



System Testing

Dr. Sangharatna Godbole
Assistant Professor
Department of CSE
NIT Warangal

System Testing

- There are three main types of system testing:
 - **Alpha Testing**
 - **Beta Testing**
 - **Acceptance Testing**

Alpha Testing

- System testing carried out by the test team, within the developing organization.
 - **Test cases are designed based on the SRS document**
- The test environment can be controlled a little in this case.

Entry Criteria to Alpha

- All features are complete/testable (no urgent bugs).
- High bugs on primary platforms are fixed/verified.
- 50% of medium bugs on primary platforms are fixed/verified.
- All features have been tested on primary platforms.
- Performance has been measured/compared with previous releases (user functions).
- Usability testing and feedback (ongoing).
- Alpha sites are ready for installation.

Exit Criteria from Alpha

After alpha testing, we must:

- Get responses/feedbacks from customers.
- Prepare a report of any serious bugs being noticed.
- Notify bug-fixing issues to developers.

Beta Testing

- System testing performed by a selected group of friendly customers, normally at customer site.
- The development team does not have any control over the test environment.

Entry Criteria to Beta

- Positive responses from alpha sites.
- Customer bugs in alpha testing have been addressed.
- There are no fatal errors which can affect the functionality of the software.
- Secondary platform compatibility testing is complete.
- Regression testing corresponding to bug fixes has been done.
- Beta sites are ready for installation.

Guidelines for Beta Testing

- Don't expect to release new builds to beta testers more than once every two weeks.
- Don't plan a beta with fewer than four releases.
- If you add a feature, even a small one, during the beta process, the clock goes back to the beginning of eight weeks and you need another 3– 4 releases.

Exit Criteria from Beta

After beta testing, we must:

- Get responses/feedbacks from the beta testers.
- Prepare a report of all serious bugs.
- Notify bug-fixing issues to developers.

Acceptance Testing

- System testing performed by the customer himself:
 - **To determine whether the system should be accepted or rejected.**

Acceptance Testing cont...

- *Acceptance testing is the formal testing conducted to determine whether a software system satisfies its acceptance criteria and to enable buyers to determine whether to accept the system or not.*
- Acceptance testing must take place at the end of the development process.
- It consists of tests to determine whether the developed system meets the predetermined functionality, performance, quality, and interface criteria acceptable to the user.

Acceptance Testing cont...

- *Therefore, the final acceptance acknowledges that the entire software product adequately meets the customer's requirements.*
- *User acceptance testing is different from system testing.*
- *System testing is invariably performed by the development team which includes developers and testers.*
- *User acceptance testing, on the other hand, should be carried out by end-users.*

Acceptance Testing cont...

Thus, acceptance testing is designed to:

- Determine whether the software is fit for the user.
- Making users confident about the product.
- Determine whether a software system satisfies its acceptance criteria.
- Enable the buyer to determine whether to accept the system or not.

Acceptance Testing cont...

- The final acceptance marks the completion of the entire development process.
- It happens when the customer and the developer has no further problems.
- Acceptance test might be supported by the testers.
- It is very important to define the acceptance criteria with the buyer during various phases of SDLC.
- A well-defined acceptance plan will help development teams to understand users' needs.
- The acceptance test plan must be created or reviewed by the customer.

Acceptance Testing cont...

- The development team and the customer should work together and make sure that they:
- Identify interim and final products for acceptance, acceptance criteria, and schedule.
- Plan how and by whom each acceptance activities will be performed.
- Schedule adequate time for the customer to examine and review the product.
- Prepare the acceptance plan.
- Perform formal acceptance testing at delivery.
- Make a decision based on the results of acceptance testing.





Entry Criteria to Acceptance

- System testing is complete and defects identified are either fixed or documented.
- Acceptance plan is prepared and resources have been identified.
- Test environment for the acceptance testing is available.

Exit Criteria from Acceptance

- Acceptance decision is made for the software.
- In case of any warning, the development team is notified.

- 
- In all types of system tests, the test cases can be the same,
 - but the difference is with respect to who designs test cases and carries out testing.

- 
- The system test cases can be classified into functionality and performance test cases.
 - Before a fully integrated system is accepted for system testing, **smoke testing** is performed.

