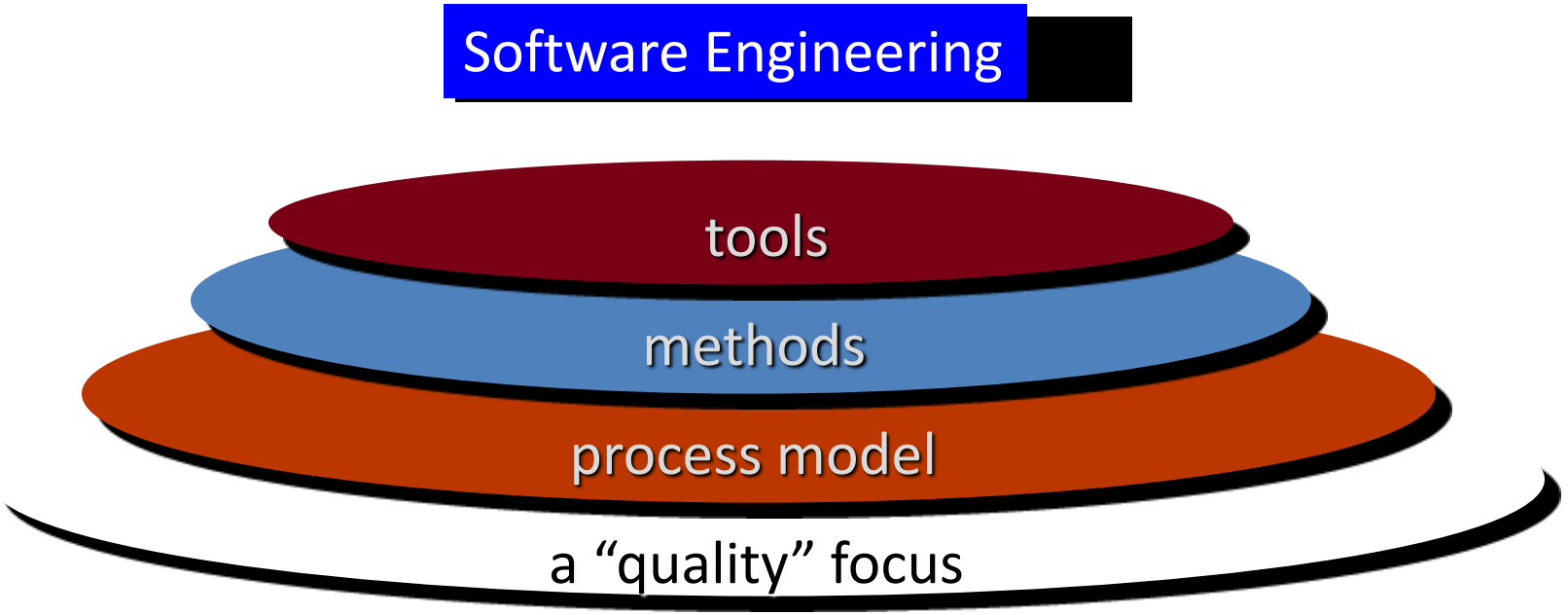
Software Quality Assurance

Introduction



Introduction

- **Quality** is defined as a characteristics or attributes of something where as attributes refer to measurable characteristics-things that we are able to compare to known standards.
- **There are two kinds of quality:**
  - **Quality of design** refers to the characteristics that designers specify for an item. The grade of materials, tolerances and performance specifications all contribute to the quality of design.
  - **Quality of conformance** is the degree to which the design specifications are followed during manufacturing. Greater the degree of conformance, the higher is the level of quality of conformance

Continue…

- ***Software quality*** *is defined as conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software.*

Continue…

- In software development, quality of design encompasses requirements, specifications, and the design of the system where as quality of conformance is an issue focused primarily on implementation.
- If the implementation follows the design and the resulting system meets its requirements and performance goals, conformance quality is high.

Software Quality Assurance(SQA)

- **Software quality assurance (SQA)** is an umbrella activity that is applied throughout the software process.
- **SQA encompasses:**
  - A quality management approach
  - Effective software engineering technology
  - Formal technical reviews
  - Multi-tier testing strategy
  - Control of software documentation and the changes made to it
  - A procedure to ensure compliance with software development standards
  - Measurement and reporting mechanism

SQA Activities

- SQA is composed of a variety of tasks associated with two different constituencies- the software engineer who do technical work and an SQA group that has responsibility for quality assurance planning, oversight , record keeping analysis and reporting.
- The charter of the SQA group is to assist software team in achieving a high quality end product.
- The SEI recommends a set of SQA activities that address quality assurance.

