Software Quality Assurance

What is Software Quality?

- Quality is an attribute of software that implies the software meets its specification

- The assessment of software quality is a subjective process where the quality management team has to use their judgment to decide if an acceptable level of quality has been achieved.

What is Software Quality?

- **The quality management team has to consider whether or not the software is fit for its intended purpose. This involves answering questions about the system's characteristics.**

1. Have programming and documentation standards been followed in the development process?
2. Has the software been properly tested?
3. Is the software sufficiently dependable to be put into use?
4. Is the performance of the software acceptable for normal use?
5. Is the software usable?
6. Is the software well structured and understandable?

Software Quality Attributes

Important software quality attributes are:

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

Software Quality Assurance

- To ensure quality in a software product, an organization must have a three-prong approach to quality management:
  - Organization-wide policies, procedures and standards must be established.
  - Project-specific policies, procedures and standards must be tailored from the organization-wide templates.
  - Quality must be controlled; that is, the organization must ensure that the appropriate procedures are followed for each project
- Standards exist to help an organization draft an appropriate software quality assurance plan.
  - ISO 9000-3
  - ANSI/IEEE standards
- External entities can be contracted to verify that an organization is standard-compliant.

	QA	QC	Testing
Purpose	Setting up adequate processes, introducing the standards of quality to prevent the errors and flaws in the product	Making sure that the product corresponds to the requirements and specs before it is released	Detecting and solving software errors and flaws
Focus	Processes	Product as a whole	Source code and design
What	Prevention	Verification	Detection
Who	The team including the stakeholders	The team	Test Engineers, Developers
When	Throughout the process	Before the release	At the testing stage or along with the development process

SQA Activities

- Applying technical methods
  - To help the analyst achieve a high quality specification and a high quality design
- Conducting formal technical reviews
  - A stylized meeting conducted by technical staff with the sole purpose of uncovering quality problems
- Testing Software
  - A series of test case design methods that help ensure effective error detection
- Enforcing standards
- Controlling change
  - Applied during software development and maintenance
- Measurement
  - Track software quality and asses the ability of methodological and procedural changes to improve software quality
- Record keeping and reporting
  - Provide procedures for the collection and dissemination of SQA information

Advantages of SQA

- Software will have fewer latent defects, resulting in reduced effort and time spent during testing and maintenance
- Higher reliability will result in greater customer satisfaction
- Maintenance costs can be reduced
- Overall life cycle cost of software is reduced

Disadvantages of SQA

- It is difficult to institute in small organizations, where available resources to perform necessary activities are not available
- It represents cultural change - and change is never easy
- It requires the expenditure of dollars that would not otherwise be explicitly budgeted to software engineering or QA

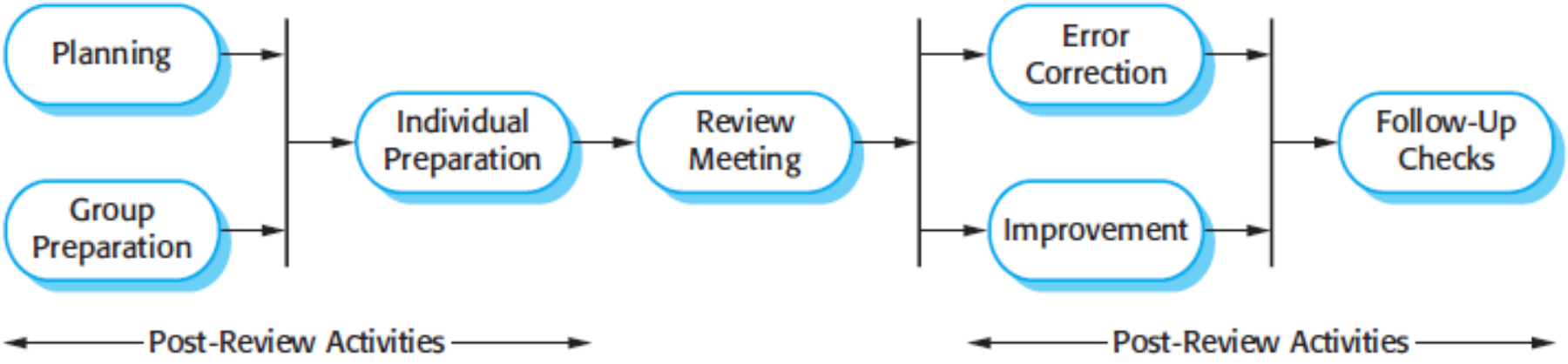
Reviews and inspections

- Reviews and inspections are QA activities that check the quality of project deliverables.
- 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.

