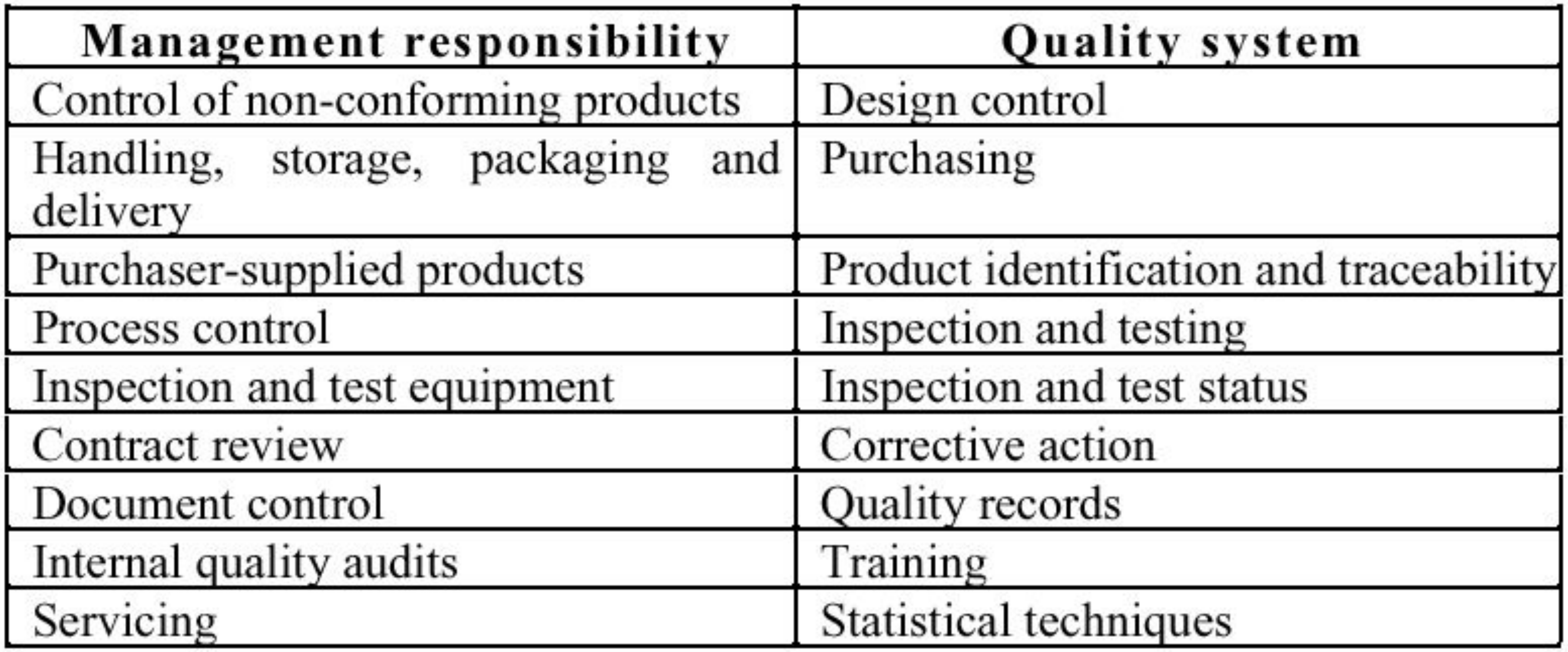
Quality management and software development



ISO 9000

- International set ofstandards for quality management
- Applicable to a range of organisations from manufacturing to service industries
- ISO 9001 applicable to organisations which design, develop and maintain products
- ISO 9001 is a generic model of the quality process.

