COURSE NAME

SOFTWARE QUALITY AND TESTING

CSC 4133

(UNDERGRADUATE)

CHAPTER 12

SYSTEM TESTING

Dr. Mohammad Mahmudul Hasan
Associate Professor
Department of Computer Science
American international university – Bangladesh
Email: m.hasan@aiub.edu
Web: http://cs.aiub.edu/profile/m.hasan

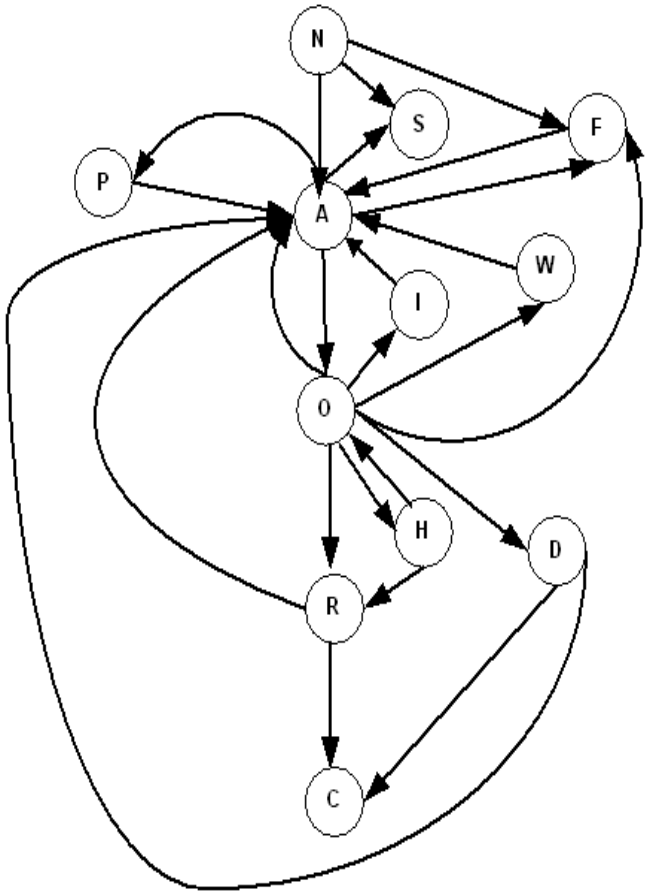
SYSTEM TESTING

- is the testing of a complete and fully integrated software product
- tests the overall system operations as a whole, typically from a customer's perspective
- should test functional and non-functional requirements of the software
- is black-box testing and often done by independent testers
- Preparing for executing system-level tests are a critical phase in a s/w development process:
  - pressure to meet a tight schedule close to delivery date
  - a need to discover most of the defects before delivering the product
  - essential to verify that defect fixes are working and have not resulted in new defects

MODELING DEFECTS

- A defect life-cycle model in the form of a state-transition diagram is shown in the next figure
- The different states are briefly explained in the preceding Table
- Two key concepts involved in modeling defects:
  - the levels of priority
  - the levels of severity
- Priority and Severity assignments are separately done
- Priority of a bug is how important/soon it is for a bug to be fixed.
  - Priority is the order in which developer has to fix the bug.
  - Defect priority levels: Critical (1), High (2), Medium (3), Low(4)
- Severity of a bug is a measure of how bad(impact) the bug is.
  - Defect severity levels: Critical (1), High (2), Medium (3), Low (4)

MODELING DEFECTS



State	Semantic	Description
N	NEW	A problem report with a severity level and a priority level is filed.
A	ASSIGNED	The problem is assigned to an appropriate person.
O	OPEN	The assigned person is actively working on the problem to resolve it.
R	RESOLVED	The assigned person has resolved the problem, and is waiting for the submitter to verify and close it.
C	CLOSED	The submitter has verified the resolution of the problem.
W	WAIT	The assigned person is waiting for additional information from the submitter.
F	FAD	The reported defect is not a true defect. Rather, it is a Function As Designed.
H	HOLD	The problem is on hold because this problem cannot be resolved until another problem is resolved.
S	SHELVED	There is a problem in the system, but a conscious decision is taken that this will not be resolved in the near future. Only the Development Manager can move a defect to this state.
D	DUPLICATE	The problem reported is a duplicate of a problem that has already been reported.
I	IRREPRODUCIBLE	The problem reported cannot be reproduced by the assigned person.
P	POSTPONED	The problem will be resolved in a later release.

