WHITESTEIN
Technologies

Object-Oriented Software Engineering

Testing

Radovan Cervenka

Goals

- To verify the interaction between objects
- To verify the proper integration of all components of the software
- To verify that all requirements have been correctly implemented
- To identify and ensure defects are addressed prior to the deployment of the software

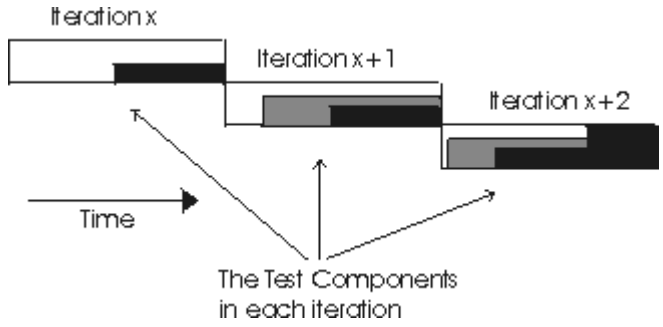
Concepts: Quality

→ the characteristic of having demonstrated the achievement of producing a product which meets or exceeds agreed upon requirements, as measured by agreed upon measures and criteria, and is produced by an agreed upon process

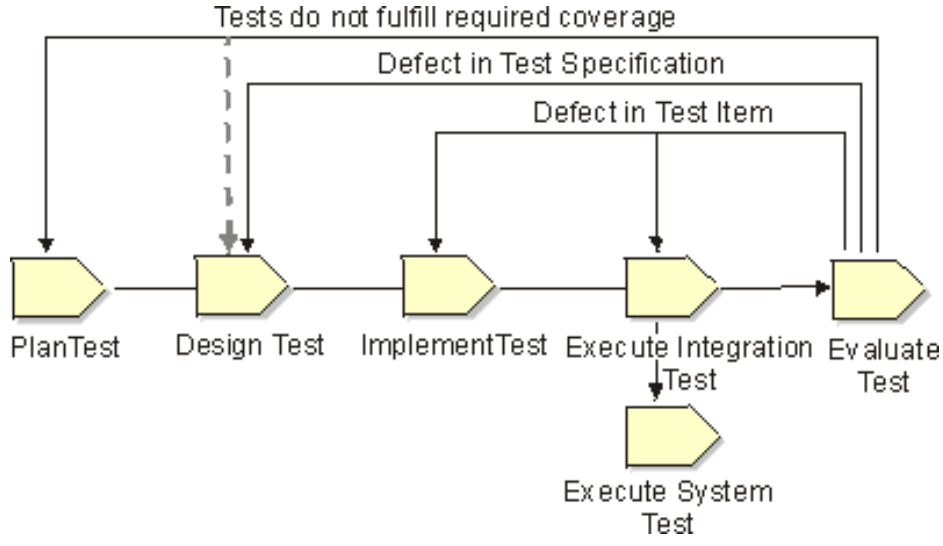
- Product quality & process quality
- Specified requirements
- Measurements
- Responsibility of everyone

Concepts: The Lifecycle of Testing

- An iterative approach with additions and refinements of the tests
- High focus on regression test
  - the same test is repeated several times in the following iterations



- The testing lifecycle



Concepts: Stages of Test

- Unit Test
  - applied to components in the implementation model
  - to verify that control flows and data flows are covered and function as expected
  - the Implementer performs unit test as the unit is developed
- Integration Test
  - target-of-test is a package or a set of packages in the implementation model
  - to ensure that the components in the implementation model operate properly when combined to execute a use case
- System Test
  - when the software is functioning as a whole, or when well-defined subsets of its behavior are implemented
  - the target is whole implementation model for the system

Concepts: Performance Testing

→ a class of tests that are implemented and executed to characterize and evaluate the performance related characteristics of an application/system

- Benchmark testing
  - measure the performance and compare it to a known reference system
- Performance testing
  - a constant workload and varying system variables to tune (or optimize) the performance of the system; # of transactions per minute, number of users, ...
- Load testing
  - verify and assess acceptability of the operational limits of a system under varying workloads while the system-under-test remains constant
- Stress testing
  - tests that focus on ensuring the system functions as intended when abnormal conditions are encountered
- Volume testing
  - testing that focuses on the ability of the system to handle large amounts of data; queries returning all data, data entries of maximum data in each field, ...