ISO 9001

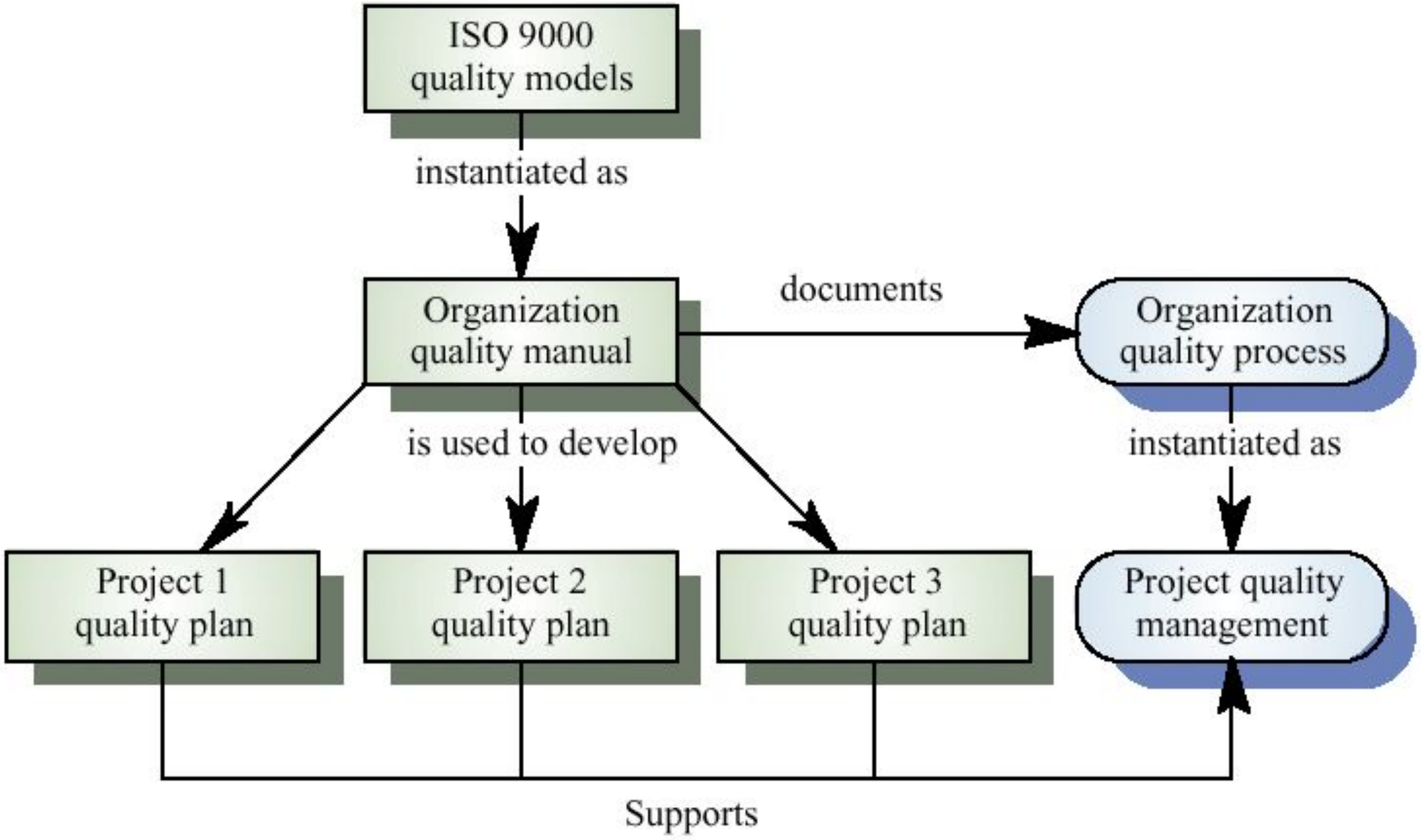


Management responsibility	Quality system
Control of non-conforming products	Design control
Handling, storage, packaging and delivery	Purchasing
Purchaser-supplied products	Product identification and traceability
Process control	Inspection and testing
Inspection and test equipment	Inspection and test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

ISO 9000 certification

- Quality standards and procedures should be documented in an organisational quality manual
- External body may certify that an organisation’s quality manual conforms to ISO 9000 standards
- Customers are, increasingly, demanding that suppliers are ISO 9000 certified

ISO 9000 and quality management



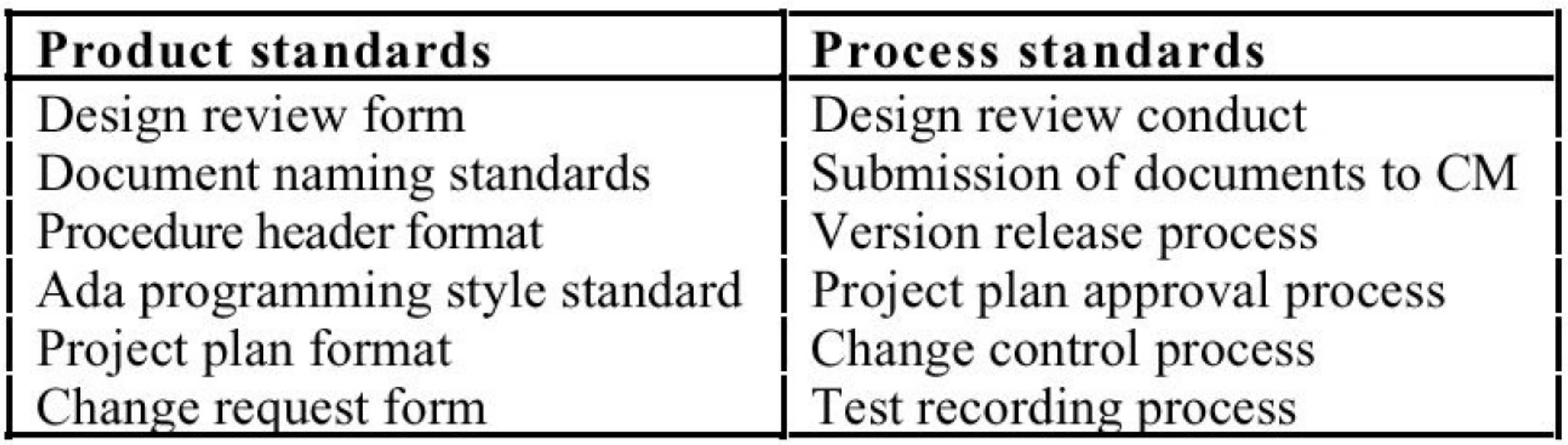
Quality assurance and standards

- 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

Importance of standards

- Encapsulation of best practice- avoids repetition of past mistakes
- Framework for quality assurance process - it involves checking standard compliance
- Provide continuity - new staff can understand the organisation by understand the standards applied

Product and process standards



Product standards	Process standards
Design review form	Design review conduct
Document naming standards	Submission of documents to CM
Procedure header format	Version release process
Ada programming style standard	Project plan approval process
Project plan format	Change control process
Change request form	Test recording process

Problems with standards

- Not seen as relevant and up-to-date by software engineers
- Involve too much bureaucratic form filling
- Unsupported by software tools so tedious manual work is involved to maintain standards

