Noakhali Science and Technology University

Institute of Information Technology

Software Engineering

Course Title: Software Testing and Quality Assurance

Course Code: SE3209

Assignment on: Introduction to Software Testing and Software Testing Terminology & Methodology

Submitted to
Tasniya Ahmed
Lecturer of IIT, NSTU

Submitted by
MD Mynuddin
Roll: ASH1825007M
1st Batch
IIT, NSTU

- **Software Testing.**
  Software testing is a process that detects important bugs with the objective of having better quality software.
- **Goals of Software Testing.**
  The goals of software testing classified into three major categories:
  - I. Immediate Goals
    - Bug Discovery
    - Bug Prevention
  - II. Long-term Goals
    - Reliability
    - Quality
    - Customer Satisfaction
    - Risk Management
  - III. Post-implementation Goals
    - Reduced Maintenance Cost
    - Improved Testing Process
- Explaining These Goals
  - **I. Immediate Goals:** After performing a test, it gives an immediate result. It sets in every phase of the SDLC. Such as
    - **a. Bug Discovery:** The Immediate Goals of testing is to find errors at any stage of the software development. Discovering more bugs at the early stage of software development, better will be success rate in software testing.
    - **b. Bug Prevention:** Coding safely such that bugs discovered shouldn't be repeated in later stages or future projects. From the behaviors and interpretations of bugs discovered, everyone in software development team should learn that. It's true that, errors cannot be prevented to zero, its can be minimized. For this reason, bug prevention is superior of testing.
  - **II. Long-term Goals:** When one cycle of the SDLC is over, these goals affect the product quality for the long time. Such as
    - **a. Quality:** Thorough testing ensures the superior quality of software products. The first goal of understanding and performing the testing process is to enhance the quality of software products. The software should be passed through rigorous reliability analysis to gain high quality standards. Reliability is the level of confidence increases with rigorous testing. The confidence and reliability increase the quality of software products.

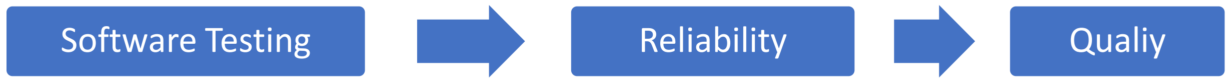


Figure: Testing produces reliability and quality

b. **Customer Satisfaction:** The prime concern of testing is customer satisfaction only from the perspective of users. For customer satisfaction, testing should be complete and thorough. Testing should be complete in the sense that it must satisfy the user for all the specified requirements mentioned in the user manual, as well as for the unspecified requirements which are otherwise understood. A complete testing process achieves reliability, reliability enhances quality, and quality increases customer satisfaction.
Software testing Reliability Quality Customer Satisfaction Provides Figure 1.4 Quality leads to customer satisfaction

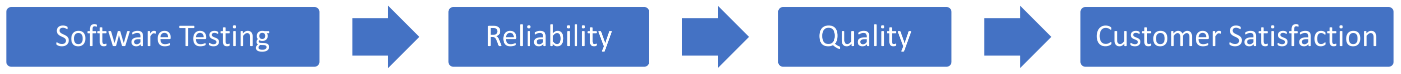


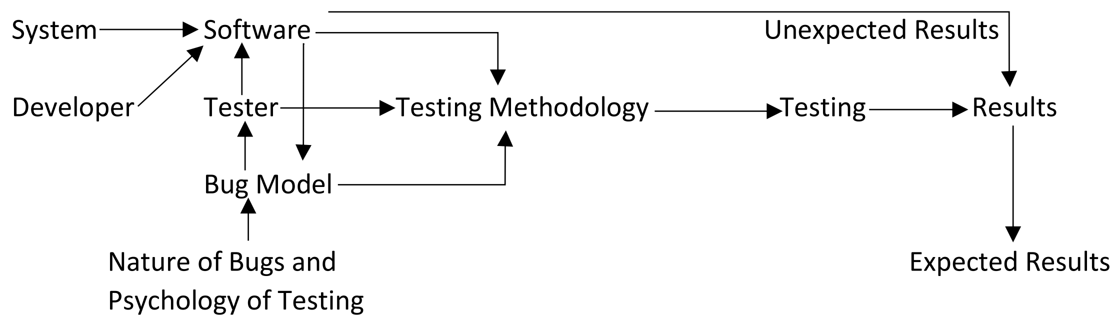
Figure: Quality leads to customer satisfaction