The software review process



Program inspections

- These are peer reviews where engineers examine the source of a system with the aim of discovering anomalies and defects.
- Inspections do 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.

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.

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.

Fan-in/Fan-out, Length of code

- **Fan-in/Fan-out**
  - Fan-in is a measure of the number of functions or methods that call another 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.

- The complexity M is then defined as

  M = E – N + 2P,where

  - *E* = the number of edges of the graph.
  - *N* = the number of nodes of the graph.
  - *P* = the number of connected components.

CYCLOMATIC COMPLEXITY

- The complexity M is then defined as

M = R + 1,

where *R* = the number of regions in the graph.

- The complexity M is then defined as

M = P + 1,

where *P* = the number of predicate nodes in the graph.

These two formulas are easy to use.

SPECIALIZATION INDEX (SIX)

The metric provides a percentage, where the class contains at least one operation. For a root class, the specialization indicator is zero. Nominal range is between 0 % and 120 %.

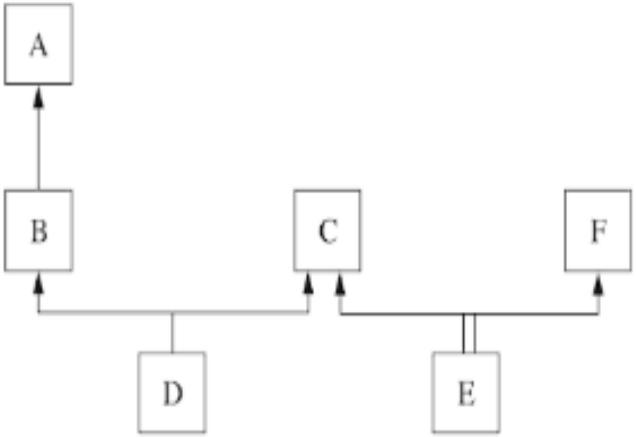
$$SIX = \frac{NMO \times DIT}{NMI + NMA + NMO} \times 100$$

The ... variable	represents the ...
DIT	depth of inheritance
NMA	the number of operations added to the inheritance
NMI	the number of inherited operations
NMO	the number of overloaded operations

NMO – Number of Overridden Methods not Overloaded.

Example: $$SIX = \frac{3 \times 4}{3 + 4 + 3} \times 100 = 120$$

How to calculate DIT :



DIT(D) = 2
DIT(E) = 1

Calculating SIX

```
Class Person{
 void read();
 void display();
}
Class Student extends Person{
 void red();
void display();
Void getAverage();
}
Class GraduateStudent extends Student{
 void red();
void display();
Void workStatus();
}
```

$$SIX = \frac{NMO \times DTO}{NMO + NMA + NMI}$$

DEFECT REMOVAL EFFICIENCY

- *A **defect** is found when the application does not conform to the requirement specification.*
- *A mistake in coding is called **Error***
- An average DRE score is usually around 85% across a full testing program.
- *DRE = E / (E + D) where:*
  - *E* is the number of errors found before delivery of the software to the end-user
  - *D* is the number of defects found after delivery.
- We found 100 defects during the testing phase and then later, say within 90 days after software release (in production), found five defects,

  DRE = 100/(100+5) = 95.2%

Fog index

- This is a measure of the average length of words and sentences in documents. The higher the value of a document's Fog index, the more difficult the document is to understand. The formula for the index is as follows:
- Fog index = ((average number of words per sentence) + (number of words of 3 syllables or more)) * 0.4
- The "ideal" score is 7 or 8; anything above 12 is too hard for most people to read.

- Average sentence length = 102/4 = 25 words
- Number of "big words" = 9
- Fog Index = (25 + 9) * 0.4 = * 0.4 = **13.6**

Key points

- Reviews of the software process deliverables involve a team of people who check that quality standards are being followed.
- 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
- Software measurement can be used to gather data about software and software processes.
- Product quality metrics are particularly useful for highlighting anomalous components that may have quality problems.