Concepts: Acceptance Testing

→ Goal: to verify that the software is ready and can be used by the end-users to perform those functions and tasks the software was built to do

→ the final test action prior to deploying the software

- Formal Acceptance Testing
  - highly managed process (carefully planned and designed)
  - often an extension of the system test (subset of those tests)
- Informal Acceptance Testing
  - no particular test cases to follow, only functions and business tasks to be explored are identified and documented
  - individual tester determines what to do; more subjective testing
- Beta Testing
  - the amount of detail, the data, and approach taken is entirely up to the individual tester
  - the most subjective

Concepts: Measures of Testing - Coverage Metrics

⇒ "How complete is the testing?"

- Requirements-based test coverage
  - → the coverage of testing, expressed by the coverage of test requirements and test cases

$$Test\ Coverage = T^{(p,i,x,s)} / RfT$$

where:

- $T^{(p,i,x,s)}$ is the number of Tests (planned, implemented, executed, or successful) as expressed as test procedures or test cases
- *RfT* is the total number of Requirements for Test

 x% of test cases ($T^{(p,i,x,s)}$) have been covered with a success rate of y%

Concepts: Measures of Testing - Coverage Metrics (cont.)

- Code-based test coverage
  - → how much code (lines of code, branch conditions, paths through the code, data flows and states) has been executed during the test, compared to how much code there is left to execute

$$Test\ Coverage = I^e / Tlic$$

where:

- I^e is the number of items executed expressed as code statements, code branches, code paths, data state decision points, or data element names
- *Tlic* is is the total number of items in the code

x% of test cases (I) have been covered with a success rate of y%

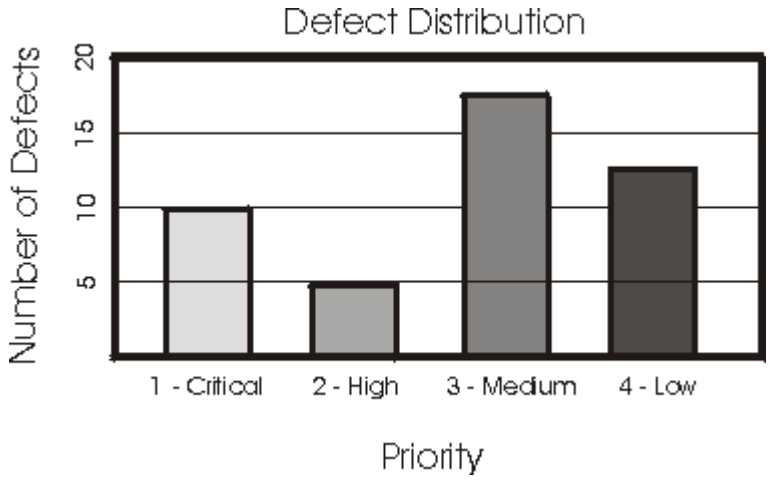
Concepts: Measures of Testing - Quality Metrics

⇒ "How reliable, stable and performance sufficient is the target system?"

- Evaluation of test and the analyses of defects discovered during the testing
- From simple defect counts to rigorous statistical modeling methods
- Defect analysis
  - → to analyze the distribution of defects over the values of one or more the parameters associated with a defect
  - parameters commonly used:
    - Status the current state of the defect (open, being fixed, closed, etc.)
    - Priority the relative importance of this defect having to be addressed and resolved
    - Severity the relative impact of this defect. The impact to the end-user, an organization, third parties, etc
    - Source where and what is the originating fault that results in this defect, or what component will be fixed to eliminate the defect

Concepts: Measures of Testing - Quality Metrics (cont.)

