Injibara University

College of Engineering and Technology

Department of Software Engineering

Subject: Software Engineering

Course Code:CoSc3061

Molla K. (MSc)

Software Quality Assurance

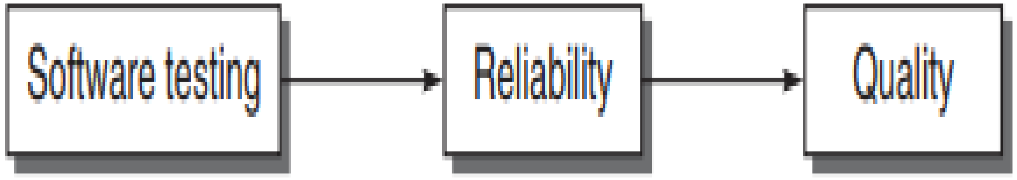
Chapter-6
Software Quality Assurance

An overview of Testing

- ***Reliability*** is a measure of success with which the observed behavior of a system conforms to the specification of its behavior.
- ***Software Reliability*** is the probability that a software system will not cause the failure of the system for a specified time under specified conditions.
- Testing is the process of analyzing a system or system component to detect the differences between specified (required) and observed (existing) behavior.

Cont'd

- ✓ ***Software Testing*** is a process used to identify the correctness, completeness and quality of developed computer software.
- ✓ It is also the process of finding differences between the expected behavior specified by system models and the observed behavior of the implemented system.
- ✓ It is the process of executing a program/ application under **positive** and **negative** conditions by manual or automated means. It checks for the :-
  - ➢ ***Specification***
  - ➢ ***Functionality***
  - ➢ ***Performance***