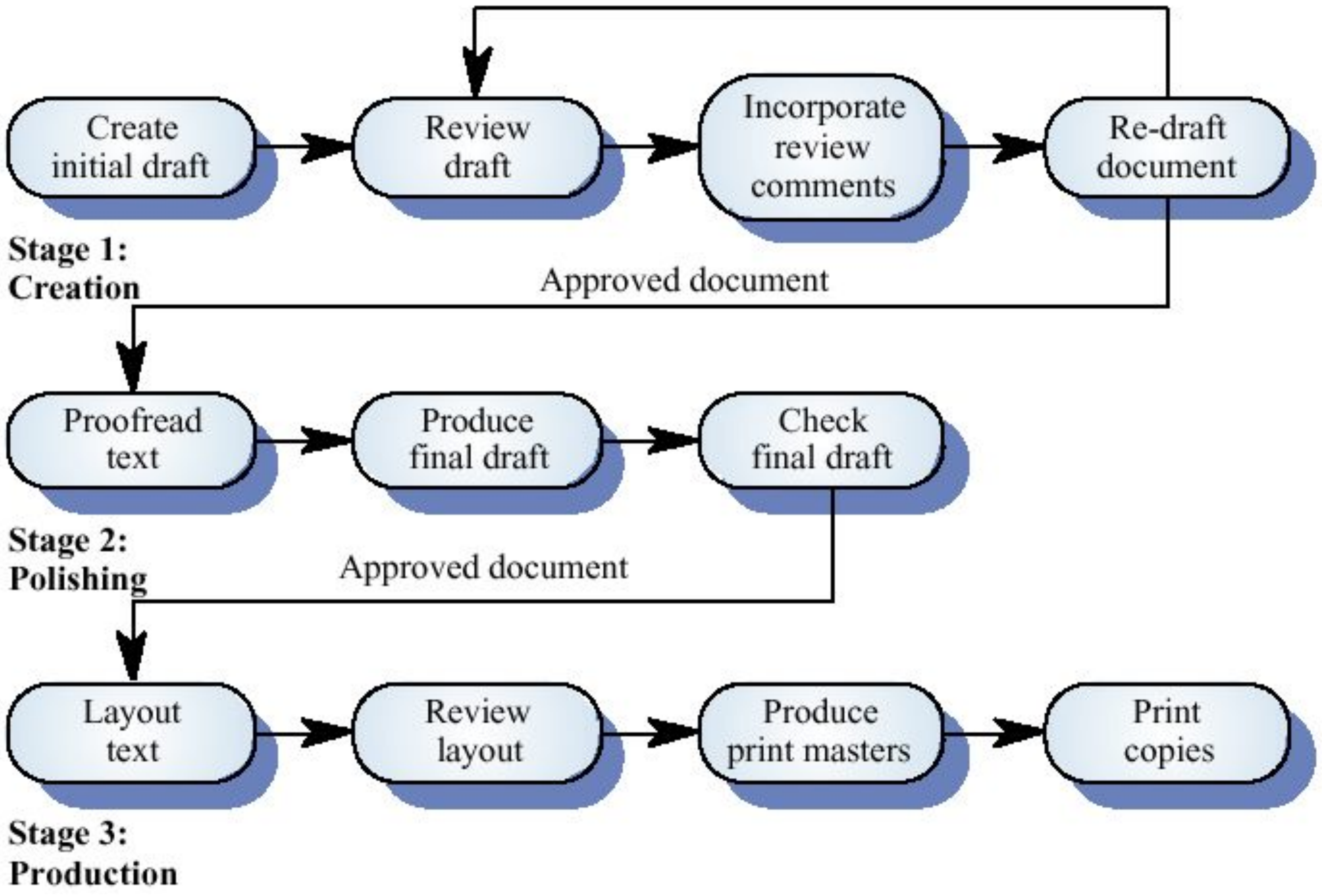
Standards development

- Involve practitioners in development. Engineers should understand the rationale underlying a standard
- Review standards and their usage regularly. Standards can quickly become outdated and this reduces their credibility amongst practitioners
- Detailed standards should have associated tool support. Excessive clerical work is the most significant complaint against standards

Documentation standards

- Particularly important - documents are the tangible manifestation of the software
- Documentation process standards
  - How documents should be developed, validated and maintained
- Document standards
  - Concerned with document contents, structure, and appearance
- Document interchange standards
  - How documents are stored and interchanged between different documentation systems

Documentation process



Document standards

- Document identification standards
  - How documents are uniquely identified
- Document structure standards
  - Standard structure for project documents
- Document presentation standards
  - Define fonts and styles, use of logos, etc.
- Document update standards
  - Define how changes from previous versions are reflected in a document

Document interchange standards

- Documents are produced using different systems and on different computers
- Interchange standards allow electronic documents to be exchanged, mailed, etc.
- Need for archiving. The lifetime of word processing systems may be much less than the lifetime of the software being documented
- XML is an emerging standard for document interchange which will be widely supported in future