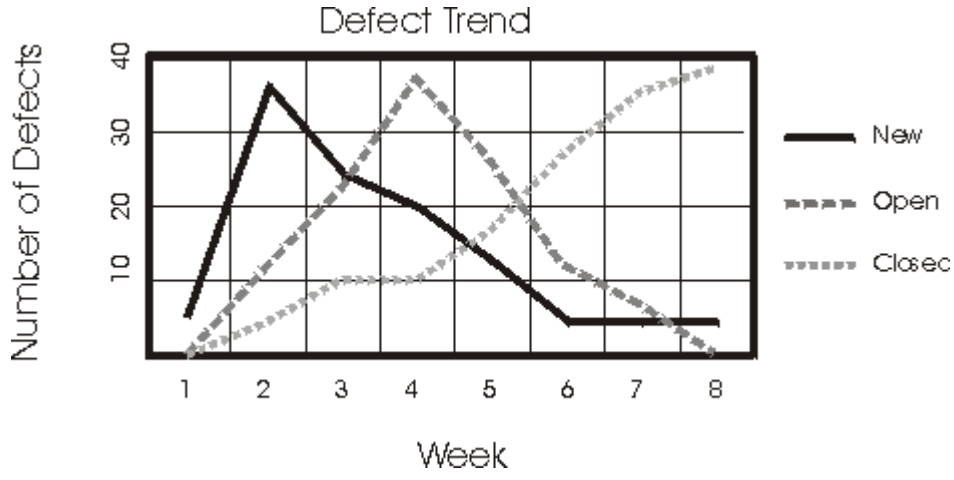
- Defect distribution reports
  - allow defect counts to be shown as a function of one or two defect parameters



- Defect age reports
  - a special type of defect distribution report
  - show how long a defect has been in a particular state
  - in any age category, defects can also be sorted by another attribute, like Owner

Concepts: Measures of Testing - Quality Metrics (cont.)

- Defect trend reports
  - show defect counts, by status (new, open, or closed), as a function of time
  - can be cumulative or non-cumulative