c. **Risk Management:** Risk is the probability that undesirable events will occur in a system. These undesirable events will prevent the organization from successfully implementing its business initiatives. Thus, risk is basically concerned with the business perspective of an organization. Risk Management is only discussed in spiral model. Software testing business risks are controlled by some risk factors such as Cost, Time, Resources and Critical features. Testing may indicate that the software being developed cannot be delivered on time, or there is a probability that high priority bugs will not be resolved by the specified time.

III. **Post-implementation goals:** After releasing a product, these goals are important. Such as

a. **Reduced Maintenance:** The only maintenance cost in a software product is its failure due to errors. Post-release errors are costlier to fix, as they are difficult to detect. Chances of failure can be minimized by testing the product rigorously and effectively as well as maintenance cost can be reduced.
b. **Improved Software Testing Process:** A testing process for one project may not be successful. Testing process should be scope for improvement. Therefore, the bug history and post-implementation results can be analyzed to find out faults in the present testing process, which can be rectified in future projects. Thus, the long-term post-implementation goal is to improve the testing process for future projects.

- **Draw the model of software testing.**



Developer should make a software model such that the software can be easily tested. Generally, developers do not make any sense of software model in testing purpose. So that, for testers it will be a complex software.

- **Define difference between failure and fault.**

Fault is caused by an error/bug/defect/mistake. Fault is synonymous with the words defect or bug. An uninitialized pointer variable in a C language program is an example of software faults. When a fault is executed, then failures are generated. When the software is tested, failure is the first term being used. Failures are manifestation of bugs. **fault ->error ->failure**

- **Test Case on Hospital Management System(HMS)**

Test Case ID	Test Scenario	Test Steps	Test Data	Expected Results	Actual Results	Pass/Fail
T-01	Verify that the portal for new patient registration has all the mandatory fields required for registering a patient.	1. Go to site www.thehospitals.com 2. Click on the Registration button		Should have all mandatory fields for registration	As Expected	Pass
T-02	Verify that after filling the patient details and successful payment a Patient-Card is printed.	1. Click on Patient Information from the menu 2. Fill the required fields 3. Click on the submit button 4. Pay fees for the Patient-Card 5. Print the Patient-Card		Should fill all the required information. Should pay required fees. Should print the Patient-Card	As Expected	Pass

T-03	Verify that card has information like patient details, doctor assigned, department, the application number, DOJ, bed allocated (if applicable) etc.	1. Click on the Patient-Card from the menu	Patient Details: Assigned Doctor: Firoz Ahmed Department: Cardiogram Application No: E09C1 DOJ: January 25, 2021 Bed Allocated: No	Should have these information	As Expected	Pass
T-04	Verify that after patient checkup based on the requirement the details are updated in the patient details database.	1. Click on the Patient Details Database from the menu 2. Enter Checkup date 3. Enter Application Number 3. Click on the Submit Button 4. Update the patient details. 5. Click on the submit button.		Should update patient information after checkup	As Expected	Pass
T-05	Verify that for existing patients based on the application number of the patient, their records are added/updated in the database.	1. Click on the Patient Details Database from the menu 2. Enter date of the checkup for patients. 3. Click on a patient's Record Updates button 4. Update Patients Records 5. Click on the submit button		Should update patient records according to their application number on the database	As Expected	Pass