METRICS FOR MONITORING TEST EXECUTION

- Test Case Escapes (TCE)
  - A significant increase in the number of test case escapes implies that deficiencies in the test design
- Planned vs. Actual Execution (PAE) Rate
  - Compare the actual number of test cases executed every week with the planned number of test cases to be executed
  - Represents the productivity of the test team
- Execution Status of Test (EST) Cases
  - Periodically monitor the number of test cases lying in different states – Failed, Passed, Invalid and Untested
  - Useful to further subdivide those numbers by test categories

DEFECT ANALYSIS TECHNIQUES

- Defect Causal Analysis (DCA)
  - Causal analysis is conducted to identify the root cause of the defects and to take actions to eliminate the sources of defects; this is done at the time of fixing defects.
- Orthogonal Defect Classification (ODC)
  - In the ODC method, assessment is not done against individual defects; rather, trends & patterns in the aggregate data are studied.
  - ODC along with the application of Pareto analysis can give a good indication of which parts of the system are error prone and may require more testing.
- Pareto Principle
  - The principle is to state that 80% of the problems can be fixed with 20% of the effort

BETA TESTING

- Conducted by the potential buyers prior to the official release of the product
- Purpose is not to find defects, but to obtain feedback from the field about the usability of product
- A decision about when to release the system to the beta customers is made by the software project team members. The beta release criteria are established by the project team.
- The system test cycle continues concurrently with beta testing
- Weekly meetings are conducted with the beta customers by the beta support team to resolve issues
- Categories of Beta Testing:
  - Marketing beta: builds early awareness and interest in the product among potential buyers.
  - Technical beta: obtains feedback about the usability of the product in a real environment with different configurations from a small number of friendly customers.
  - Acceptance beta: conducted to ensure that the product meets its acceptance criteria

FIRST CUSTOMER SHIPMENT

- All the test cases from the test suite should have been executed
- Test case results are updated with Passed, Failed, Blocked, or Invalid status
- The pass rate of test cases is very high, say, 98%
- No crash in the past two weeks of testing has been observed
- No known defect with CRITICAL or HIGH severity exists in the product
- Not more than a certain number of known defects with MEDIUM and LOW levels of severity exist
- All the resolved defects must be in the CLOSED state
- The user guides are in place
- Trouble shooting guide is available
- The test report is completed and approved

PRODUCT SUSTAINING

- Once the product is shipped to one of the paying customers, the software project is moved to sustaining phase
- The goal of this phase is to maintain the software quality throughout the product's market life
- Software maintenance activities occur because software testing cannot uncover all the defects in a large software system
- Different types of maintenance
  - Corrective maintenance [about 20%]: Modification to fix a problem
  - Enhancement [about 80%]
    - Perfective maintenance (modification to improve usability,…) [about 60%]
    - Adaptive maintenance (modification to keep up-to-date) [about 20%]
    - Preventive maintenance (modification to avoid any future error) [about 20%]

MEASURING TEST EFFECTIVENESS

- A common measure of test effectiveness is the number of defects found by the customers that were not found by the test engineers prior to the release of the product.
- The objective of a test effectiveness metric is to evaluate the effectiveness of a system testing effort in the development of a product in terms of the number of defects escaped from the system testing effort.
- Metrics to measure test effectiveness:
  - Defect removal efficiency (DRE)
  - Defect Age
  - Spoilage

DEFECT REMOVAL EFFICIENCY (DRE)

- DRE is the ratio of the number of defects discovered in an activity to the number of defects that should have been found.

$$\text{DRE} = \frac{\text{Number of Defects Found in Testing}}{\text{Number of Defects Found in Testing} + \text{Number of Detects Not Found}}$$

- One way to approximate this number is to count the defects found by the customers within the first six months of its operation.
- Defects submitted by customers that need corrective maintenance are taken into consideration in this measurement.
- Fault seeding is a process of intentionally adding known faults in a computer program for the purpose of monitoring the rate of detection and removal of faults and estimating the number of faults remaining in the program.

DEFECT AGE & DEFECT SPOILAGE METRIC

Defect Age

- Defect Age (PhAge) is a metric that can be used to measure the test effectiveness which assigns a numerical value to the fault depending on the phase in which it is discovered.
- For example, a requirement defect discovered during a high-level design review would be assigned a PhAge of 1. If this defect had not been found until system testing, it would have been assigned a PhAge of 6.

Defect Spoilage Metric

- Defect spoilage is a metric that uses Phase Age and distribution of defects to measure the effectiveness of defect removal activities.
- A lower value of spoilage indicates a more effective defect discovery process, the optimal value is 1

$$\text{Spoliage} = \frac{\sum (\text{Number of Defects} \times \text{Discovered PhAge})}{\text{Total Number of Defects}}$$

TESTING CRITERIA

- Boot tests: verifies that the system can boot up its software image from the supported boot options
- Upgrade/downgrade tests verifies that the system software can be upgraded or downgraded (rollback) in a graceful manner from the previous version to the current version or vice versa
- Diagnostic tests: verifies that the hardware components of the system are functioning as desired
- Command Line Interface (CLI) tests verifies that the system can be configured in a specific way by using the command line interface (Debian Linux environment)
- Functionality tests: verify the system as thoroughly as possible over the full range of requirements specified in the SRS document. Provide comprehensive testing over the full range of the requirements within the capabilities of the system