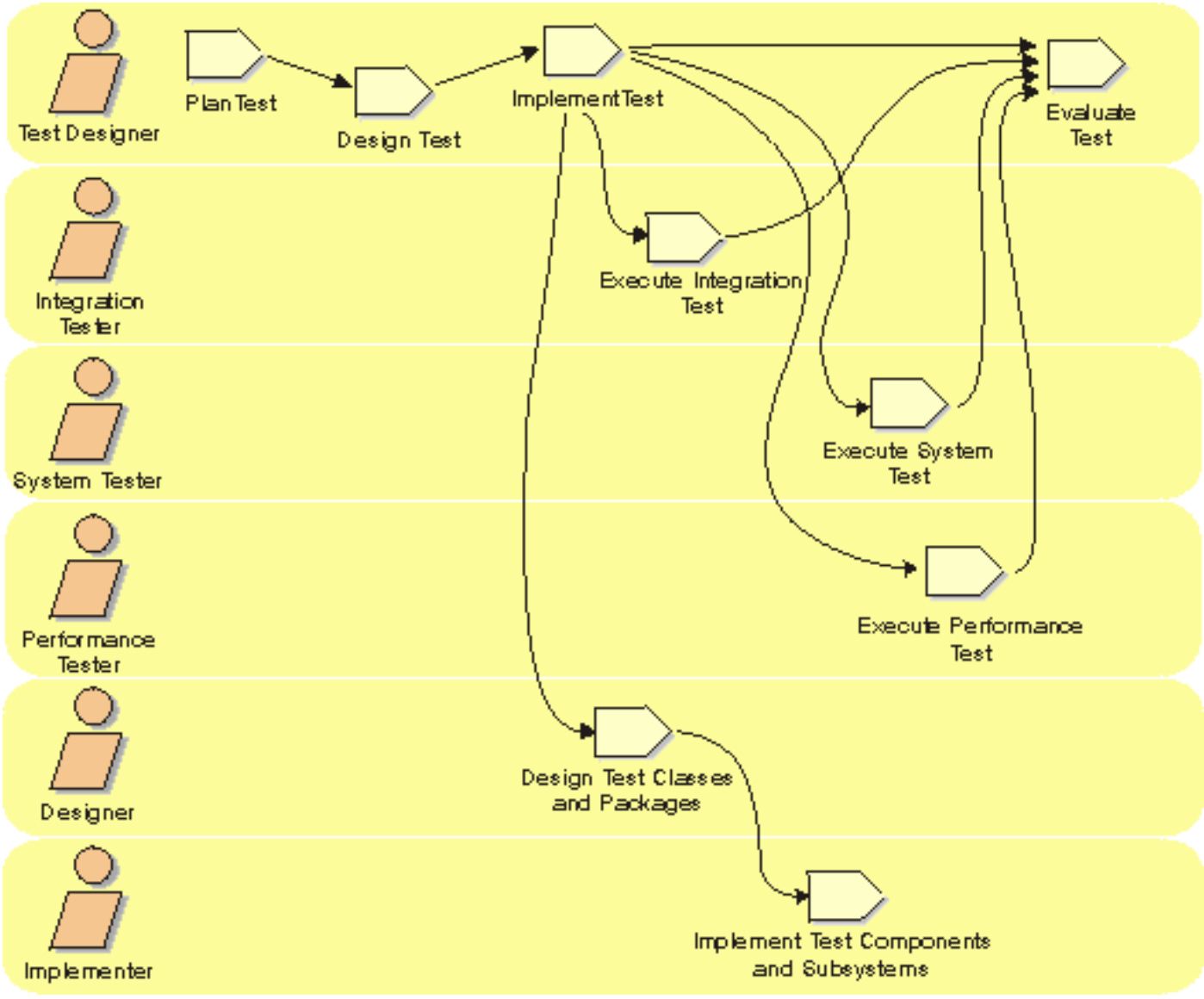
- Test results and progress reports
  - show the results of test procedure execution over a number of iterations and test cycles for the application-under-test

Concepts: Test Strategy

→ Describes the general approach and objectives of the test activities used for test planning

- Subject of test strategy:
  - Testing techniques and tools to be employed
  - What test completion and success criteria are to be used
  - Special considerations affect resource requirements or have schedule implications
    - The testing of interfaces to external systems
    - Simulating physical damage or security threat
- Timing criteria:
  - What iteration you are you in, and what the goals of that iteration are
  - What stage of test (unit test, integration test, system test) you are performing
- Testing kind criteria:
  - Types of test (functional, stress, volume, performance, usability, distribution, ...)
  - Evaluation criteria used (code-based, requirements-based, number of defects, ...)
  - Testing techniques used (manual and automated)

Workflow


Test Designer
Plan Test
Design Test
Implement Test
Evaluate Test
Integration Tester
Execute Integration Test
System Tester
Execute System Test
Performance Tester
Execute Performance Test
Designer
Design Test Classes and Packages
Implementer
Implement Test Components and Subsystems