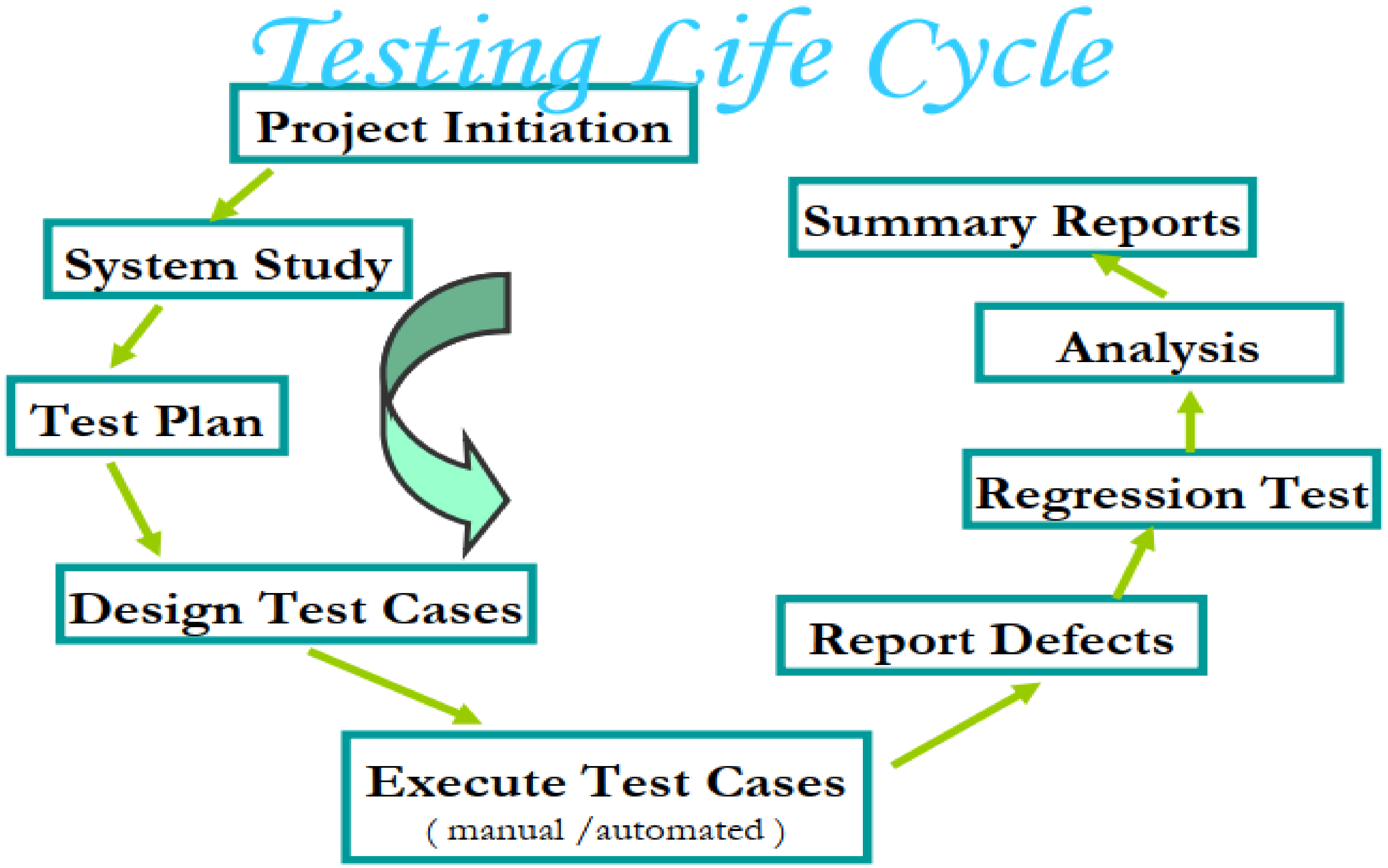
Importance Of Software Testing in SDLC

✓ *When to start testing in SDLC?*

- ➢ Requirement
- ➢ Analysis design
- ➢ Coding
- ➢ Testing
- ➢ Implementation
- ➢ Maintenance

✓ Testing starts from Requirement Phase.

Cont'd


Testing Life Cycle
Project Initiation
System Study
Test Plan
Design Test Cases
Execute Test Cases
(manual / automated)
Report Defects