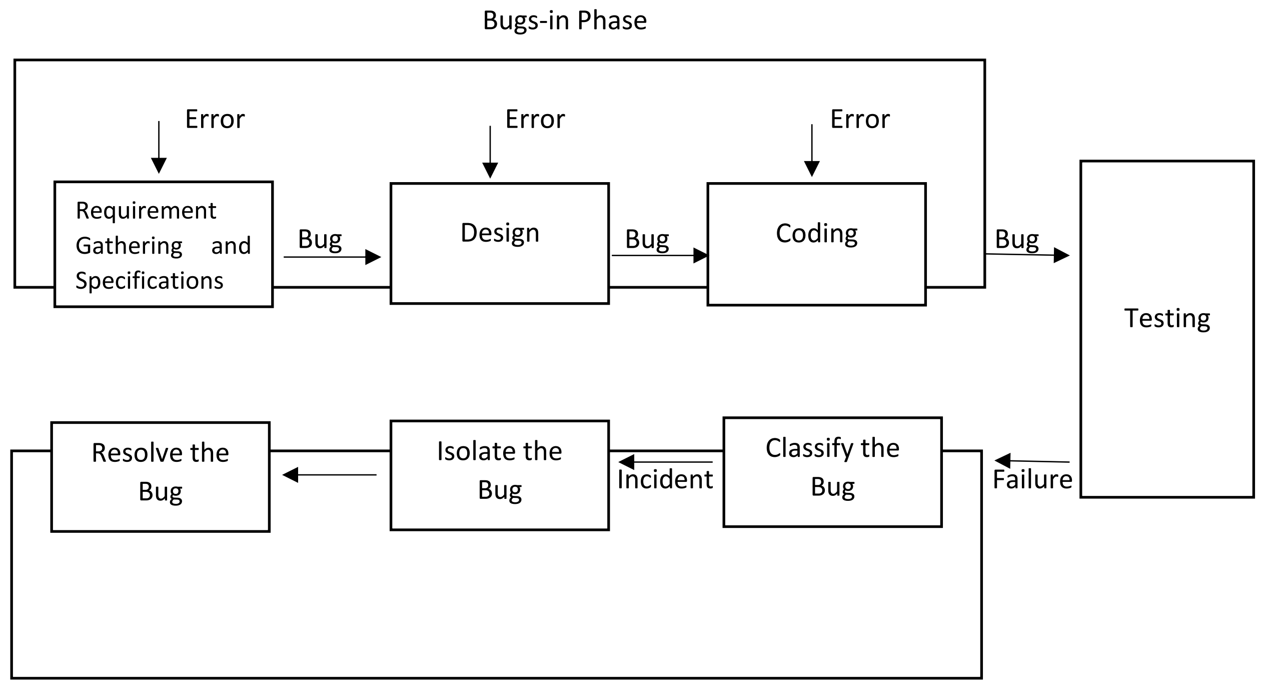
T-06	Verify that the system has an admin for doctors as well.	1. Click on the Doctors from the menu		Should have doctor module	As Expected	Pass
T-07	Verify that for each doctor's details like their timings, specialty, fee, patient visited etc is visible to the authorized users.	1. Click on the Department from the menu 2. Select a department 3. Select a doctor from this department		Should have doctor's information, timings, specialty, fee etc	As Expected	Pass
T-08	Verify that new details of new doctors can be added to the system.	1. Click on the Doctors from the menu 2. Click on Add button to add a new doctor 3. Give all the required information about the doctor 4. Click on submit button		Should add Doctor and have required information about the doctor	As Expected	Pass
T-09	Verify that the details of existing users can be updated in the system.	1. Click on the Account Settings from the menu 2. Select Edit option 3. Edit required fields 4. After finishing edit, click on the submit button	Password: !073eq@b Address: Maijdee, Noakhali	Should update required information	As Expected	Pass
T-10	Verify that the doctor's record can be deleted from the system.	1. Click on the Department from the menu 2. Select a Department 3. Select a Doctor 4. Click on manage doctor 5. Select Delete button		Should delete a Doctor from the System	As Expected	Pass

		6. Select Yes button from the popup box				
T-11	Verify the billing admin of the system calculates the bill based on the patient's unique application number from the data generated from different systems.	1. Click on the Bill from menu 2. Select Create a New Bill for Patient's 3. Enter Application Number 4. Enter Patient's Bill information 5. Calculate total bill	Application No: E09C1 Bill Information	Should Calculate the Patient's Bill	As Expected	Pass
T-12	Verify that the hard copy of the bill can be generating by printing the bill.	1. Click on the Bill from menu 2. Select Create a New Bill for Patient's 3. Enter Application Number 4. Enter Patient's Bill information 5. Calculate total bill 6. Print the bill	Application No: E09C1 Bill Information	Should print the bill	As Expected	Pass
T-13	Verify that authorized users can also see total day-wise billing done.	1. Enter User ID and Password 2. Click Login 3. Click on the Bill from menu 4. Select a date 5. Click on the submit button	User ID: Firoj47 Password: !073eq@b Date: 23 January, 2021	Can see total day-wise billing done	As Expected	Pass