Plan Test

→ To collect and organize test-planning information

→ To create the test plan

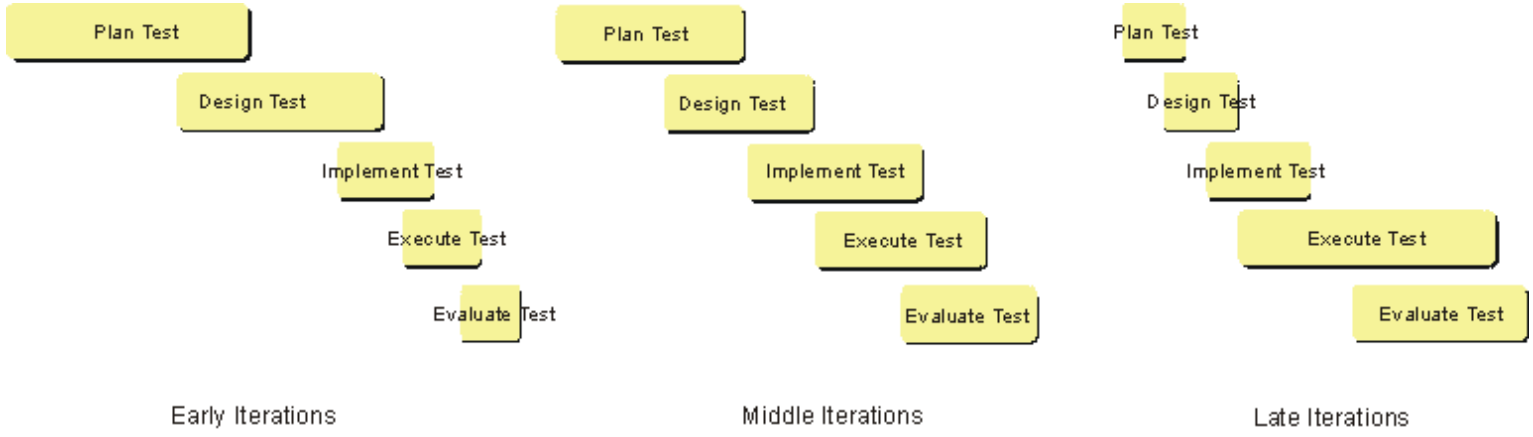
- Identify Requirements for Test
  - → to identify what is being tested and indicate the scope and role of the test effort
  - • review all materials
  - • indicate the requirements for test
- Assess Risk
  - → to maximize test effectiveness and prioritize test efforts
  - → to establish an acceptable test sequence
  - • identify and justify a risk factor
  - • identify and justify an operational profile factor
  - • identify and justify a test priority factor (upon risk factors, operational profile factors, contractual obligations, ...)

Plan Test (cont.)

- Develop Test Strategy
  - → identifies and communicates the test techniques and tools
  - → identifies and communicates the evaluation methods for determining product quality and test completion
  - identify and describe the approach to test
  - identify the criteria for test
  - identify any special considerations for test
- Identify Resources
  - → identify the resources necessary to test, including, human resources (skills, knowledge, availability), hardware, software, tools, etc.
  - identify the skills, knowledge, and number of human resources
  - identify hardware, software, tools, and other non-human resource needs

Plan Test (cont.)

- Create Schedule
  - → identify and communicate test effort, schedule, and milestones
  - estimate test effort
  - generate test schedule



- Generate Test Plan
  - → to organize and communicate to others the test-planning information
  - review / refine existing materials
  - identify test deliverables
  - generate the test plan

Design Test

→ To identify a set of verifiable test cases for each build

→ To identify test procedures that show how the test cases will be realized

- Workload Analysis (for performance testing only)
  - → to identify and describe the different variables that affect system use and performance
  - → to identify the sub-set of use cases to be used for performance testing
- Identify and Describe Test Cases
  - → to identify and describe the test conditions to be used for testing
  - → to identify the specific data necessary for testing
  - → to identify the expected results of test
  - analyze application workflows
  - identify and describe test cases
  - identify test case data

Design Test (cont.)

- Identify and Structure Test Procedures
  - → to analyze use case workflows and test cases to identify test procedures
  - → to identify, in the test model, the relationship(s) between test cases and test procedures creating the test model
  - review application workflows
  - develop the test model
  - structure test procedures
- Review and Assess Test Coverage
  - → to identify and describe the measures of test that will be used to identify the completeness of testing
  - identify test coverage measures
  - generate and distribute test coverage reports

Implement Test

→ To create reusable test scripts

→ To maintain traceability of the test implementation artifacts back to the associated test cases and use cases or requirements for test

- Record or program test scripts
  - → to create appropriate test scripts which implement (and execute) the test cases and test procedures as desired
  - create or acquire test scripts
  - test / debug test scripts
- Identify test-specific functionality in the design and implementation models
  - → to specify the requirements for software functions needed to support the implementation or execution of testing
- Establish external data sets
  - → to create and maintain data, stored externally to the test scripts, that are used by the test scripts during test execution
  - create / maintain external data sets
  - test / debug test scripts