Regression Test
Analysis
Summary Reports

Test Planning

Test Plan

- ✓ A test plan is a systematic approach to testing a system i.e. software.
- ✓ The plan typically contains a detailed understanding of what the eventual testing workflow will be.

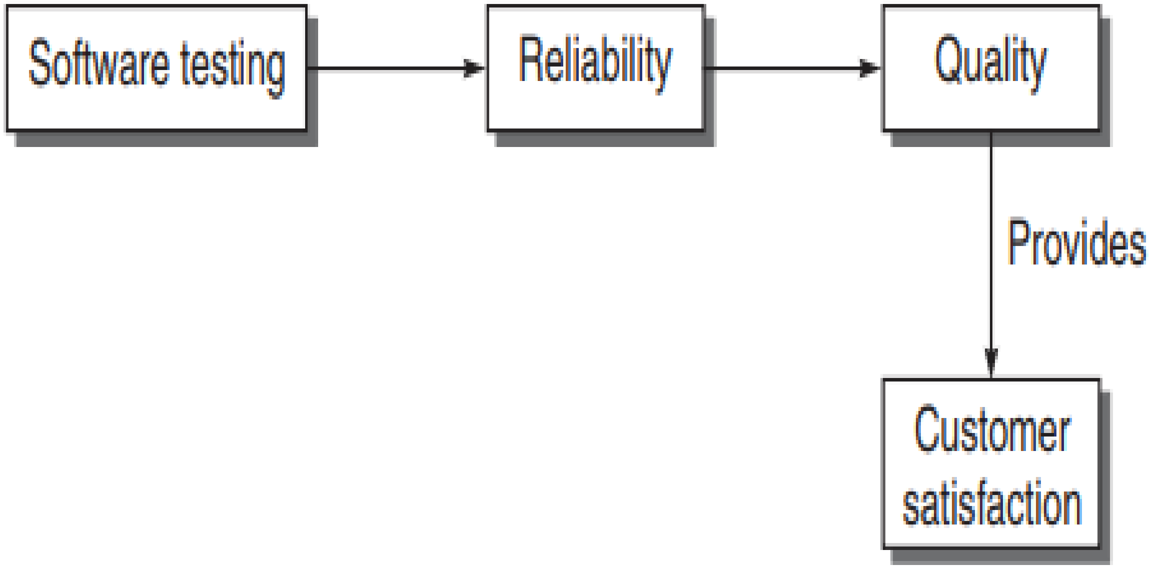
Test Case

A test case is a specific procedure of testing a particular requirement. It will include:

- ✓Identification of specific requirement tested
- ✓Test case success/failure criteria
- ✓Specific steps to execute test
- ✓Test Data

Quality Assurance and Quality Control

- ✓ *Quality* is defined as meeting the ***Customer's Requirements*** and according to the standards.
- ✓ The ***quality*** of software is the extent to which it meets its specifications.
- ✓ The best measure of Quality is given by ***FURPS***
  - ➢ Functionality
  - ➢ Usability
  - ➢ Reliability
  - ➢ Performance
  - ➢ Scalability



Why Quality?

- Quality is the important factor affecting an organization's long term performance.
- Quality improves productivity and competitiveness in any organization.
- A quality product is defined in terms of its fitness of purpose robustness.
- That is, a quality product does exactly what the users want it to do. Some software product has several quality factors such as the following:
  - ✓**Portability**
  - ✓**Usability**
  - ✓**Reusability**
  - ✓**Correctness**

Quality Assurance

- Quality Assurance is a planned and systematic set of activities necessary to provide adequate confidence that products and services will conform to specified requirements and meets user needs.
  - ✓It is process oriented.
  - ✓Defect prevention based.
  - ✓Throughout the Life Cycle.
  - ✓It's a management process.

Quality Control

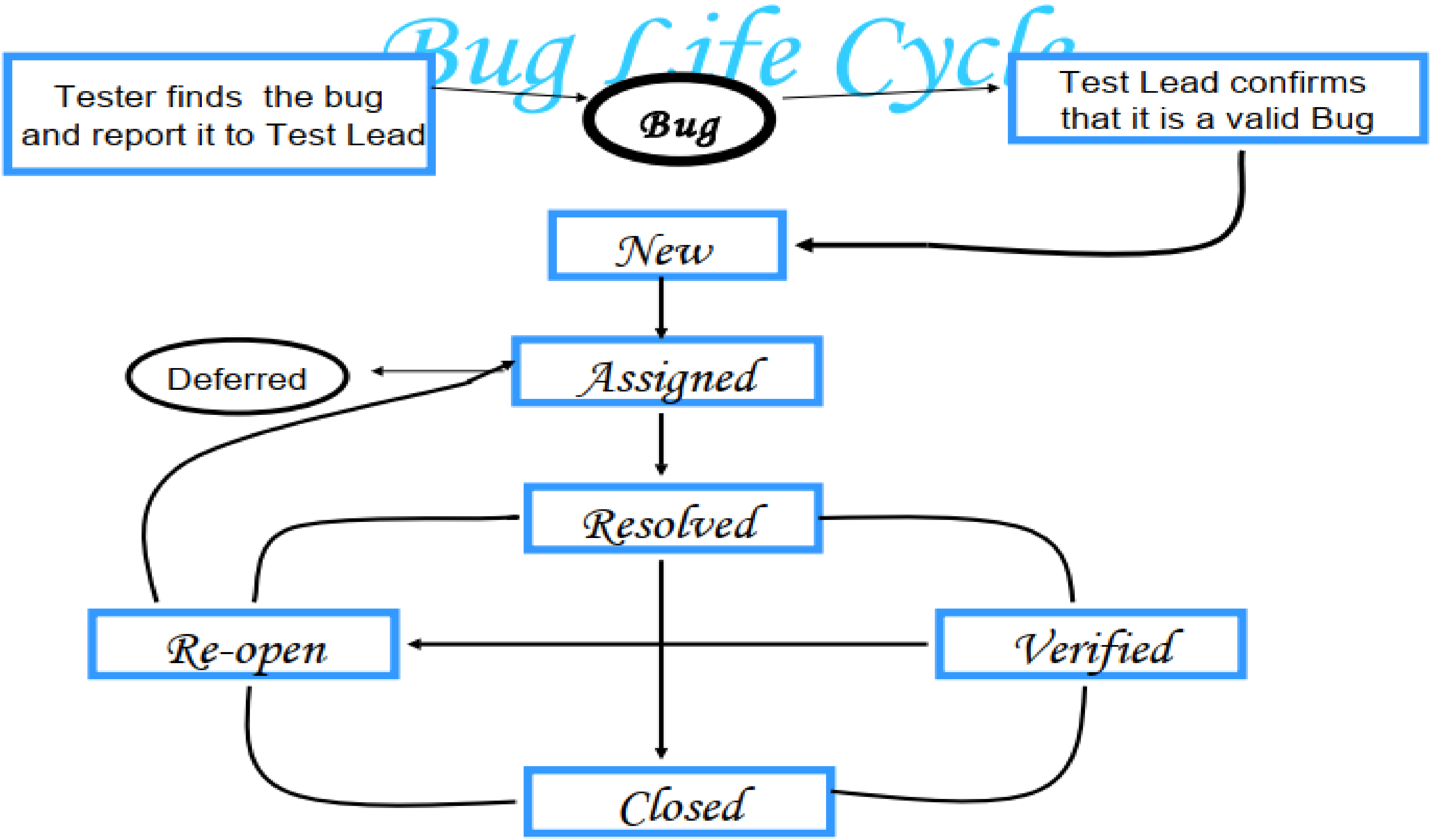
- Quality control is the process by which product quality is compared with the applicable standards and the action taken when non conformance is detected.
    - ✓ It is product oriented.
    - ✓ Defect detection based