T-14	Verify the admin for hospital inventory, room and bed management.	1. Select Hospital Inventory from the menu 2. Select Room from the menu 3. Select Bed Management from the menu		Should have an admin who can manage Hospital Inventory, Room and Bed Management	As Expected	Pass
T-15	Verify that the admin has the record of all the equipment, machines and medicines and the same gets updated when used or added to the system.	1. Select Hospital Records from the menu 2-a. Click on the Records of Equipment 3-a. Select add button 4-a. Enter new Equipment details 5-a. Click on the OK button 2-b. Click on the Records of Machines 3-b. Same as 3-a 4-b. Same as 4-a 5-b. Same as 5-a 2-c. Click on the Records of Medicines 3-c. Same as 3-a 4-c. Same as 4-a 5-c. Same as 5-a	Equipment: 4 Beds Room: B-403 Cost: 8000	Should add new records to the system	As Expected	Pass

T-16	Verify that the admin has a record of rooms and beds availability and the same gets updated based on their allotment and departure to patients	1. Select Rooms from the menu 2-a. Enter Room number 3-a. Select an available bed 4-a. Enter Patient-Card number 5-a. Click on the done button to allot the bed for the patient 2-b. Enter Patient-Card number 3-b. Click on the Depart button 4-b. Click OK button	Room: B-403 Bed: B-5 Patient-Card No: 1234-5678-9	a. Should allot a bed for a patient b. Should depart a patient	As Expected	Pass