Design Test Packages and Classes

→ To design test-specific functionality

- Identify Test-Specific Packages and Classes
  - → to identify and design the classes and packages that will provide the needed test specific functionality
- Design Interface to Automated Test Tool
  - → to identify the interface necessary for the integration of an automated test tool with test-specific functionality
- Design Test Procedure Behavior
  - → to automate test procedures for which there is no automated test tool available

Implement Test Components and Subsystems

→ To implement test-specific functionality

- Implement and Unit Test Drivers / Stubs
  - → to identify and design the classes and packages that will provide the needed test specific functionality
- Implement and Unit Test Interface to Automated Test Tool
  - → to identify the interface necessary for the integration of an automated test tool with test-specific functionality
- Implement and Unit Test Test Procedure Behavior
  - → to automate test procedures for which there is no automated test tool available

Execute Integration / System / Performance Tests

→ To execute integration/system/performance test

→ To review test results

→ To investigate and organize test results for evaluation

→ To log defects

- Execute Test Procedures
  - → to execute the test procedures (or test scripts if testing is automated)
- Evaluate Execution of Test
  - → to determine whether the tests completed successfully and as desired
  - → to determine if corrective action is required

Execute Integration / System / Performance Tests (cont.)

- Recover From Halted Tests
  - → to determine the appropriate corrective action to recover from a halted test
  - → to correct the problem, recover, and re-execute the tests
- Verify Test Results
  - → to determine if the results are within the expected range
- Investigate Unexpected Results
  - → to identify appropriate action when actual results differ from expected
- Log Defects
  - → to enter defect information into a tracking tool to initiate corrective action
  - → to initiate defect tracking and defect management