Activities…

- **Prepare an SQA plan for a project**
  - The plan is developed during project planning and is reviewed by all interested parties. SQA activities performed by the software engineering team and the SQA team group are governed by the plan. The plan identifies:
  - Evaluations to be performed
  - Audits and reviews to be performed
  - Standards that are applicable to the project
  - Procedures for error reporting and tracking
  - Documents to be produced by the SQA team
  - Amount of feedback provided to the software project team

Continue…

- **Participates in the development of the project's software process description**
  - The software team selects a process for the work to be performed
  - The SQA reviews the process description for compliance with organization policy, internal software standards, externally imposed standards and other parts of software project plan.
- **Reviews software engineering activities to verify compliances with defined software process**
  - The SQA group identifies, documents and track deviations from the process and verifies that corrections have been made.

Continue…

- **Audits designated software work products to verify compliance with those defined as part of the software process**
  - The SQA reviews selected work products, identifies, documents and track deviations; verifies that correction have been made; and periodically reports the results of its works to the project manager
- **Ensures that deviations in software work and work products are documented and handled according to a documented procedures**
- **Records any noncompliance and reports to senior management**

Quality Concepts

- 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.

Continue…

- Examples:
  - No two products are similar
  - All engineered and manufactured parts exhibit variation.
  - Variation control is the heart of quality control.

Software Reviews

- Software reviews are a filter for the software engineering process.
- Reviews are applied at various points during software development and serve to uncover errors and defects that can then be removed.
- Software reviews purify the software engineering activities that we have called analysis , design and coding.

Continue…

- A review is a way of using the diversity of a group of people to:
  - Point out needed improvements in the product of a single person or team;
  - Confirm those parts of a product in which improvement is either not desired or not needed;
  - Achieve technical work of more uniform, or at least more predictable, quality than can be achieved without reviews, in order to make technical work more manageable.

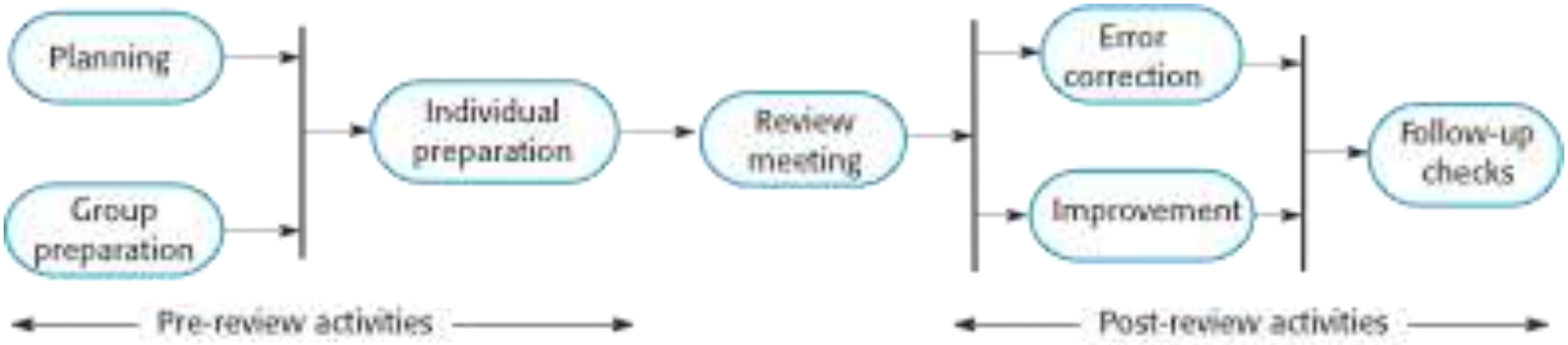
Software Reviews

- 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.

Software Reviews 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.

Inspection checklists

- Checklist of common errors should be used to drive the inspection.
- Error checklists are programming language dependent and reflect the characteristic errors that are likely to arise in the language.
- In general, the 'weaker' the type checking, the larger the checklist.
- Examples: Initialisation, Constant naming, loop termination, array bounds, etc.

An inspection checklist (a)

Fault class	Inspection check
Data faults	• Are all program variables initialized before their values are used? • Have all constants been named? • Should the upper bound of arrays be equal to the size of the array or Size -1? • If character strings are used, is a delimiter explicitly assigned? • Is there any possibility of buffer overflow?
Control faults	• For each conditional statement, is the condition correct? • Is each loop certain to terminate? • Are compound statements correctly bracketed? • In case statements, are all possible cases accounted for? • If a break is required after each case in case statements, has it been included?
Input/output faults	• Are all input variables used? • Are all output variables assigned a value before they are output? • Can unexpected inputs cause corruption?

