Testing Overview (cont.)

Course Code: CSC4133 Course Title: Software Quality and Testing

Dept. of Computer Science
Faculty of Science and Technology

Lecturer No:	**9**	**Week No:**	**5**	**Semester:**	
Lecturer:	*Prof. Dr. kamruddin Nur,* *kamruddin@aiub.edu*				

Lecture Outline

- Testing vs. Debugging
- Testing Related Questions
- Major Testing Techniques
  - Black-box testing
  - White-box testing
- When to stop testing?
  - Resource-based criteria
  - Quality-based criteria

Objectives and Outcomes

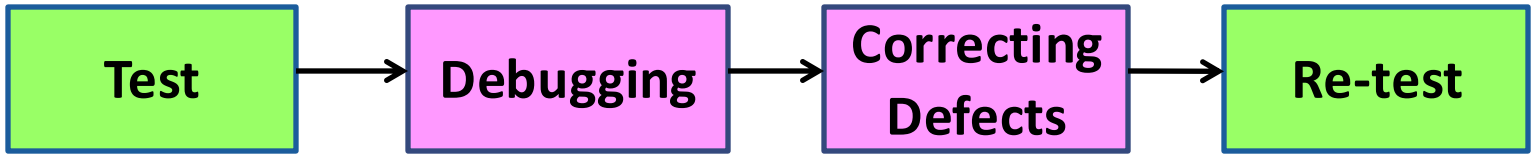
- **Objectives**: To understand the difference between testing and debugging, to understand the key considerations in testing, to understand the difference between white-box testing and black-box testing, to understand the criteria when to stop testing in a software development project.

- **Outcomes**: Students are expected to be able to explain how testing is different from debugging, be able to distinguish between white-box and black-box testing, be able to explain the stopping criteria of testing in a software development project.

Testing vs. Debugging

- Debugging and testing are different. Dynamic testing can show failures that are caused by defects. Debugging is the development activity that finds, analyses and removes the cause of the failure.
- Test and re-test are test activities
  - Testing shows system failures
  - Re-testing proves, that the defect has been corrected
- Debugging and correcting defects are developer activities
  - Through debugging, developers can reproduce failures, investigate the state of programs and find the corresponding defect in order to correct it.

Testing vs. Debugging



- Test and re-test are ***test activities***
- Test & Re-test ==> done by **Tester** (Test Engineer)

- **Debugging and correcting defects are *developer activities***
- Debugging & correcting defect/bug ==> done by **programmer**

Roles & Responsibilities: **Developer** and **Tester**

Developer Role

– Implements requirements
– Develops structures
– Designs & programs the software
–Creating a product is his success

Tester Role

– Plans testing activities
– Design test cases & executes test cases
– Is concerned *only with* finding defects
– Finding an error made by a developer is his success

Perception:

Developers are constructive! Testers are destructive!

Wrong!

Testing is a constructive activity as well,
It aims at eliminating defects from a product!

Testing & QA alternatives

- Defect & QA:
  - Defect: ==> error/fault/failure
  - Defect prevention/removal/containment
  - Map to major QA activities
- Defect prevention:
  - Error blocking & error source removal
- Defect removal:
  - Testing, Inspection, Walkthrough etc.
- Defect containment:
  - Fault tolerance & failure containment (safety assurance)

QA and Testing

- Testing as part of QA:
  - Activities focus on testing phase
  - QA/testing in Waterfall and V-models
  - One of the most important parts of QA
  - Defect removal

Testing: Key Questions

- **WHY**: Quality demonstration *vs*. defect detection & removal

- **HOW**: Techniques/activities/process/etc.
- **VIEW**: Functional/external/black-box

  vs.

  Structural/internal/white-box

- **EXIT**: Coverage ***vs.*** Usage-based

Testing: Why?

- The purpose of software testing is to ensure that the software systems would work as expected when they are used by their target customers and users.
- Most natural way to show fulfillment of expectations is to demonstrate their operation through some “dry-runs” or controlled experimentation in laboratory settings before the products are released/delivered.
  - Such controlled experimentation through program execution is generally called testing

Testing: Why?

- Original/primary purpose: Demonstration of proper behavior or quality demonstration
  - "Testing" in traditional settings
  - Provide evidence of quality in the context of QA
- New purpose: Defect detection & removal
  - Mostly defect-free software development vs. traditional development
  - Flexibility of software (ease of change)
  - Failure observation ==> fault removal
  - defect detection ==> defect fixing
  - Eclipsing original purpose

Testing: Why?

- Summary: Testing fulfills two primary purposes:
  1) To demonstrate quality or proper behavior
  2) To detect and fix problems (defects/bugs)

Testing: How

- **How? Run** ==> **Observe** ==> **Follow-up**

  (particularly in case of failure observations)

- Refinement ==> **generic testing process**

- Generic testing process as instantiation of SQE process

Generic Testing Process

- Major test activities include the following in roughly chronological order:
  1) Test planning and preparation
  2) Test Execution
  3) Analysis and follow-up

Activities in Generic Testing Process

- Major testing activities : The Generic Testing Process
  1) Test planning and preparation ➔ Sets the goals for testing, select an overall testing strategy, and prepare specific test cases and the general test procedure
  2) Test Execution ➔ Include related observation & measurement of product behavior
  3) Analysis and follow-up ➔ Include result checking and analysis to determine if a failure has been observed, and if so, follow-up activities are initiated & monitored to ensure removal of the underlying causes/faults, that led to the observed failures in the first place.