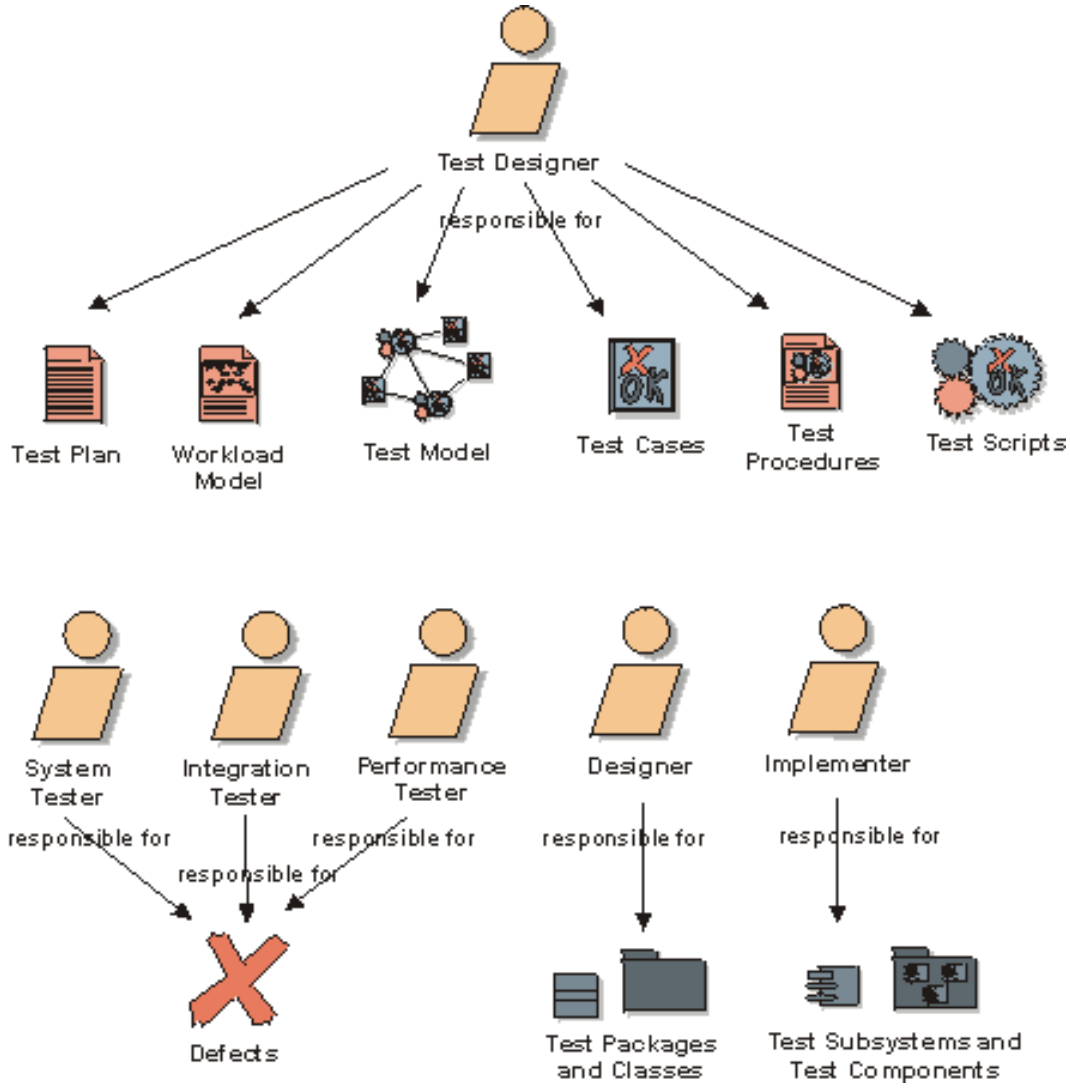
Evaluate Test

→ To deliver quantifiable measures of testing progress

→ To generate a test evaluation report

- Evaluate Test-Case Coverage
  - → to determine if the required (or appropriate) requirements-based test coverage has been achieved
- Evaluate Code Coverage
  - → to determine if the required (or appropriate) code-based test coverage has been achieved
- Analyze Defects
  - → to evaluate the defects and recommend the appropriate follow-on activity
  - → to produce objective reports communicating the results of testing
- Determine if Test Completion and Success Criteria Have Been Achieved
  - → to determine if testing has been executed completely and acceptably
  - → to identify the appropriate follow-on test activity
  - → to produce objective reports communicating the results of testing

Artifacts


Test Designer
responsible for
Test Plan
Workload Model
Test Model
Test Cases
Test Procedures
Test Scripts
System Tester
Integration Tester
Performance Tester
Designer
Implementer
responsible for
responsible for
responsible for
responsible for
responsible for
Defects
Test Packages and Classes
Test Subsystems and Test Components

Testing Guidelines

- Testing Organization
  - Analysts and Domain Experts play the role of Testers
  - Crash tests are executed by “vicious” testers
  - End-users and/or customers are invited for end-iteration testing
  - Customers must be a part of acceptance testing team
- Test Planning
  - Execute more important test cases first
  - Execute crash tests before execution of other types of tests
  - Before execution of acceptance tests run internal testing with the same test cases and following bugs fixing
  - End-iteration testing: test only application parts that are new or may be corrupted since the last testing
  - End-release testing: test the whole application, regardless the previous testing
  - End-release testing: synchronize the installation of new component with iterations
  - End-release testing: if new component is delivered into the testing environment then test the whole application again

Testing Guidelines (cont.)

- Test Planning (cont.)
  - Choose the iteration duration in Transition phase according to estimation of needed time for execution the most important test cases
  - For next release plan the duration of end-release testing longer than for the previous one
  - Schedule the test case executions in accordance to the dependencies and possible execution collisions in test cases
  - Assign similar test cases to one tester (or testing group)
  - Shift attention during testing, if needed
- Test Execution
  - Always let people know
  - Identify obsolete or new test cases, changes to existing test cases while testing
  - End-iteration testing is performed on the development environment
  - End-release testing is performed on a separate testing environment
- Finding Reporting
  - Neatly report the failure context
  - Do not try to localize the bug in the program