An inspection checklist (b)

Fault class	Inspection check
Interface faults	• Do all function and method calls have the correct number of parameters? • Do formal and actual parameter types match? • Are the parameters in the right order? • If components access shared memory, do they have the same model of the shared memory structure?
Storage management faults	• If a linked structure is modified, have all links been correctly reassigned? • If dynamic storage is used, has space been allocated correctly? • Is space explicitly deallocated after it is no longer required?
Exception management faults	• Have all possible error conditions been taken into account?

Software Reliability

- ✧ **Reliability** is a measurable system attribute so non-functional reliability requirements may be specified quantitatively. These define the number of failures that are acceptable during normal use of the system or the time in which the system must be available.
- **Functional reliability** requirements define system and software functions that avoid, detect or tolerate faults in the software and so ensure that these faults do not lead to system failure.
- **Software reliability** requirements may also be included to cope with hardware failure or operator error.

Software Reliability

- **Reliability**
  - The probability of failure-free system operation over a specified time in a given environment for a given purpose
- **Availability**
  - The probability that a system, at a point in time, will be operational and able to deliver the requested services
- Both of these attributes can be expressed quantitatively e.g. availability of 0.999 means that the system is up and running for 99.9% of the time.

Reliability Specification Process

- **Risk identification**
  - Identify the types of system failure that may lead to economic losses.
- **Risk analysis**
  - Estimate the costs and consequences of the different types of software failure.
- **Risk decomposition**
  - Identify the root causes of system failure.
- **Risk reduction**
  - Generate reliability specifications, including quantitative requirements defining the acceptable levels of failure.

Types of system failure

Failure type	Description
Loss of service	The system is unavailable and cannot deliver its services to users. You may separate this into loss of critical services and loss of non-critical services, where the consequences of a failure in non-critical services are less than the consequences of critical service failure.
Incorrect service delivery	The system does not deliver a service correctly to users. Again, this may be specified in terms of minor and major errors or errors in the delivery of critical and non-critical services.
System/data corruption	The failure of the system causes damage to the system itself or its data. This will usually but not necessarily be in conjunction with other types of failures.

Reliability Metrics

- Reliability metrics are units of measurement of system reliability.
- System reliability is measured by counting the number of operational failures and, where appropriate, relating these to the demands made on the system and the time that the system has been operational.
- A long-term measurement programme is required to assess the reliability of critical systems.
- Metrics
  - Probability of failure on demand
  - Rate of occurrence of failures/Mean time to failure
  - Availability

Examples: Probability of failure on demand (POFOD)

- This is the probability that the system will fail when a service request is made. Useful when demands for service are intermittent and relatively infrequent.
- Appropriate for protection systems where services are demanded occasionally and where there are serious consequence if the service is not delivered.
- Relevant for many safety-critical systems with exception management components
  - Emergency shutdown system in a chemical plant.

Rate of fault occurrence (ROCOF)

- Reflects the rate of occurrence of failure in the system.
- ROCOF of 0.002 means 2 failures are likely in each 1000 operational time units e.g. 2 failures per 1000 hours of operation.
- Relevant for systems where the system has to process a large number of similar requests in a short time
  - Credit card processing system, airline booking system.
- Reciprocal of ROCOF is Mean time to Failure (MTTF)
  - Relevant for systems with long transactions i.e. where system processing takes a long time (e.g. CAD systems). MTTF should be longer than expected transaction length.

Perceptions of reliability

- The formal definition of reliability does not always reflect the user's perception of a system's reliability
  - The assumptions that are made about the environment where a system will be used may be incorrect
    - Usage of a system in an office environment is likely to be quite different from usage of the same system in a university environment
  - The consequences of system failures affects the perception of reliability
    - Unreliable windscreen wipers in a car may be irrelevant in a dry climate
    - Failures that have serious consequences (such as an engine breakdown in a car) are given greater weight by users than failures that are inconvenient

Reliability and specifications

- Reliability can only be defined formally with respect to a system specification i.e. a failure is a deviation from a specification.
- However, many specifications are incomplete or incorrect – hence, a system that conforms to its specification may 'fail' from the perspective of system users.
- Furthermore, users don't read specifications so don't know how the system is supposed to behave.
- Therefore perceived reliability is more important in practice.