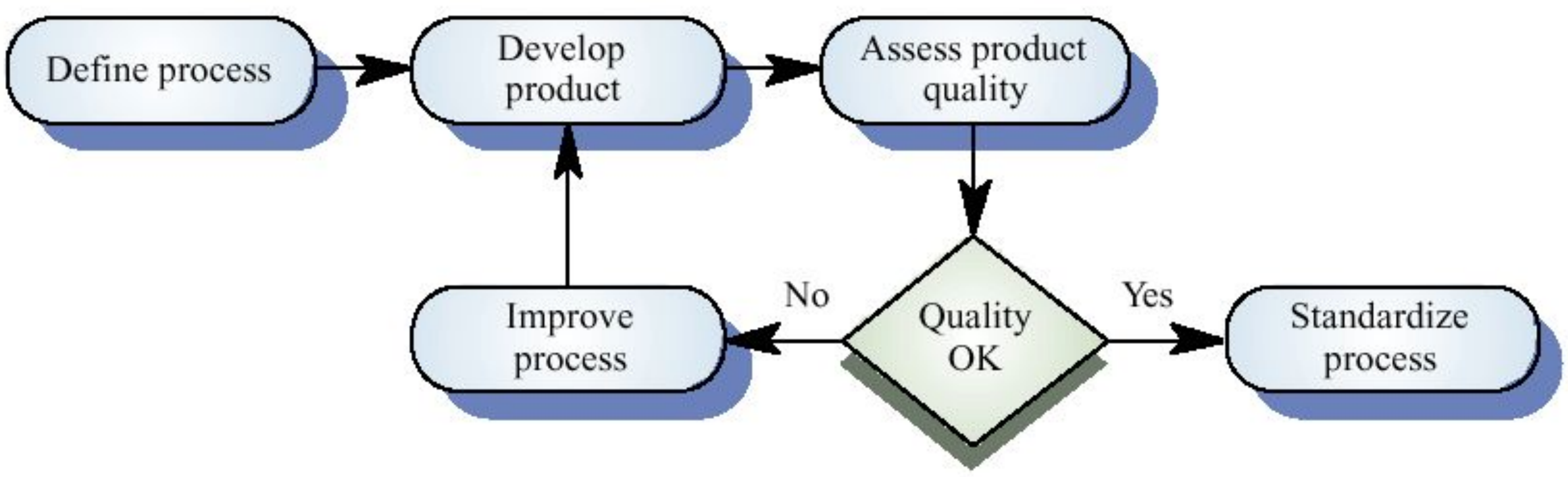
Process and product quality

- The quality of a developed product is influenced by the quality of the production process
- Particularly important in software development as some product quality attributes are hard to assess
- However, there is a very complex and poorly understood between software processes and product quality

Process-based quality

- Straightforward link between process and product in manufactured goods
- More complex for software because:
  - The application of individual skills and experience is particularly imporant 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

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

Quality planning

- A quality plan sets out the desired product qualities and how these are assessed ande define the most significant quality attributes
- It should define the quality assessment process
- It should set out which organisational standards should be applied and, if necessary, define new standards