- **Draw the life cycle of a bug.**

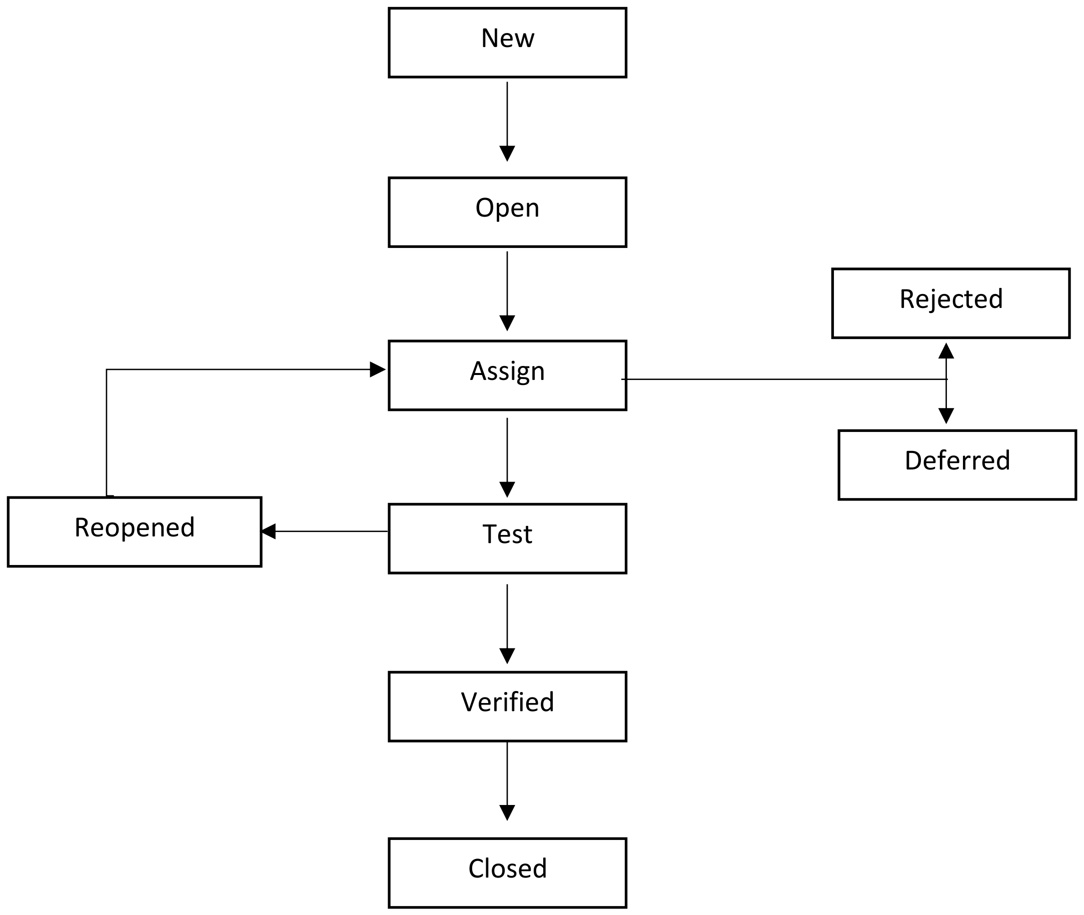


- **What are the activities to be performed to get rid of the bugs?**
  When developers develop a software, they don't know where the bugs. They can find bugs easily, if they use bug model at the beginning of software development. So, first of all they should prevent the software from bugs. There are three points to be performed to get rid of bugs.
  Bug classification: In that case, testers observe the failure and classify the bugs according to its nature. A bug can be critical or catastrophic in nature or it may have no adverse effect on the output behavior of the software. In this way, we classify all the failures. This is necessary, because there may be many bugs to be resolved. But a tester may not have sufficient time. Thus, categorization of bugs may help by handling high criticality bugs first and considering other trivial bugs on the list later, if time permits. We have taken various considerations to classify different bugs.

  Bug Isolation: Bug isolation is the activity by which we locate the module in which the bug appears. Incidents observed in failures help in this activity. We observe the symptoms and back-trace the design of the software and reach the module and the condition inside it which has caused the bug. This is known as bug isolation.

  Bug Resolution: Once we have isolated the bug, we back-trace the design to pinpoint the location of the error. In this way, a bug is resolved when we have found the exact location of its occurrence.

- **Draw the states of a bug.**



- **Describe the states of a bug.**
  A bug attains the different states in its life cycle that I have shown above.
  New: The state is new when the bug is reported first time by a tester.
  Open: The new state does not verify that the bug is genuine. When the test leader approves that the bug is genuine, its state becomes open.
  Assign: An open bug comes to the development team where the development team verifies its validity. If the bug is valid, a developer is assigned the job to fix it and the state of the bug now is 'ASSIGN'.
  Deferred: The developer who has been assigned to fix the bug will check its validity and priority. If the priority of the reported bug is not high or there is not sufficient time to test it or the bug does not have any adverse effect on the software, then the bug is changed to deferred state which implies the bug is expected to be fixed in next releases.
  Rejected: It may be possible that the developer rejects the bug after checking its validity, as it is not a genuine one.
  Test: After fixing the valid bug, the developer sends it back to the testing team for next round of checking. Before releasing to the testing team, the developer changes the bug's state to 'TEST'. It specifies that the bug has been fixed by the development team but not tested and is released to the testing team.
  Verified/fixed: The tester tests the software and verifies whether the reported bug is fixed or not. After verifying, the developer approves that the bug is fixed and changes the status to 'VERIFIED'.
  Reopened: If the bug is still there even after fixing it, the tester changes its status to 'REOPENED'. The bug traverses the life cycle once again. In another case, a bug which has been closed earlier may be reopened if it appears again. In this case, the status will be REOPENED instead of OPEN.
  Closed: Once the tester and other team members are confirmed that the bug is completely eliminated, they change its status to 'CLOSED'.

- **How do bugs get into a software product?**
  There may be various reasons: unclear or constantly changing requirements, software complexity, programming errors, timelines, errors in bug tracking, communication gap, documentation errors, deviation from standards, etc. Some examples are given below:
  - Miscommunication in gathering requirements from the customer is a big source of bugs.
  - If the requirements keep changing from time to time, it creates a lot of confusion and pressure, both on the development as well as the testing team. Often, a new feature added or an existing feature removed can be linked to other modules or components in the software.
  - If the effect of changes in one module to another module is overlooked, it causes bugs.
  - Rescheduling of resources, re-doing or discarding an already completed work, and changes in hardware/software requirements can affect the software. Assigning a new developer to the project midway can cause bugs. If proper coding standards have not

been followed, then the new developer might not get all the relevant details of the project. Improper code documentation and ineffective knowledge transfer can limit the developer's ability to produce error-free codes. Discarding a portion of the existing code might just leave its trail behind in other parts of the software. Overlooking or not eliminating such code can cause bugs. Serious bugs can especially occur with larger projects, as it gets tougher to identify the problem area.