Cont’d

QA vs. QC

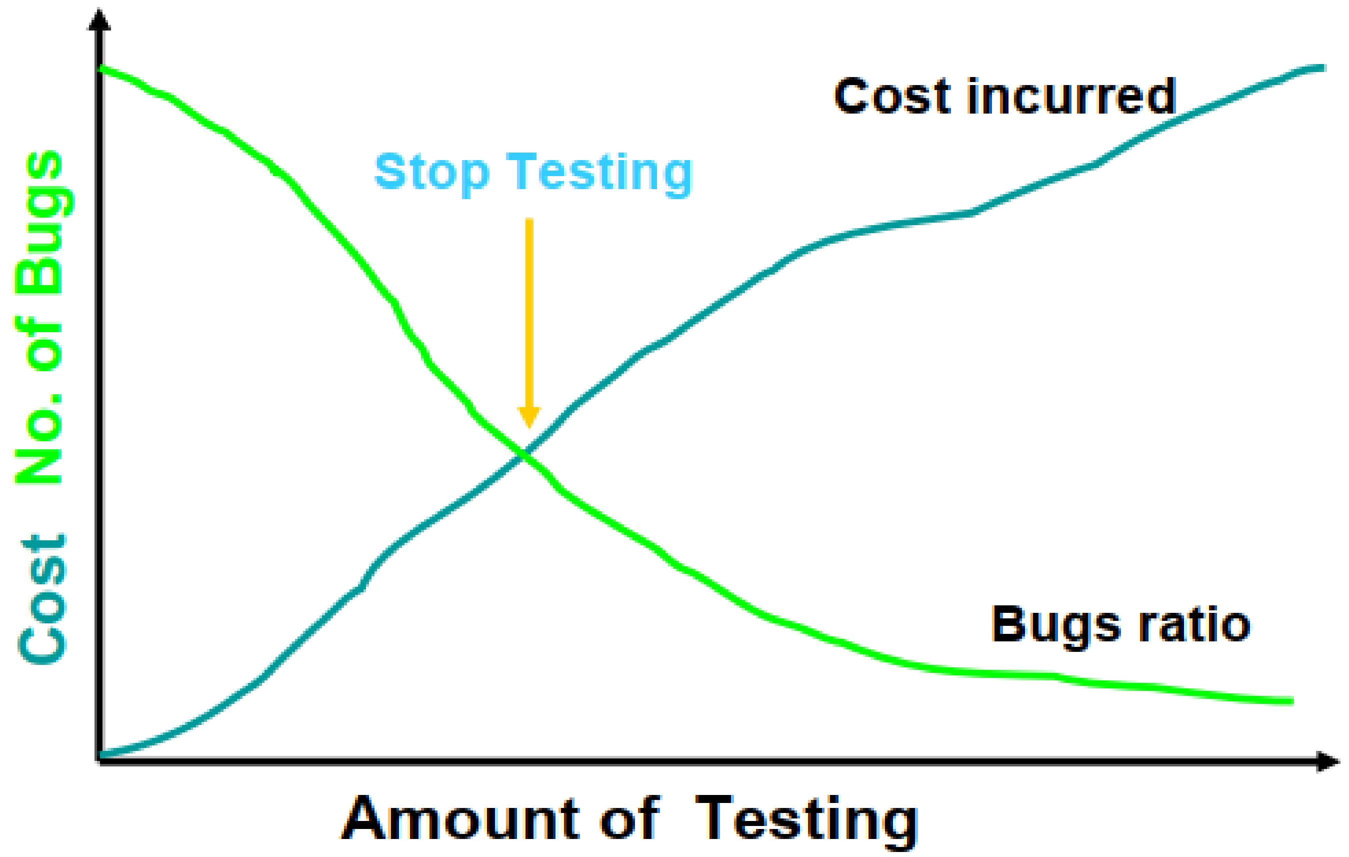
• Quality Assurance makes sure that we are doing the right things, the right Way.	• Quality Control makes sure the results of what we’ve done are what we expected .
• QA focuses on building in quality and hence preventing defects.	• QC focuses on testing for quality and hence detecting defects.
• QA deals with process.	• QC deals with product.
• QA is for entire life cycle.	• QC is for testing part in SDLC.
• QA is preventive process.	• QC is corrective process.

