# Software Testing

# Fault & Failure

- fault - an abnormal condition or defect at the component
- failure - lack of ability to perform intended function as designed. Failure may be the result of one or many faults.

# Verification & Validation

- verification - checking if the project is compatible with requirements set during the Requirement Definition stage
- validation -final checking; is the project compatible with all requirements (client's requirements as well)

# Audit

- Audit - a phase that evaluates the software according to specification, licenses, contracts, standards, instructions...
- Audit must be performed by someone from the outside

# Testing Strategies

- Bottom – up
- Top – down

# Versions

- Alpha
- Beta
- Gamma

# International Standards

- Over the years a number of types of document have been invented to allow for the control of testing.
- Every organisation develops these documents themselves and gives them different names, thus confuses their purpose

# ISO 9126

- ISO 9126 is an international standard for the evaluation of software. It classifies the areas in a structured manner as follows:
- Functionality
- Reliability
- Usability
- Efficiency
- Maintainability
- Portability

# IEEE 829

- IEEE 829 - Standard for Software Test Documentation. It specifies the form and content of a set of documents for use in eight defined stages of software testing, each stage producing its own separate type of document
  - How the testing will be done
  - Test Design Specification
  - Test Case Specification
  - Test Procedure Specification
  - Test Item Transmittal Report
  - Test Log
  - Test Incident Report
  - Test Summary Report

# Black & White

- White box testing:
  - testing internal mechanisms of the system
  - done by programmers and expirienced staff
- Black box testing:
  - testing the system in its envioroment
  - done by anyone, usually a team of hired specialists

# Testing Team

- Manager
- Secretary
- members: users, development manager, development engineers, programing librarian, quality assurance, independent verification and testing team, independent specialists

# Testing activities

- Unit testing:
  - break the code in several small units and test each one individually
  - encourages making changes, simplifies integration
  - doesn't test system-wide or integration errors
  - examples: Junit, PyUnit, Nunit
- Integration testing:
  - tests how well  modules fit eachother

# Testing activities continued

- Integration testing cont.:
  - testing groups of units as black boxes
- System testing:
  - testing the whole system as a black box
- Regression testing:
  - regression bugs: a functionality that previously worked stops working
  - re-run old tests that uncovered a bug before

# Testing activities continued

- Load testing:
  - testing the system by simulating multiple users accessing the program's services concurrently
- Performance testing:
  - how fast the system works under a particular load
- Stress testing:
  - testing how system performs beyond its normal operational capacity

# Testing activities continued

- Installation testing:
  - testing the system outside of the development enviorment
- Stability testing:
  - testing if an application will crash
- Usability testing:
  - measuring how well people can use the system
  - usually for testing user interfaces

# Testing activities final

- Conformance testing:
  - determining whether a system meets some specified standard
- User acceptance testing:
  - testing before a new system is accepted by the customer
  - one of the final testing phases (before gamma testing)

# Formal verification

- proving or disproving the correctness of the system using formal methods
- system is regarded as: FSM, LTS, automaton, digital circuit, etc...
- usually formal verification is carried out algorithmically
- automatic thorem provers

# Automated testing

- done using special testing scripts
- test cases are rarely generated automatically using model-based testing
- can be used to test simple an even GUI based applications
- test output is compared with the expected one if possible

# Code coverage

- measures: Statement Coverage, Condition Coverage, Path Coverage
- full path coverage is usually impractical or impossible (eg. could solve the “halting” problem)
- usually some tests are run to achieve a certain precantage of code coverage (eg. “the tests gave us 56% of code coverage”)

# Error seeding

- A certain ammount of faults are delibertly inserted into the code and the resulting failures are measured for various reasons
- eg. $FU = FG \cdot (FE / FEG)$
  - FU undetected errors
  - FG non seeded errors detected
  - FE seeded errors
  - FEG seeded errors detected

# Reliablity measure

- Observing the amount of failures during a statistical test help us figure out the reliability of the application. There are 4 measurements that make the failure rate:
  - probability of the transaction failure
  - frequency of failure
  - time average between the failure
  - accessibility