- Complexity in keeping a track of all the bugs can in turn cause more bugs. This gets harder when a bug has a very complex life cycle, i.e. when the number of times it has been closed, re-opened, not accepted, ignored, etc. goes on increasing.

➢ **Describe bugs classification based on Criticality.**

Bugs can be classified based on the impact they have on the software under test. This classification can be used for the prioritization of bugs, as all bugs cannot be resolved in one release. Since all bugs are not of the same criticality, prioritization will put high criticality bugs on top of the list. We can divide bugs based on their criticality in the following broad categories:

Critical Bugs: This type of bugs has the worst effect such that it stops or hangs the normal functioning of the software. The person using the software becomes helpless when this type of bug appears. For example, in a sorting program, after providing the input numbers, the system hangs and needs to be reset.
Major Bug: This type of bug does not stop the functioning of the software but it causes a functionality to fail to meet its requirements as expected. For example, in a sorting program, the output is being displayed but not the correct one.
Medium Bugs: Medium bugs are less critical in nature as compared to critical and major bugs. If the outputs are not according to the standards or conventions, e.g. redundant or truncated output, then the bug is a medium bug.
Minor Bugs: These types of bugs do not affect the functionality of the software. These are just mild bugs which occur without any effect on the expected behavior or continuity of the software. For example, typographical error or misaligned printout.

➢ **Describe bug classification based on SDLC**.

Since bugs can appear in any phase of SDLC, they can be classified based on SDLC phases which are described below:
Requirements and Specifications Bugs: The first type of bug in SDLC is in the requirement gathering and specification phase. It has been observed that most of the bugs appear in this phase only. If these bugs go undetected, they propagate into subsequent phases. Requirement gathering and specification is a difficult phase in the sense that requirements gathered from the customer are to be converted into a requirement specification which will become the base for design. There may be a possibility that requirements specified are not exactly what the customers want. Moreover, specified requirements may be incomplete, ambiguous, or inconsistent. Specification problems lead to wrong missing, or superfluous features.

Design Bugs: Design bugs may be the bugs from the previous phase and in addition those errors which are introduced in the present phase. Some of the design bugs are

- Control Flow Bugs
- Logic Bugs: Any type of logical mistakes made in the design is a logical bug.
- Processing Bugs: Any type of computation mistakes results in processing bugs.
- Data Flow Bugs: Control flow cannot identify data errors. For this, we use data flow. There may be data flow anomaly errors like un-initialized data, initialized in wrong format, data initialized but not used, data used but not initialized, redefined without any intermediate use, etc.

Coding Bugs: There may be a long list of coding bugs. If you are a programmer, then you are aware of some common mistakes made. For example, undeclared data, undeclared routines, dangling code, typographical errors, documentation bugs, i.e. erroneous comments lead to bugs in maintenance.

Interface and Integration Bugs: External interface bugs include invalid timing or sequence assumptions related to external signals, misunderstanding external input and output formats, and user interface bugs. Internal interface bugs include input and output format bugs, inadequate protection against corrupted data, wrong subroutine call sequence, call parameter bugs, and misunderstood entry or exit parameter values. Integration bugs result from inconsistencies or incompatibilities between modules discussed in the form of interface bugs. There may be bugs in data transfer and data sharing between the modules.

System Bugs: There may be bugs while testing the system as a whole based on various parameters like performance, stress, compatibility, usability, etc. For example, in a real-time system, stress testing is very important, as the system must work under maximum load. If the system is put under maximum load at every factor like maximum number of users, maximum memory limit, etc. and if it fails, then there are system bugs.

Testing Bugs: One can question the presence of bugs in the testing phase because this phase is dedicated to finding bugs. But the fact is that bugs are present in testing phase also. After all, testing is also performed by testers – humans. Some testing mistakes are: failure to notice/report a problem, failure to use the most promising test case, failure to make it clear how to reproduce the problem, failure to check for unresolved problems just before the release, failure to verify fixes, failure to provide summary report.