Bug Life Cycle


Bug Life Cycle
Tester finds the bug and report it to Test Lead
Bug
Test Lead confirms that it is a valid Bug
New
Assigned
Deferred
Resolved
Re-open
Verified
Closed

Cont'd

When to Stop Testing



Quality Control Techniques

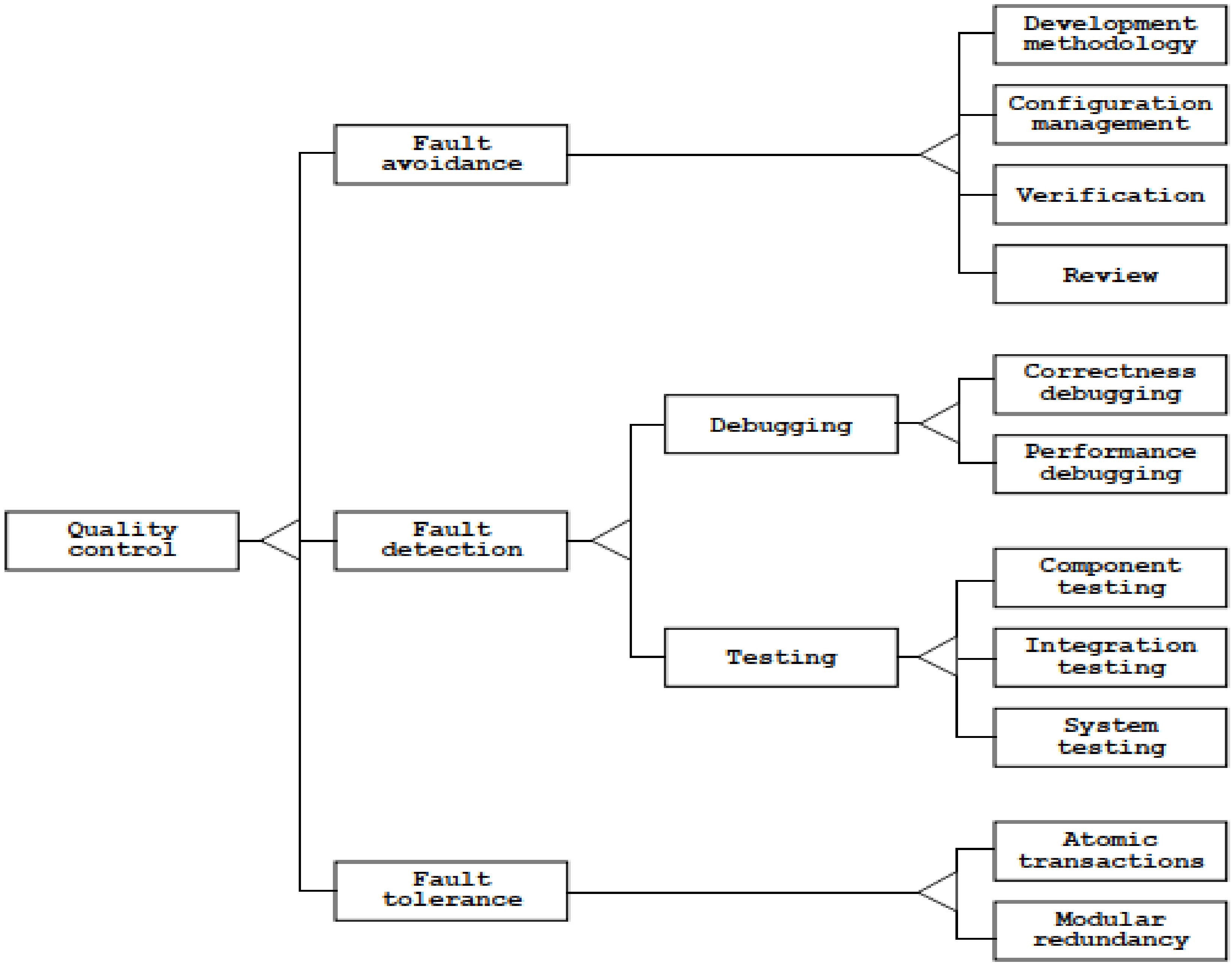
- There are many **techniques** for increasing the reliability of a software system.

√**Fault Avoidance Techniques**

√ **Fault Detection Techniques**

√**Fault Tolerance Techniques**

Cont'd


Quality control
Fault avoidance
Fault detection
Fault tolerance
Development methodology
Configuration management
Verification
Review
Debugging
Testing
Correctness debugging
Performance debugging
Component testing
Integration testing
System testing
Atomic transactions
Modular redundancy

Fault Avoidance Techniques

- It tries to prevent the occurrence of errors and failures by finding **faults** in the system before it is released. It include:

- *Development methodologies, configuration management, verification techniques and reviews.*
- **Development Methodologies-It** avoid faults by providing techniques that minimize fault introduction in the system models and code.
- **Configuration Management-** avoids faults caused by undisciplined change in the system models .
- **Verification-**attempts to find faults before any execution of the system
- A **review** is the manual inspection of parts or all aspects of the system without actually executing the system.

Fault Detection Techniques

- Fault detection techniques assist in **finding faults in systems** but do not try to recover from the failures caused by them.
- In general, they are applied during development, but in some cases they are also used after the release of the system. There are two types of fault detection techniques, **Debugging** and **Testing**.
- Debugging assumes that faults can be found by starting from an unplanned failure. There are two types of debugging:
- The goal of correctness debugging is to find any deviation between the observed and specified functional requirements.
- Performance debugging addresses the deviation between observed and specified nonfunctional requirements, such as response time.

Cont'd

- ***Testing*** is a fault detection technique that tries to create failures or errors in a planned way.
- This allows the developer to detect failures in the system before it is released to the customer.
- ***Unit Testing*** tries to find faults in participating objects and/or subsystems with respect to the use cases from the use case model
- ***Integration Testing*** is the activity of finding faults when testing the individually tested components together
- ***System Testing*** tests all the components together, seen as a single system to identify errors with respect to the scenarios from the problem statement and the requirements and design goals identified in the analysis and system design, respectively:

Cont’d

- Functional testing tests the requirements from the RAD and, if available, from the user manual.
- **Performance testing** checks the nonfunctional requirements and additional design goals from the SDD.
- Note that functional and performance testing are both done by developers.
- **Acceptance testing** and installation testing check the requirements against the project agreement and should be done by the client, if necessary with support by the developers.

