**Introduction and Testing Body of Knowledge**

* Software testing fundamentals,
* Test levels,
* Test techniques,
* Test related measures,
* Test process
* Test Documentation

Software testing life cycle identifies what test activities to carry out and when (what is the best time) to accomplish those test activities. Even though testing differs between organizations, there is a testing life cycle.

**Cycle of testing**

* Requirements Analysis: Testing should begin in the requirements phase of the software development life cycle.
* Design Analysis: During the design phase, testers work with developers in determining what aspects of a design are testable and under what parameter those tests work.
* Test Planning: Test Strategy, Test Plan(s), Test Bed creation.
* Test Development: Test Procedures, Test Scenarios, Test Cases, Test Scripts to use in testing software.
* Test Execution: Testers execute the software based on the plans and tests and report any errors found to the development team.
* Test Reporting: Once testing is completed, testers generate metrics and make final reports on their test effort and whether or not the software tested is ready for release.
* Retesting the Defects

**Introduction**

* *Testing*: set of activities to check if software output matches with theoretical results.
* Dynamic.
* Finite set *selected* from many test cases.
* *Verification*: Testing against a specification (is the software doing the thing right?)
* *Validation*: Testing against user expectations (is the software doing the right thing?)

**Software Testing Fundamentals**

**Terminology:**

* Fault, Failure, Error ...

**Key Issues:**

* Test selection criteria,
* Effectiveness,
* Defect identification,
* Deciding to pass or fail (oracle problem),
* Limitations,
* Testability.

**Other activities:**

* Correctness proof,
* Testing vs. debugging,
* Testing vs. programming

**Test Levels**

**Target**

* **Unit testing**
* **Integration testing**
* **System testing**

**Objectives (some)**

Acceptance / qualification / installation,

Alpha & beta,

Conformance / functional / correctness,

Regression,

Performance / stress,

Recovery / Configuration,

Usability,

Test-driven development

**GUI Testing**

**Windows Compliance Standards**

1.1. Application   
1.2. For Each Window in the Application   
1.3. Text Boxes

1.4. Option (Radio Buttons)   
1.5. Check Boxes

1.6. Command Buttons   
1.7. Drop Down List Boxes

1.8. Combo Boxes

1.9. List Boxes

**Tester's Screen Validation Checklist**

2.1. Aesthetic Conditions   
2.2. Validation Conditions   
2.3. Navigation Conditions   
2.4. Usability Conditions   
2.5. Data Integrity Conditions  
2.6. Modes (Editable Read-only) Conditions

2.7. General Conditions   
2.8. Specific Field Tests   
2.8.1. Date Field Checks   
2.8.2. Numeric Fields   
2.8.3. Alpha Field Checks

**Validation Testing - Standard Action**

3.1. On every Screen   
3.2. Shortcut keys / Hot Keys   
3.3. Control Shortcut Keys

**Testing essentially involves 3 basic steps**

Selecting the inputs

Applying to the system

Checking the output

**Software System Testing**

* A series of system integration and validation tests
* To fully exercise the software-based system
* Simulating actual use environment
* Along with other system elements like target hardware, field data, etc.
* By people with little knowledge of how the software is implemented only Black box view

**User Acceptance Testing (UAT)**

The question of what User Acceptance Testing (UAT) is and why as a business user are you interested in it, becomes very important as you reach a stage of having a piece of software delivered to you. The system is claimed to give you major benefits, your team is already stretched running your organization, and you can ill afford to use their time on something that may be a waste of it. So why bother?

**What Is User Acceptance Testing?**

* The explanation of UAT is actually in its name. Taking each part separately explains what it is about.

Users mean the real business users, User who will have to operate the system; normally the staff of an organization, but it could be your suppliers or customers. They are the only people who understand exactly what the business is, and how it operates. Therefore they are the only people qualified to check a system to see if it will deliver any benefit to the business or organization.

* System developers cannot do it, as although they are expert in writing software, they are unlikely to know anything about the realities of running the organization, other then what they have acquired from requirements specifications, and similar documents. In addition they have been closely involved in the design compromises that always take place when a system is being developed, and so have a commitment to the system as it is.

**Acceptance**

* The acceptance of a system means you are confident it will give benefit to the organization. It does not mean that it only meets the original specification as requested. A system may very well meet all the specifications asked of it, but when trying to see how it will work in the business it is realized that it will not give anything positive to the organization, or may even damage the organization. This may be for a number of reasons such as a change in the business or business environment, such as a takeover.
* The point is that a system may not be acceptable, even if it meets specification. You may still have to pay the supplier, but you will not incur any costs to implement it. Of course it may not even work to specification, which makes the question of acceptance even harder to answer. There are cases where it is worth implementing and paying for systems, which are imperfect, but that deliver real business value.