Software Testing (UE18CS400SB) Unit 4

Aronya Baksy

November 2021

1 Acceptance Testing

- Testing done in accordance with customer specified criteria (mutually agreed)
- Done in the customer environment by customer-specified individuals
- Types of Acceptance testing:
  - User Acceptance Testing
  - Business Acceptance Testing
  - Contract Acceptance Testing
  - Regulations Acceptance Testing
  - Operational Acceptance Testing
  - α and β testing
- Approaches to acceptance testing:
  - **Design / Architecture based approach**: Full functionality, Cases that fail here may trigger review
  - **Business vertical / customized instance**: Typically for products / frameworks Customized instances are put through testing based on business domain
  - **Deployment focused**: Large applications tested in complex hw or sw environments
- Criteria for acceptance testing:
  - SPecifying the business requirements (may be functional or non functional)
  - May be process requirements (test coverage, training etc.)
- Acceptance test cases cover: Critical functionality, End-to-end scenarios. New functionalities – during upgrade, Legal / statutory needs, Functionality to work on a defined corpus
- Acceptance test execution: on site, in the presence of dev engineers, with proper documentation of the execution process
- Challenges: criteria not easy to specify, multiple iterations with multiple cust. representatives needed

2 Non-Functional Testing

- Testing the non-functional attributes (performance, reliability, scalability, load, security, compatibility and others) of the system
- Reasons:
  - Identify and correct design faults
  - Identify limits of the system
  - Whether the product can behave gracefully during stress and load conditions
  - To ensure that the product can work without degrading for a long duration
  - To compare with other products and previous versions
  - To avoid un-intended side effects.
- When the product has no basic issues and meets the minimum entry criteria, then the non-functional test can start.
- The non-functional testing is stopped when there is enough data to make a judgement. This must be aligned with release schedule

2.1 Types of Non-Functional Testing

2.1.1 Scalability Test

- Ability of a system to handle increasing amounts of work without unacceptable level of performance (degradation)
- Scalability may be vertical or horizontal
- The test cases will focus on to test the maximum limits of the features, utilities and performing some basic operations

2.1.2 Performance Test

- Evaluating response time of product to perform required actions under stated conditions
- comparison with different versions of same product and competitive products.
- Test cases focus on getting response time and throughput for different operations, under defined environment and load

2.1.3 Reliability Test

- Evaluate the ability of the product to perform it's required functions under stated conditions for a specified period of time or number of iterations
- The test cases will focus on the product failures when the operations are executed continuously for a given duration or iterations
- Focus on frequently used operations

2.1.4 Stress Test

- Evaluate a system beyond the limits of the specified requirements or environment resources to ensure the product behavior is acceptable.
- Well-designed system shows graceful degradation and safe behaviour
- System should show performance decrease when resource/load ratio reduces, and symmetric increases when load is reduced

2.1.5 Security Test

- Both static and dynamic in nature
- Test tools identify vulnerabilities at application level (access control, SQL injection, buffer overflow, encryption, API design)

2.1.6 Regression Test

- A **black-box** technique, that ensures that code change does not impact existing functionality
- Types of regression testing:
  - **Corrective**: No changes to requirements, reuse existing test cases
  - **Retest-all**: Re-doing all tests again, not advisable for minor changes
  - **Selective**: Using a subset of the test cases, analyze impact of new code on existing code
  - **Progressive**: When change in spec, new test cases, ensures that new features do not break existing ones
  - **Complete**: Used for major changes in code.
  - **Partial**: Tests issues when new code is added to already existing code, ensures that a system continues to work after adding new code
  - **Unit**: focus on a single unit, block all dependencies