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

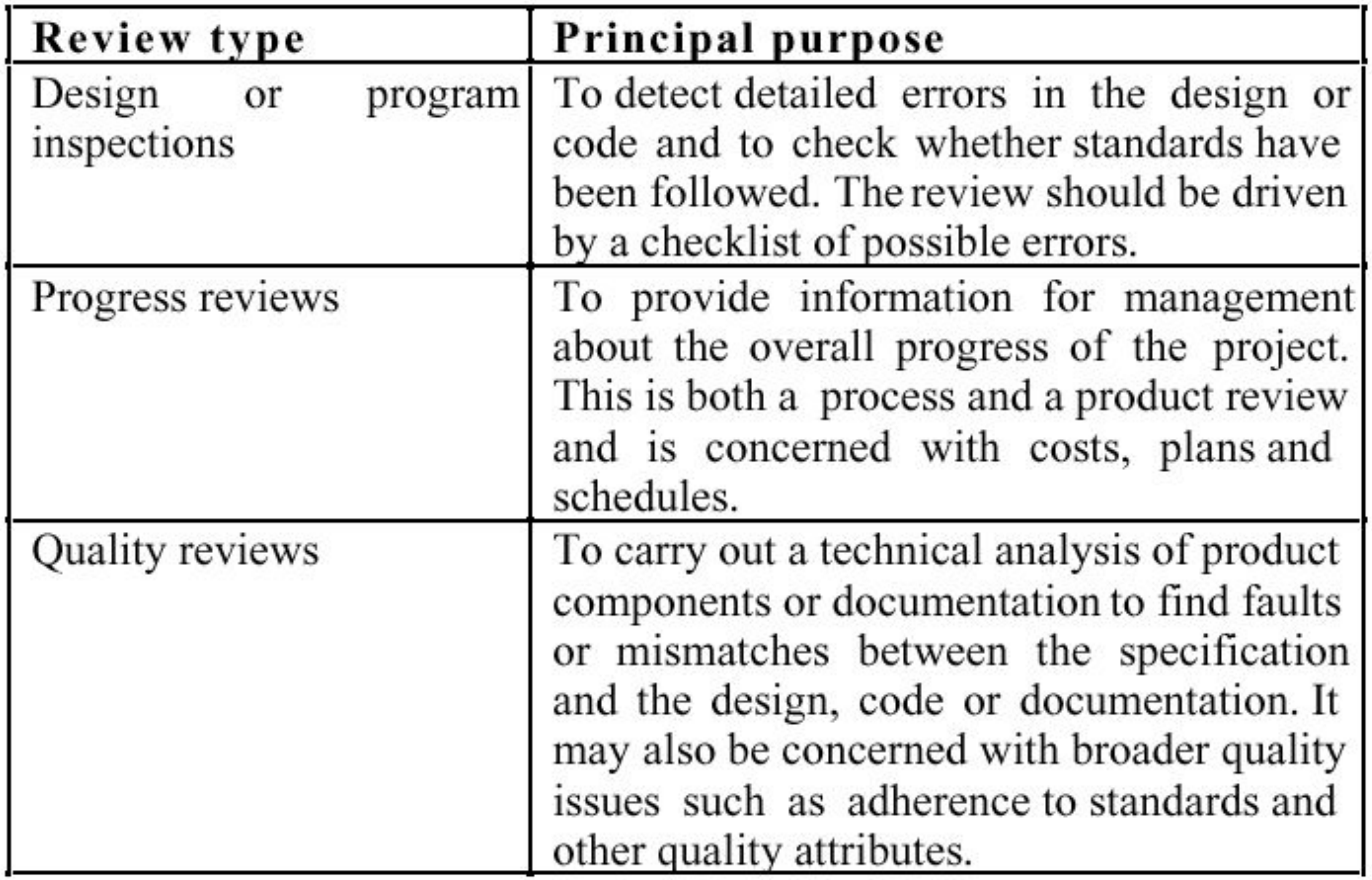
Quality control

- Checking the software development process to ensure that procedures and standards are being followed
- Two approaches to quality control
  - Quality reviews
  - Automated software assessment and software measurement

Quality reviews

- The principal method of validating the quality of a process or of a product
- Group examined 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

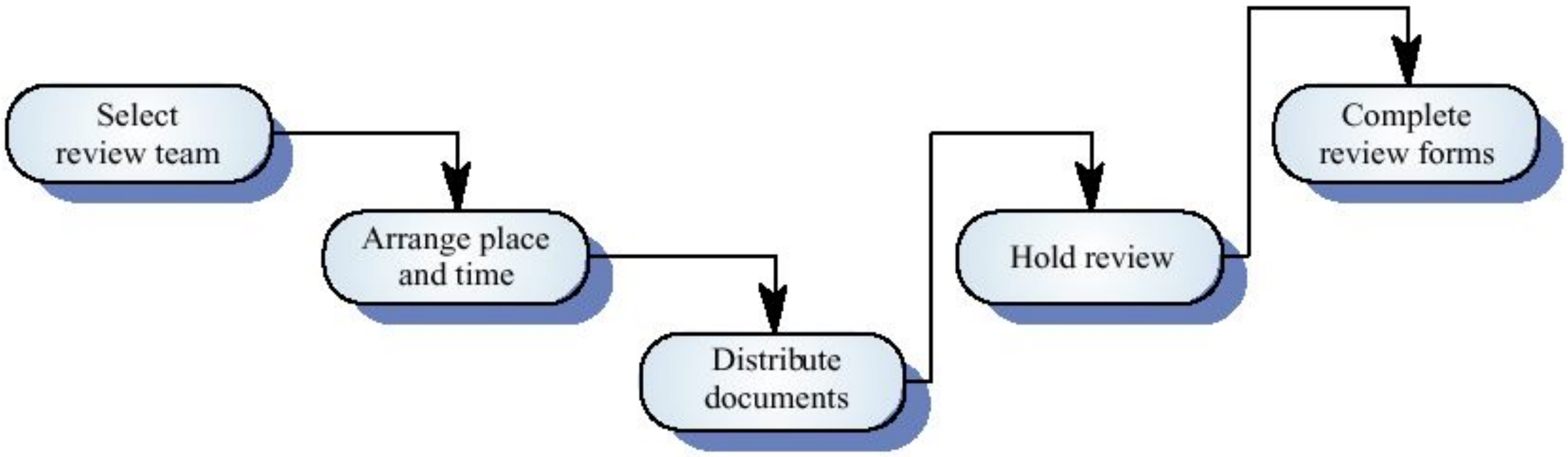


Review type	**Principal purpose**
Design or program inspections	To detect detailed errors in the design or code and to check whether standards have been followed. The review should be driven by a checklist of possible errors.
Progress reviews	To provide information for management about the overall progress 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 faults or mismatches between the specification and the design, code or documentation. It may also be concerned with broader quality issues such as adherence to standards and other quality attributes.

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.

The review process



Review functions

- Quality function - they are part of the general quality management process
- Project management function - they provide information for project managers
- Training and communication function - product knowledge is passed between development team members

Quality reviews

- 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
- Review should be recorded and records maintained

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.