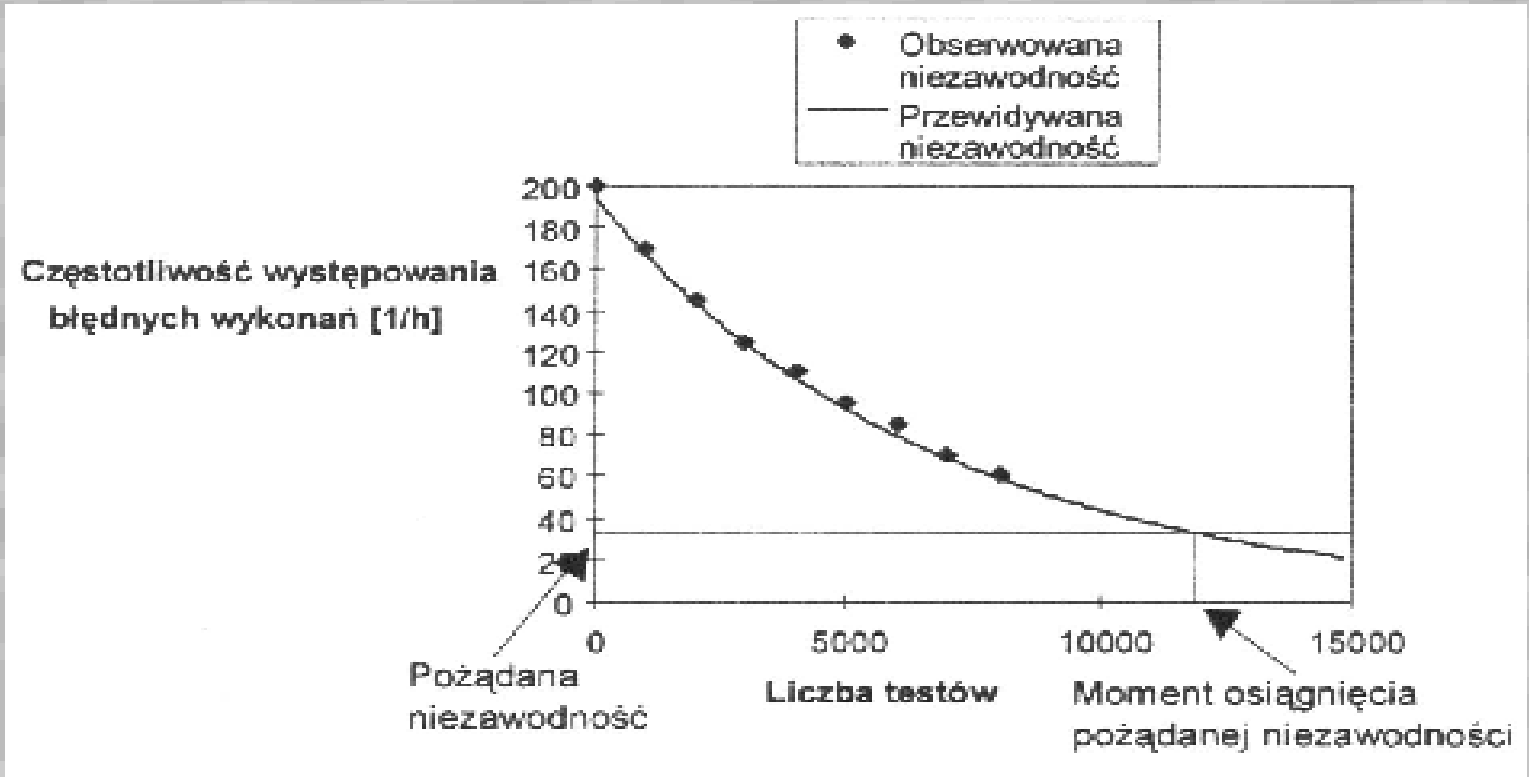
# Failure rate

- The failure rate is very important for the client because:
  - frequency of a failure has a great influence on the value of maintaining the software
  - failure let us estimate service costs
  - knowledge about failure rates let us figure out and make better creating processes to decrease the maintainance costs

# Increasing the reliability

- When a failure was found and removed and no new errors was made, this is called “increasing the reliability”
- Reliability equation:
  - Reliability = Initial Reliability * exp ( -C * Amount of tests )
- C factor depends on the concrete system. It may be figured by observing statistics failure of the system

# Increasing the reliability




·The best way to increase the reliability is to choose the right testing data - not randomly, but checking all possibilities

# Priorities

- But testing all the possible data is impossible. We need to choose the best combinations. When we choose testing data we must remember about:
  - The possibility to execute the function is more important than the quality
  - The functions of the previous system are more important than the new one
  - The typical situation is more important than all others

# System security

- Security not always means Reliability
- The illusive system may be secured, if the faults are not dangerous
- The most important is that the system is secured even when the exception was occurred or during hardware failure

# Increasing the security

- more attention about security while implementing the software
- more important modules should be realized by more experienced people
- testing the software must be very carefully

# Success factor of the testing process

- making the testing team more important
- the right motivation for testing – eg. awards for the best testers

# Result of the testing process

- Correct code, project, model and requirement
- The report of the testing process include the information about each test
- Evaluation about software failure and the maintaining cost