INTER-OPERABILITY (COMPATIBILITIES) TEST

- Inter-operability tests determine whether the system can inter-operate with other third-party products
- Compatibility tests verify that the system works the same way across different platforms, operating systems, database management systems
- Backward compatibility tests verify that the current software build flawlessly works with older version of platforms

SCALABILITY TESTS

- Scalability tests determine the scaling limits of the system
- Tests are designed to verify that the system can scale up to its engineering limits
- Scaling tests are conducted to ensure that the system response time remains the same, or increases as the number of users are increased (911 emergency line after 9/11, VUES registration, Exam result publish)
- Extrapolation is often used to predict the limit of scalability

RELIABILITY TESTS

- Reliability tests measure the ability of the system to keep operating for a long time without developing failures.
- Mean Time Between Failures (MTBF) = MTTF + MTTR
  - Average of all time (Mean) Time To Failure (MTTF)
  - Average of all time (Mean) Time To Repair (MTTR)

ROBUSTNESS TESTS

- Robustness tests determine how well the system recovers from various input errors and other failure situations.
- Robustness means how much sensitive a system is to erroneous input and changes its operational environment
- Tests in this category are designed to verify how gracefully the system behaves in error situations and in a changed operational environment

DOCUMENT TESTS

- Documentation tests ensure that the system's user guides are accurate and usable
- Documentation test means verifying the technical accuracy and readability of the user manuals, tutorials and the on-line help

REGULATORY TESTS

- Regulatory tests ensure that the system meets the requirements of government regulatory bodies.
- The final system is shipped to the regulatory bodies in those countries where the product is expected to be marketed. The idea is to obtain compliance marks on the product from various countries.
- The regulatory agencies are interested in identifying flaws in software that have potential safety consequences. The safety requirements are primarily based on their own published standards.
- Most of these regulatory bodies issue safety, emissions/delivery, and immunity/protection compliance certificates.

SECURITY TESTING

- It checks to see if the application is vulnerable to attacks, if anyone hack the system or login to the application without any authorization.
- Software security aimed at:
  - Preventing unauthorized access to the system or parts of it
  - Detection of unauthorized access & the activities performed by the penetration
  - Recovery of damages caused by unauthorized penetration cases
- Examples of security flaws in an application:
  - A Student Management System is insecure if 'Admission' branch can edit the data of 'Exam' branch
  - An online Shopping Mall has no security if customer's Credit Card Detail is not encrypted

SAFETY TESTING

- Software safety is defined in terms of hazards. A hazard is a state of a system or a physical situation which combined with certain environmental conditions, could lead to an accident.
- An accident is an unintended event or series of events that results in death, injury, illness, damage or loss of property, or harm to the environment. The concept of safety is concerned with preventing hazards.
- A software in isolation cannot do physical damage. However, a software in the context of a system and an embedding environment could be vulnerable.
- Example:
  - A software module in a database application is not hazardous by itself, but when it is embedded in a missile navigation system, it could be hazardous.
  - If a missile takes a U-turn because of a software error in the navigation system, and destroys the submarine that launched it, then it is not a safe software.

PERFORMANCE TESTING

- The main focus is checking a software program's speed and response time.
  - speed (e.g., database response times)
  - throughput (transactions per second)
  - capacity (concurrent usage loads)
  - timing (hard real-time demands)
- Performance requirements should also address how the system's performance will degrade in an overloaded situation, such as when a 911 emergency telephone system is flooded with calls.
  - *PE-1. Every Web page shall download in 15 seconds or less over a 50 KBps modem connection.*
  - *PE-2. Authorization of an ATM withdrawal request shall not take more than 10 seconds.*
- Includes: Load testing, Stress testing, Volume testing

LOAD TESTING

- Load testing is defined as a short-term test of performance under real world conditions.
- Load testing is performed to find out whether the system can handle the expected load upon deployment under real-world conditions.
- The objective is to identify performance bottlenecks before the software application goes live.

STRESS TESTING

- Stress testing is conducted to evaluate & determine the behavior of a software component when the offered load is in excess of its designed capacity.
- Stress testing is performed to find the application's breaking point.
- Stress tests put a system under stress in order to determine the limitations of a system and, when it fails, to determine the manner in which the failure occurs.

VOLUME TESTING

- Volume testing is performed to find the stability of the system with respect to handling large amounts of data over extended time periods.
- Under volume testing, large amounts of data is populated in database and the overall software system's behavior is monitored
- The objective is to check application's performance under varying database volumes

REFERENCES

- Software Testing And Quality Assurance – Theory and Practice - Kshirasagar Naik & Priyadarshi Tripathy
- Software Quality Engineering: Testing, Quality Assurance and Quantifiable Improvement - Jeff Tian