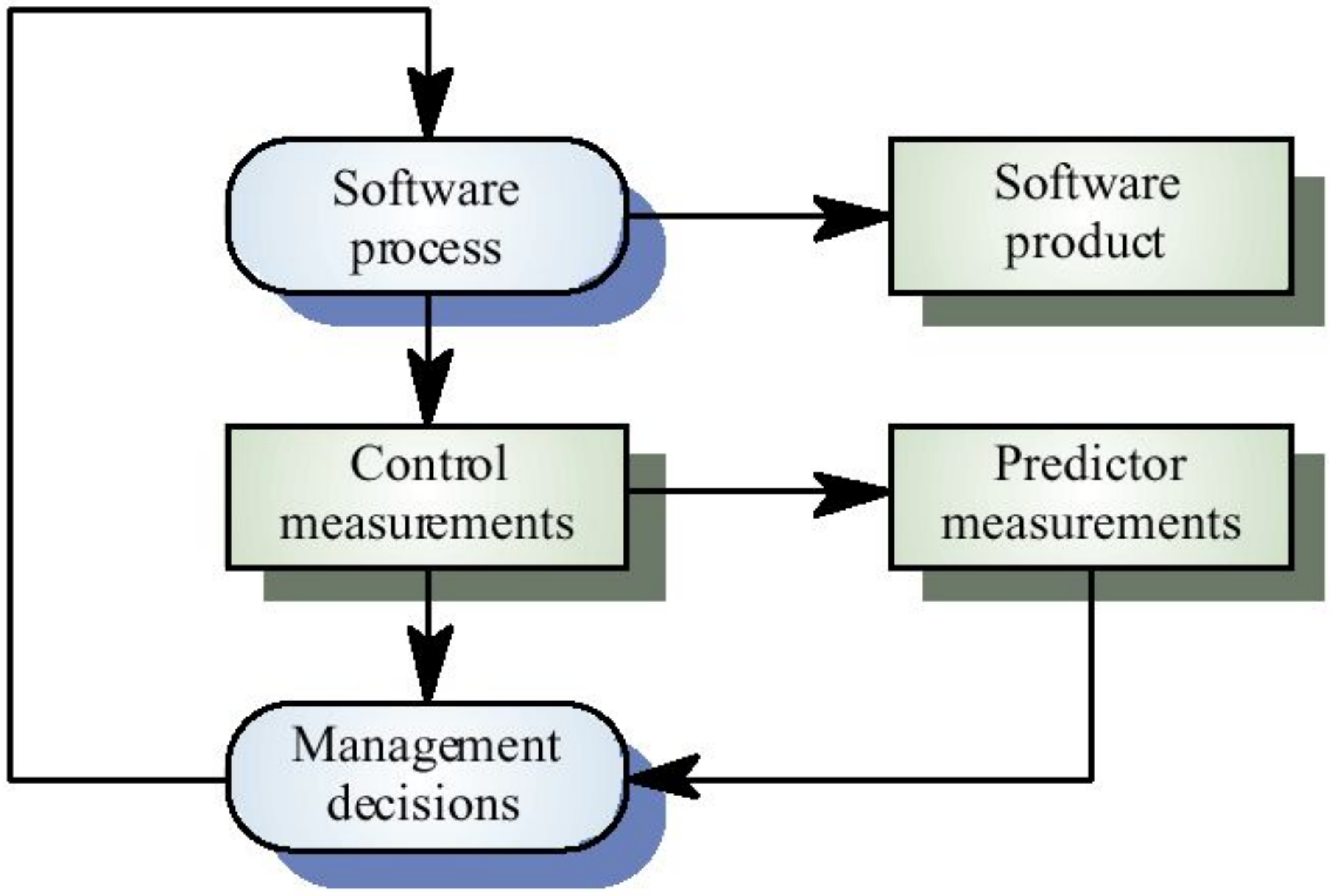
Software measurement and metrics

- 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 measurment programmes, the systematic use of measurement is still uncommon
- There are few 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, number of person-days required to develop a component
- Allow the software and the software process to be quantified
- Measures of the software process or product
- May be used to predict product attributes or to control the software process

