TESTING THROUGHOUT THE SOFTWARE LIFE CYCLE (II)

COMPILED BY –

NAZMUS SAKIB AKASH

2 Testing types: The targets of testing
Test types and Test levels

Test levels

- The previous section explained the various testing levels, i.e. component test, integration test etc.
- At every test level, the test objectives have a different focus!
- Therefore different test types are applied during different test levels.

Test types

- Functional testing
  - (Goal: Testing of function)
- Non-functional testing
  - (Goal: Testing product characteristics)
- Structural Testing
  - (Goal: Testing of SW structure/architecture)
- Confirmation/regression testing
  - (Goal: Testing after changes)

3 Testing types: The targets of testing
Functional Testing

- **Goal: the function of the test object**
  - Functionality can be linked to input and output data of the test object
  - ***Black box methods*** are applied to design the relevant test cases
  - Testing is to verify functional requirements (as stated in specifications, concepts, case studies, business rules or relevant documents)
- **Area of use**
  - Functional testing may be performed at all test levels

- **Execution**
  - The test object is executed using test data derived from test cases
  - The result of the test execution are compared to the expected results
  - **Security testing**
    - Type of functional testing delaying with external threats.
    - Malicious attacks could damage program or data.

Testing types: The targets of testing
Non Functional Testing

- **Goal: software product characteristics**
  - How well does the software perform its functions?
  - The non-functional quality characteristics (ISO 9126): ***reliability***, ***usability***, ***efficiency***, ***maintainability***, ***portability*** are often vague, incomplete or missing all together, making testing difficult.

- **Execution**
  - Compliance with the non-functional requirements is measured using selected functional requirements

- **Area of use**
  - Non-Functional testing may be performed at **all test levels**
  - Typical non-functional testing:
    - Load testing/ performance testing/ volume testing/ stress testing
    - Testing of safety features
    - Reliability and robustness testing / compatibility testing
    - Usability testing / configuration testing

5 Non Functional Testing I (System Test)

- **Load test**
  - System under load (minimum load, more user/tractions)
- **Performance test**
  - How fast does the system perform a certain function?
- **Volume test**
  - Processing huge volumes of data / files
- **Stress test**
  - Reaction to overload / recovery after return to normal
- **Reliability test**
  - Performance while in "continuous operation mode"
- **Test of robustness**
  - Reaction to input of wrong or unspecified data
  - Reaction to hardware failures / disaster recovery

6 Non Functional Testing II (System Test)

- **Compliance testing**
  - Meeting rules and regulations (internal / external)
- **Test usability**
  - Structured, understandable, easy to learn for user
- **Other non-functional quality aspects:**
  - **portability:** replace ability, install-ability, conformance/ compliance, adaptability
  - **maintainability:** verifiability, stability, analyzability, changeability
  - **reliability:** maturity, robustness, recoverability

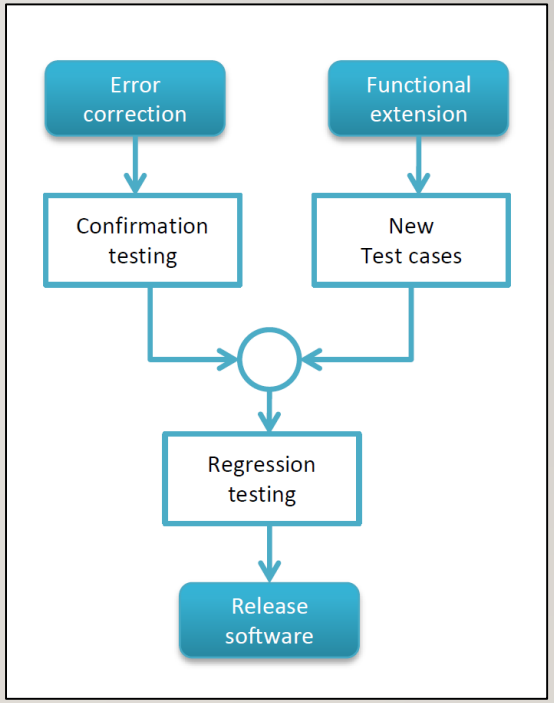
7 Structural testing

- **Goal: Coverage**
  - Analyses the structure of the test object (**white box approach**)
  - Testing aims at *measuring how well the structure of the test object is covered by the test case*
- **Area of use**
  - Structural testing possible on **all test levels**, *code coverage testing* using tools mainly done during **component and integration testing**.
  - Structural test design is finalized after *functional tests have been designed*, aiming at producing a high degree of coverage.

- **Execution**
  - Will test the internal structure of a test object (e.g. control flow within components, flow through a menu structure)
  - Goal: all identified structural elements should be covered by test cases

8 Confirmation / Regression testing I

- **Goal: test object after changes**
  - After a test object or its system environment has been changed, results related to the change have become invalid - test has to be repeated.
- Two main reasons for changing software:
  - Error correction
  - Functional extension
- Because of undesired side effects of extended or new functionality, it is necessary to also **retest adjacent areas**!