Smoke testing

The idea behind smoke testing is that

- if the integrated program cannot pass even the basic tests, it is not ready for a vigorous testing.
- For smoke testing, a few test cases are designed to check whether the basic functionalities are working.
- For example, for a library automation system, the smoke tests may check whether books can be created and deleted, whether member records can be created and deleted, and whether books can be loaned and returned.

Performance Testing

- Addresses non-functional requirements.
 - **May sometimes involve testing hardware and software together.**
 - **There are several categories of performance testing.**

Stress Testing

- Stress testing (also called endurance testing):
 - **Impose abnormal input to stress the capabilities of the software.**
 - **Input data volume, input data rate, processing time, utilization of memory, etc. are tested beyond the designed capacity.**

Stress Testing

- If the requirements is to handle a specified number of users, or devices:
 - **Stress testing evaluates system performance when all users or devices are busy simultaneously.**

Stress Testing

- If an operating system is supposed to support 15 multiprogrammed jobs,
 - **The system is stressed by attempting to run 15 or more jobs simultaneously.**
- A real-time system might be tested
 - **To determine the effect of simultaneous arrival of several high-priority interrupts.**

Stress Testing

- Stress testing usually involves an element of time or size,
 - Such as the number of records transferred per unit time,
 - The maximum number of users active at any time, input data size, etc.
- Therefore stress testing may not be applicable to many types of systems.

Volume Testing

- Addresses handling large amounts of data in the system:
 - Whether data structures (e.g. queues, stacks, arrays, etc.) are large enough to handle all possible situations.
 - Fields, records, and files are stressed to check if their size can accommodate all possible data volumes.

Configuration Testing

- Analyze system behavior:
 - in various hardware and software configurations specified in the requirements
 - sometimes systems are built in various configurations for different users
 - for instance, a minimal system may serve a single user,
 - other configurations for additional users.

Compatibility Testing

- These tests are needed when the system interfaces with other systems:
 - Check whether the interface functions as required.

Compatibility testing Example

- If a system is to communicate with a large database system to retrieve information:
 - A compatibility test examines speed and accuracy of retrieval.

Recovery Testing

- These tests check response to:
 - Presence of faults or to the loss of data, power, devices, or services
 - Subject system to loss of resources
 - Check if the system recovers properly.

Maintenance Testing

- Diagnostic tools and procedures:
 - help find source of problems.
 - It may be required to supply
 - memory maps
 - diagnostic programs
 - traces of transactions,
 - circuit diagrams, etc.

Maintenance Testing

- Verify that:
 - all required artifacts for maintenance exist
 - they function properly

Documentation tests

- Check that required documents exist and are consistent:
 - user guides,
 - maintenance guides,
 - technical documents

Documentation tests

- Sometimes requirements specify:
 - Format and audience of specific documents
 - Documents are evaluated for compliance



Latent Errors: How Many Errors are Still Remaining?

- Make a few arbitrary changes to the program:
 - Artificial errors are seeded into the program.
 - Check how many of the seeded errors are detected during testing.

Error Seeding

- Let:
 - N be the total number of errors in the system
 - n of these errors be found by testing.
 - S be the total number of seeded errors,
 - s of the seeded errors be found during testing.

Error Seeding

- $n/N = s/S$
- $N = S \cdot n/s$
- Remaining defects:
$$N - n = n \cdot ((S - s)/s)$$

Quiz 2

- 100 errors were introduced.
- 90 of these errors were found during testing
- 50 other errors were also found.
- Find errors remaining in code.

Quiz 2: Solution

- 100 errors were introduced.
- 90 of these errors were found during testing
- 50 other errors were also found.
- Remaining errors=
 $50 \quad (100-90)/90 = 6$

Error Seeding

- The kinds of seeded errors should match closely with existing errors:
 - However, it is difficult to predict the types of errors that exist.
- Categories of remaining errors:
 - Can be estimated by analyzing historical data from similar projects.

Quiz 3

- Before system testing 100 errors were seeded.
- During system testing 90 of these were detected.
- 150 other errors were also detected
- How many unknown errors remain after system testing?



Thank You