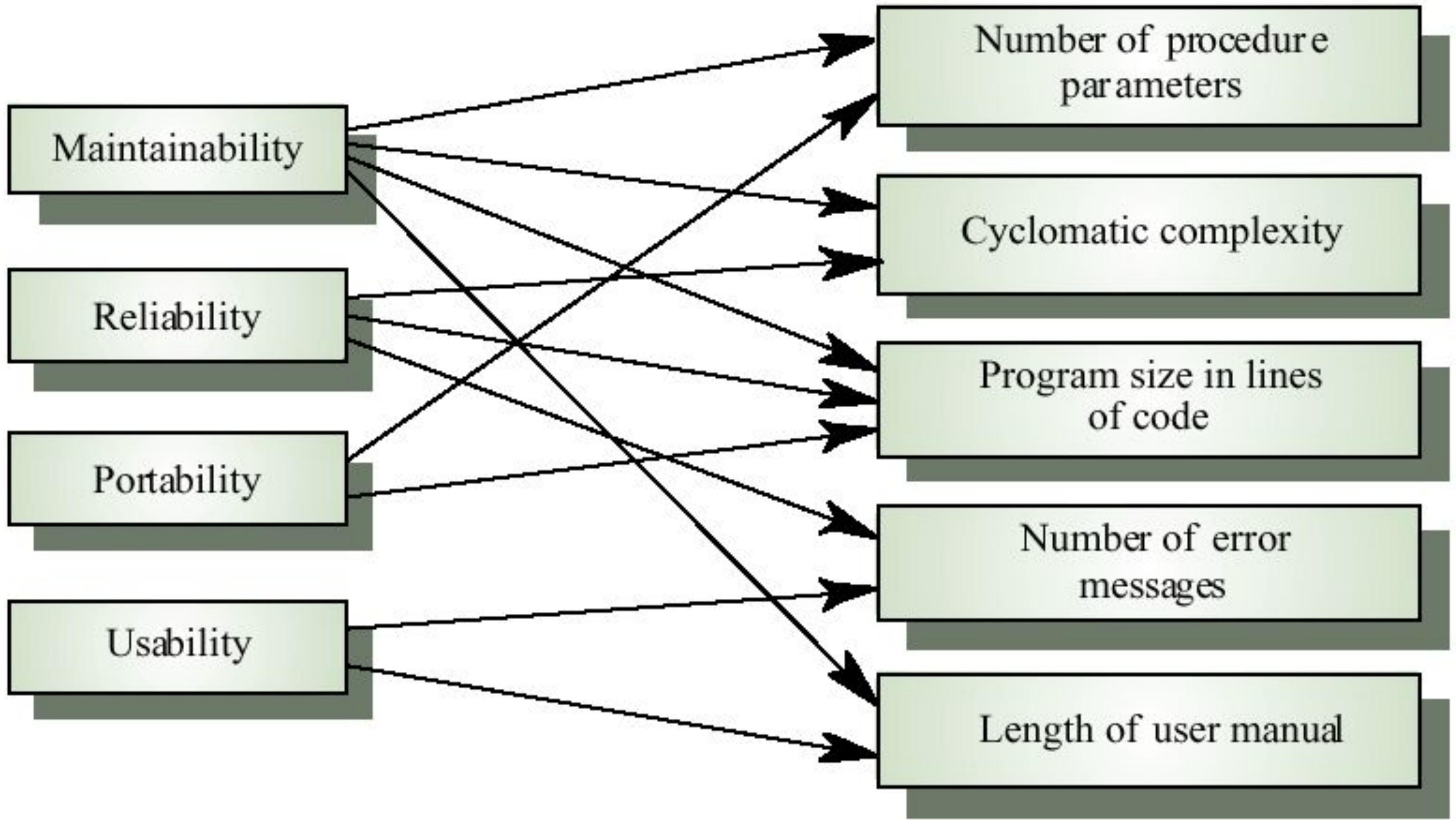
Predictor and control metrics



Metrics assumptions

- A software property can be measured
- The relationship exists between what we can measure and what we want to know
- This relationship has been formalized and validated
- It may be difficult to relate what can be measured to desirable quality attributes

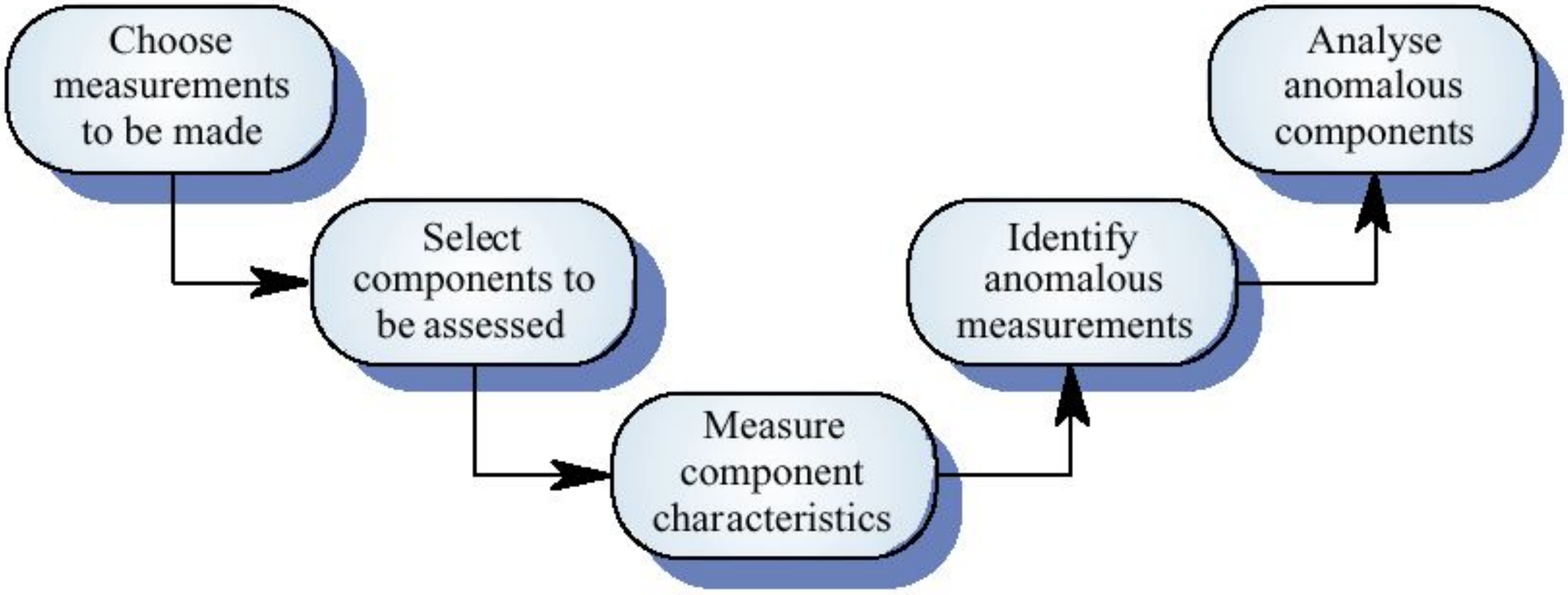
Internal and external attributes



The measurement process

- A software measurement process may be part of a quality control process
- Data collected during this process should be maintained as an organisational resource
- Once a measurement database has been established, comparisons across projects become possible