9 Confirmation / Regression testing II

- **Area of use**
  - Replacing a test of functionality that has already been verified is called a regression test.
  - The **scope** of the regression test depends on the **risk**, that the newly implemented functionality (extension or error fix) imposes to the system.
  - Analyzing this risk can be done with an **impact analysis**
  - Confirmation / Regression testing may be performed at **all test levels**.
  - Typical test after changes are:
    - Confirmation testing (= Testing **after correction of errors**)
    - Regression testing (= Testing **to uncover newly introduced defects**)

10 Confirmation / Regression testing III

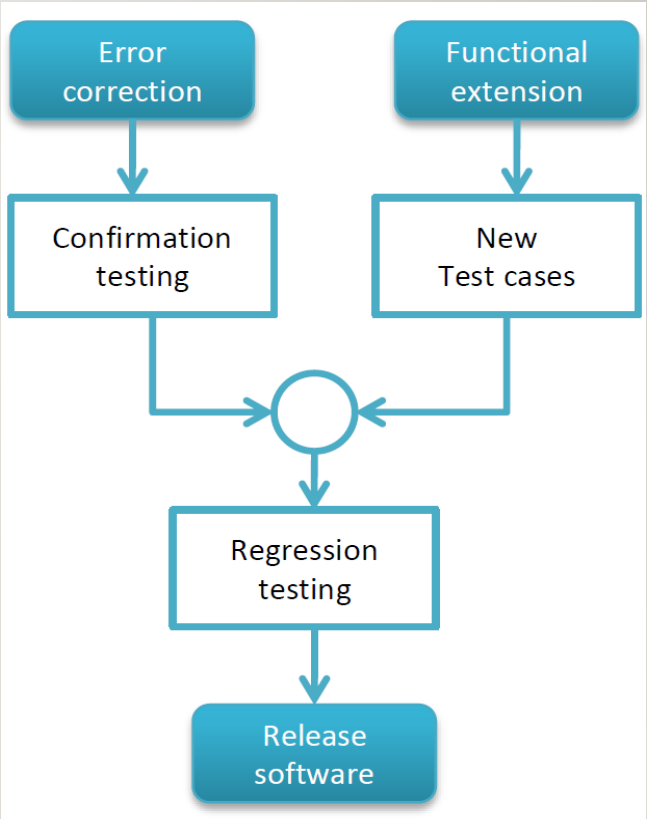
- **Execution**
  - Basically, execution takes place as in previously executed test iterations
  - In most cases, **a complete regression test is not feasible**, because it is too expensive and takes too much time
  - A high degree of modularity in the software allows for **more appropriate reduced regression tests**
  - Criteria for the selection of the regression test cases:
    - Test **case with high priority**
    - Only test **standard functionality**, skip special cases and variations
    - Only test **configuration that is used most often**
    - Only test **subsystem / selected areas of the test object**
  - If during early project phases, it becomes obvious that certain tests are suitable for regression testing, *test automation should be considered*.

Testing after Product Acceptance I

- **Customer has approved the product and sets it into production**
  - The initial development cycle, including its related tests, has been completed
- The software itself is at the beginning of its life cycle:
  - it will be used for many years to come, it will be extended
  - it most likely still has errors, hence it will be further modified and corrected
  - it needs to adapt to new conditions and to be integrated into new environments
  - it will one day be retired, put out of operation.
- **Any new version of the product, any new update and any other change in the software requires additional testing!**

12 Testing after Product Acceptance II

- **Software maintenance covers two different fields:**
  - maintenance as such: **correction of error**, that already were part of the initial version of the software
  - **software extension**: adaptations as a result of a *changed environment* or *new customer requirements*
- **Test scope of maintenance testing**
  - Error correction requires **retests**
  - Extended functionality requires **new test case**
  - Migration to another platform requires **operational tests**
  - In addition, **intensive regression testing** is needed



13 Testing after Product Acceptance III

- **Scope of testing** is affected by the impact of the change
  - **Impact analysis** is used to determine the affected areas
  - Problems might occur if **documentation** of the *old software* is **missing** or **incomplete**
- Software **retirement**
  - Test after software retirement may include
    - Data migration test
    - Verifying archiving data and programs
    - Parallel testing of old and new systems

14 Summary I

- On different test levels, different types of tests are used
- Test types are: functional, non-functional, structural and change-related testing
- Functional testing examines the input / output behavior of a test object
- Non-functional testing checks product characteristics
- Non-functional testing includes, but is not limited to, load testing, stress testing, performance testing, robustness testing
- Common structural tests are tests that check data and control flow within the test object, measuring the degree of coverage
- Important test after changes are: confirmation tests(re-tests) and regression tests

15 Summary II

- Ready developed software needs to be adapted to new conditions, errors have to be corrected
- An impact analysis can help to judge the changes related risks
- Maintenance tests make sure , that
  - New function are implemented correctly (new test cases)
  - Error have been fixed successfully (old test cases)
  - Functionality, that has already been verified, is not affected (regression test)
- If software gets retired, migration tests or parallel tests may be necessary