Fault Tolerance Techniques

- If we cannot prevent errors we must accept the fact that the released system contains faults that can lead to failures.
- Fault tolerance is the recovery from failure while the system is executing.
- It allows a system to recover from a failure of a component by passing the erroneous state information back to the calling components.
- Database systems provide **atomic transactions** to recover from failure during a sequence of actions that need to be executed together or not at all.

Cont'd

- **Modular Redundancy** is based on the assumption that system failures are usually based on component failures.
- Modular redundant systems are built by assigning more than one component to perform the same operation.
- The system can continue even if one component is failing, because the other components are still performing the required functionality.

Testing Concepts

- **A component** is a part of the system that can be isolated for testing. A component can be an object, a group of objects, or one or more subsystems.
- **A fault,** also called bug or defect, is a design or coding mistake that may cause abnormal component behavior.
- **An error** is a manifestation of a fault during the execution of the system.
- **A failure** is a deviation between the specification of a component and its behavior. A failure is triggered by one or more errors.
- **A Test case** is a set of inputs and expected results that exercises a component with the purpose of causing failures and detecting faults.

Cont'd

- ***A Test Stub*** is a partial implementation of components on which the tested component depends.
- ***A Test Driver*** is a partial implementation of a component that depends on the tested component.

✓ Test stubs and drivers enable components to be isolated from the rest of the system for testing.

- ***A Correction*** is a change to a component.

✓ The purpose of a correction is to repair a fault.

Testing Activities

- Testing activities include:
- **Inspecting Components**, which finds faults in an individual component through the manual inspection of its source code
- **Unit Testing**, which finds faults by isolating an individual component using test stubs and drivers and by exercising the component using a test case
- **Integration Testing**, which finds faults by integrating several components together
- **System Testing**, which focuses on the complete system, its functional and nonfunctional requirements, and its target environment.

Managing Testing

- In this section, we describe how to manage testing activities to minimize the resources need.
- Many testing activities occur near the end of the project, when resources are running low and when the delivery pressure increases.
- Often, trade-offs have to be weighed between the faults that need to be repaired before delivery and those that can be repaired in a subsequent revision of the system.
- In the end, however, developers should detect and repair a sufficient number of faults such that the system meets functional and nonfunctional requirements to an extent acceptable to the client.

Planning Testing

- Developers can reduce the cost of testing and the elapsed time necessary for its completion through careful planning. Two key elements are to start the selection of test cases early and to parallelize tests.
- Developers responsible for testing can design test cases as soon as the models they validate become stable.
- Functional tests can be developed when the use cases are completed.
- Unit tests of subsystems can be developed when their interfaces is defined.
- Similarly, test stubs and drivers can be developed when component interfaces are stable.
- Developing tests early enables the execution of tests to start as soon as components become available.