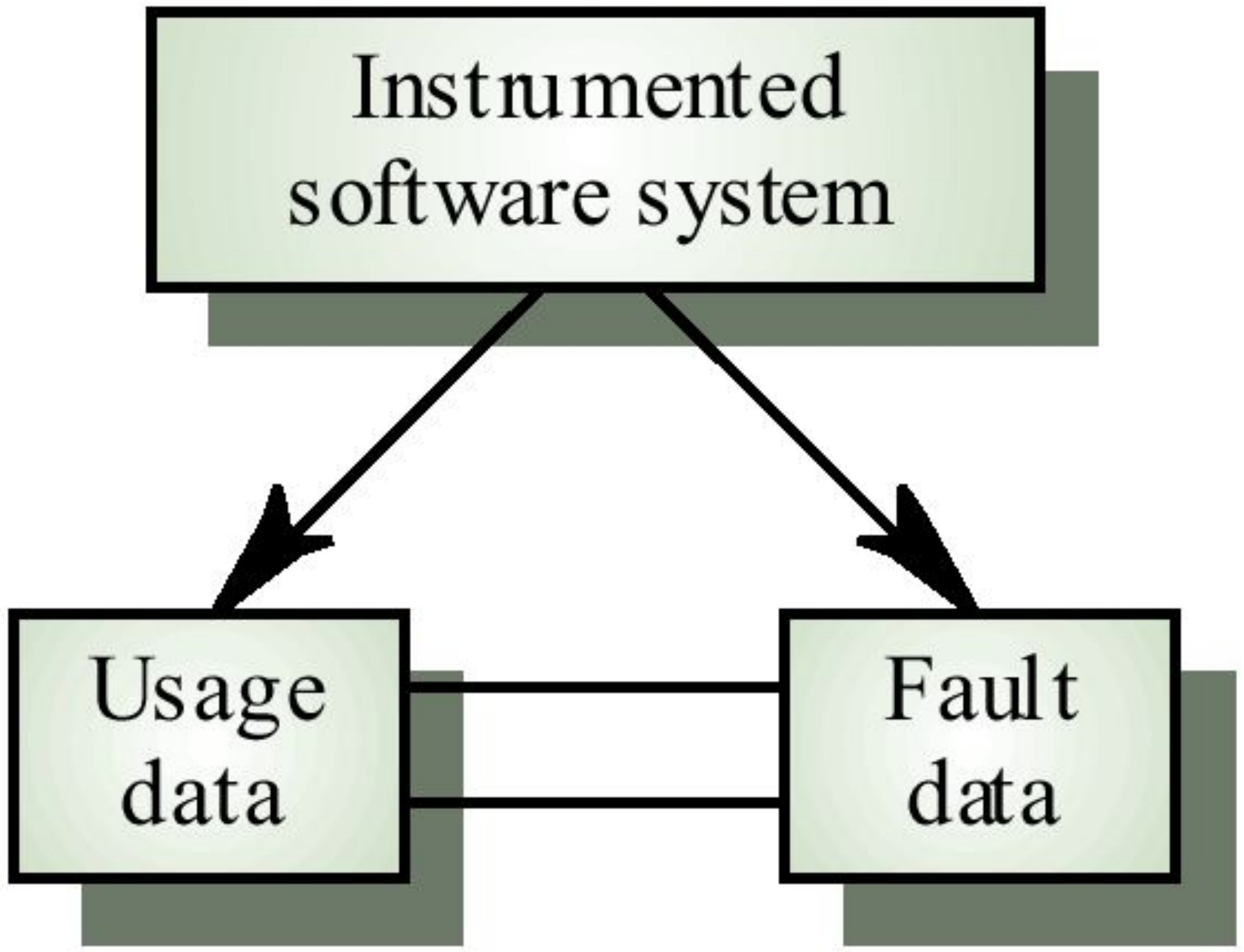
Product measurement process


Choose measurements to be made
Select components to be assessed
Measure component characteristics
Identify anomalous measurements
Analyse anomalous components

Data collection

- A metrics programme should be based on a set of product and process data
- Data should be collected immediately (not in retrospect) and, if possible, automatically
- Three types of automatic data collection
  - Static product analysis
  - Dynamic product analysis
  - Process data collation

Automated data collection



Data accuracy

- Don’t collect unnecessary data
  - The questions to be answered should be decided in advance and the required data identified
- Tell people why the data is being collected
  - It should not be part of personnel evaluation
- Don’t rely on memory
  - Collect data when it is generated not after a project has finished

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

Software product metrics

Software metric	Description
Fan in/Fan-out	Fan-in is a measure of the number of functions that call some other function (say X). Fan-out is the number of functions which 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 program's components, the more complex and error-prone that component is likely to be.
Cyclomatic complexity	This is a measure of the control complexity of a program. This control complexity may be related to program understandability. The computation of cyclomatic complexity is covered in Chapter 20.
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 may be 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 as, potentially, many different object classes 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 above 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 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	These are the number of operations in a super-class which are over-ridden 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.

Measurement analysis

- It is not always obvious what data means
  - Analysing collected data is very difficult
- Professional statisticians should be consulted if available
- Data analysis must take local circumstances into account

Measurement surprises

- Reducing the number of faults in a program leads to an increased number of help desk calls
  - The program is now thought of as more reliable and so has a wider more diverse market. The percentage of users who call the help desk may have decreased but the total may increase
  - A more reliable system is used in a different way from a system where users work around the faults. This leads to more help desk calls

Key points

- Software quality management is concerned with ensuring that software meets its required standards
- Quality assurance procedures should be documented in an organisational quality manual
- Software standards are an encapsulation of best practice
- Reviews are the most widely used approach for assessing software quality

Key points

- Software measurement gathers information about both the software process and the software product
- Product quality metrics should be used to identify potentially problematical components
- There are no standardised and universally applicable software metrics