1) Test Planning and Preparation

- **Test planning:**
  - Goal setting based on customer's quality perspectives & expectations
  - Overall strategy based on the above and product/environment characteristics.
- **Test preparation:**
  - Preparing test cases & test suites
  - Prepare test procedure

2) Test Execution

- General steps in test execution:
  - Allocating test time (& resources)
  - Invoking test
  - Identifying system failures (& gathering information for follow-up actions)
- Key to execution: Handling both normal vs. abnormal cases
- Activities closely related to execution:
  - Failure identification
  - Data capturing & other measurement

3) Test Analysis and Follow-up

- Analysis of testing results:
  - Result checking (as part of execution)
  - Further result analyses
    - Defect/reliability/ etc. analyses
- Follow-up activities:
  - Feedback based on analysis results
  - Immediate: Defect removal (& re-test)
  - Other follow-up(longer term):
    - Decision making (exit testing, etc.)
    - Test process improvement

Testing: How?

- How to test?
  - Refine into three sets of questions
    i. Basic questions
    ii. Testing technique questions
    iii. Activity/management questions
- Basic questions
  - What artifacts are tested?
  - What to test?
    - From which view?
    - Related: type of faults found
  - When to stop testing?

Functional *vs.* Structural Testing

- **Key distinction**: Perspective on what need to be checked/tested.
- **Functional testing:**
  - Tests external functions
  - As described by external specs.
  - **Black-box** in nature
  - Functional mapping: Input ==> Output
  - Without involving internal knowledge

Functional *vs.* Structural Testing

- **Structural testing:**
  - Tests internal implementations
  - Components and structures
  - **White-box** in nature
  - "white" here ==> seeing through ==> internal elements visible
  - Really clear/glass/transparent box

Black-Box *vs.* White-Box View

- Object abstraction/representation:
  - High-level: Whole system ==> black-box
  - Low –level: Individual statements, data, and other elements ==> white-box
  - Middle-levels of abstraction ==> Gray-box
    - Functional/subroutine/procedure, module , subsystem etc.
    - Method, class, super-class
- Gray-box (mixed black-box & white-box) testing:
  - Many of the middle levels of testing
  - Example: procedures in modules
    - Procedures individually as black box,
    - Procedure interconnection ➔ white box at module level

White-Box Testing

- Program component/structure knowledge (or implementation details)
  - Statement/component checklist
  - Path (control flow) testing
  - Data (flow) dependency testing
- **Applicability**
  - Test in the small/early
  - Dual role of programmers/testers
- Criterion for stopping
  - Mostly coverage goals
  - Occasionally quality/reliability goals

Black-Box Testing

- Input/output relations or external functional behavior
  - Specification checklist
  - Testing expected/specified behavior
- **Applicability**
  - Late in testing: system testing etc.
  - Suitable for **IV&V**
- Criteria: when to stop
  - Traditional: functional coverage
  - Usage-based: reliability target

Comparing BBT with WBT

- Perspective:
  - **BBT** views the objects of testing as a black-box while focusing on testing the input-output relations or external functional behavior
  - **WBT** views the objects as a glass-box where internal implementation details are visible & tested

Comparing BBT with WBT

- Objects:
  - **WBT** is generally used to test **small objects** (e.g., **small** software products or small units of large software products)
  - **BBT** is generally more suitable for **large** software systems or substantial parts of them as a whole
- Timeline:
  - **WBT** is used more in early sub-phases (e.g., unit and component testing)
  - **BBT** is used more in the late sub-phases (e.g., system and acceptance testing)

Comparing BBT with WBT

- Defect Focus:
  - In **BBT**, failures related to specific external functions can be observed, leading to corresponding faults being detected & removed. *Emphasis* ==> Reduce chances of encountering functional problems by target customers.
  - In **WBT**, failures related to internal implementations can be observed, leading to corresponding faults being detected & removed directly. *Emphasis* ==> Reduce internal faults so that there is less chance for failures later on.

Comparing BBT with WBT

- Defect detection & Fixing:
  - Defects detected through **WBT** are easier to fix than those through BBT
  - **WBT** may miss certain types of defects (e.g., omission & design problems) which could be detected by **BBT**
  - *In general*: **BBT** is effective in detecting & fixing problems of interfaces & interactions, *while* **WBT** is effective for problems localized within a small unit.

Comparing BBT with WBT

- Techniques:
  - A specific technique is **BBT** if external functions are modeled
  - A specific technique is **WBT** if internal implementations are modeled

Comparing BBT with WBT

- Tester:
  - **BBT** is typically performed by dedicated professional **testers**, and could also be performed by third-party personnel in a setting of **IV&V**
  - **WBT** is often performed by **developers** (programmers) themselves

When to Stop Testing?

- **Exit Criteria**
  - Not finding(any more) defects is **NOT** an appropriate criteria to stop testing activities
- **Why?!**?

- When to stop testing in a software development project?

When to Stop Testing?

- **Resource-based criteria**: A decision is made based on resource consumptions.
  - Stop when you run out of time
  - Stop when you run out of money
  - Such criteria are irresponsible, as far as product quality is concerned
- **Quality-based criteria:**
  - Stop when quality goals reached
    - Direct quality measure: reliability
      - Resemble actual customer usages
    - Indirect quality measure: coverage
  - **Other surrogate:** Activity completion ("stop when you complete planned test activities")

Books

- *Software Quality Engineering: Testing, Quality Assurance and Quantifiable Improvement*, by Jeff Tian

References

1. *Software Testing and Quality Assurance: Theory and Practice*, by Kshirasagar Naik, Priyadarshi Tripathy
2. *Software Quality Assurance: From Theory to Implementation*, by Daniel Galin
3. *Software Testing and Continuous Quality Improvement*, by William E. Lewis
4. *The Art of Software Testing*, by Glenford J. Myers, Corey Sandler and Tom Badgett
5. *Software Testing Fundamentals: Methods and Metrics* by Marnie L. Hutcheson