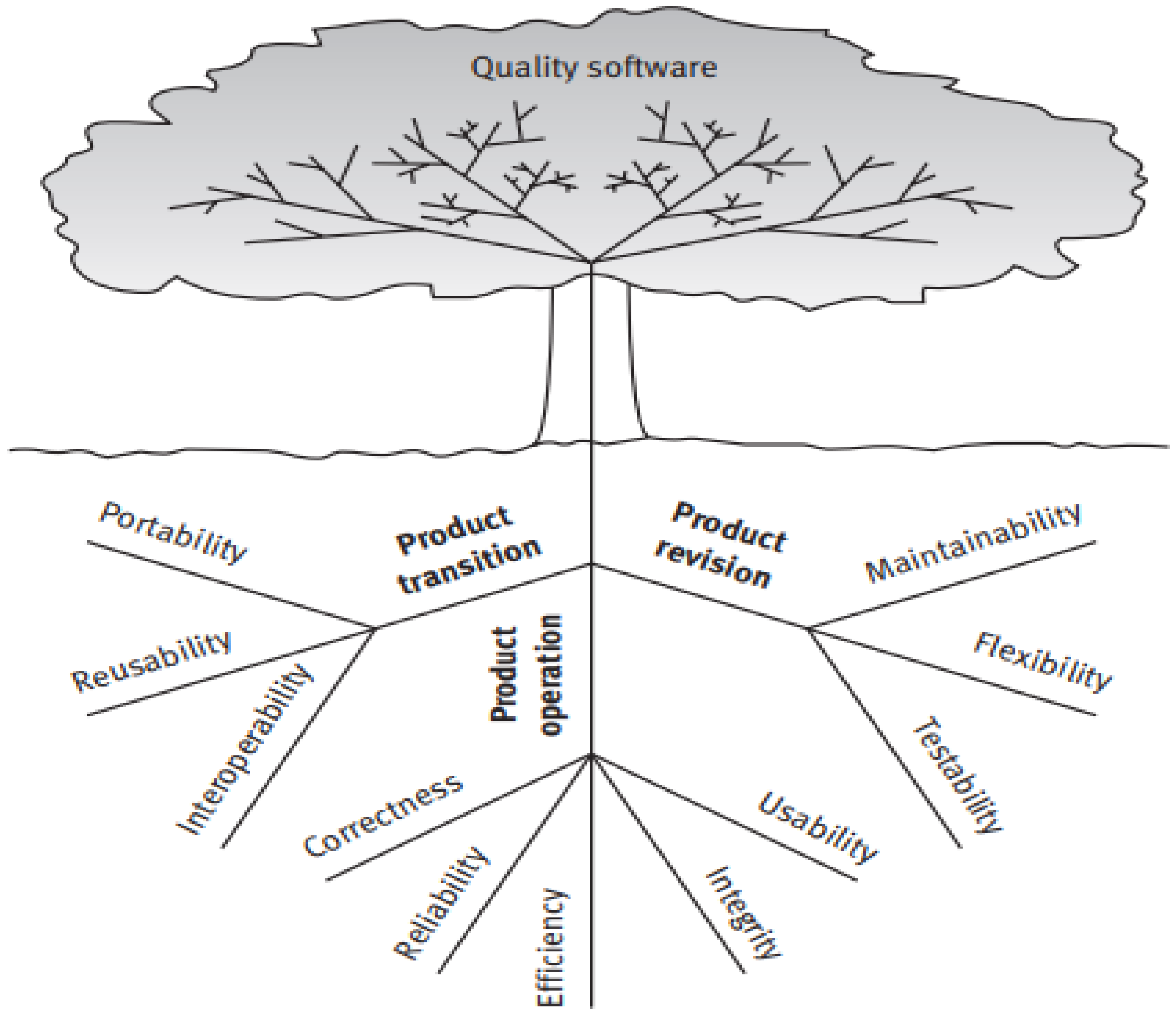
Documenting Testing

- Testing activities are documented in four types of documents, the Test Plan, the Test Case Specifications, the Test Incident Reports, and the Test Summary Report
- **Test Plan:** The Test Plan focuses on the managerial aspects of testing.
- It documents the *scope*, *approach*, *resources*, and *schedule* of testing activities.
- The *requirements* and the components to be tested are identified in this document.
- **Test Case Specification:** Each test is documented by a test case specification.
- This document contains the inputs, drivers, stubs, and expected outputs of the tests.
- This document also contains the tasks to be performed.

Cont’d

- **Test Incident Report:** Each execution of each test is documented by a Test Incident Report.
- The actual results of the tests and differences from the expected output are recorded.
- **Test Summary Report:** This document lists all the failures that were discovered during the tests and that need to be investigated.
- From the Test Report Summary, the developers analyze and prioritize each failure and plan for changes in the system and in the models.
- These changes in turn can trigger new test cases and new test executions.

McCall's Factor Model Tree



The End of this Course!

2014 E.C