Testing Software Systems

Overview

Testing OO Software Systems

- Introduction to Software Testing

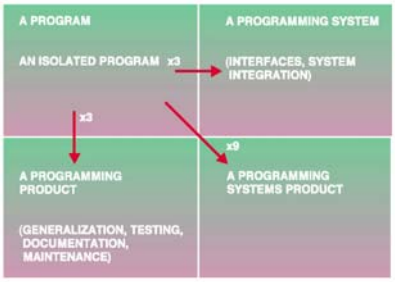
References

- **B. Hailpern, P. Santhanam, *Software debugging, testing, and verification*, IBM Systems Journal, Vol. 40, No1. 2002, http://www.research.ibm.com/journal/sj/411/hailpern.html**
- ***Skipt zur Vorlesung Softwaretechnik II*, FB Informatik, Universitaet Oldenburg, WS 2000/2001**
- **Steve McConnell, *Software Quality at Top Speed*, Software Development, August 1996, http://www.construx.com/stevemcc/articles/**
- **Bernd Kahlbrandt, *Skript Sofwtare Enginering II*, http://www.informatik.fh-hamburg.de/~khb/st4se2/st4se2.html**
- **Robert v. Binder, *Testing OO Systems,* Addison Wesley, 2000**

Evolution of Programming Systems

- **There is a big difference between an isolated program created by a lone programmer and a programming systems product.**
  - A **programming systems** product "can be run, tested, repaired, and extended by anybody …in many operating environments, for many sets of data" and forms a part of "a collection of interacting programs, coordinated in function and disciplined in format, so that the assemblage constitutes an entire facility for large tasks." (Frederic Brooks)

Figure 1 Evolution of a programming systems product



F. P. Brooks, *The Mythical Man-Month*, p.5, Figure 1-1. © 1995, 1975 Addison Wesley Longman, Inc. Reproduced by permission of the author and Pearson Education, Inc.

3

Testing is essential

- **Objective:**
  - Develop quality software at low costs to maximize margin.
- **Strategy**
  - Minimize costs by reducing testing effort.
  - However, consequences can be severe.
- **Result:**
  - Software for power plants (Tschernobyl?),
  - Software for military (747 shoot down),
  - Software for aviation (Airbus crash),
  - Software for production control (Seveso),
  - 2000-Problem (Y2K)

4

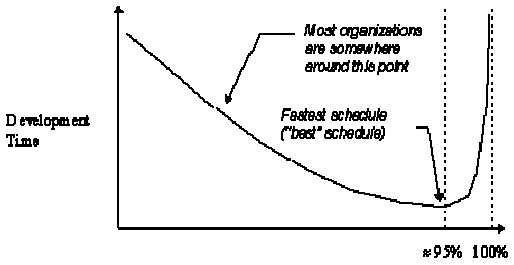
Software Systems

- **Software quality has improved, however software systems grew at faster speed**
  - **1977:** 7 - 20 defects per 1000 LOC
  - **1994:** 0.2 - 0.05 defects per 1000 LOC
- **SAP R/3**
  - 7.000.000 LOC, 100.000 function calls, 20.000 functions, 17.000 menus, 21.000 reports
- **Space Shuttle**
  - 48.000.000 LOC
- **Windows 2000**
  - Ca. 50.000.000 LOC
  - Estimated no. of defects: 60.000 → 1,2 defects per 1000 LOC

5

Software Quality at Top Speed

- **Software quality**
  - Some project managers try to shorten their schedules by reducing the time spent on quality-assurance practices such as design and code reviews.
  - Some shortchange the upstream activities of requirements analysis and design.
  - Others--running late--try to make up time by compressing the testing schedule, which is vulnerable to reduction since it's the critical-path item at the end of the schedule.
- **In software, higher quality (in the form of lower defect rates) and reduced development time go hand in hand.**



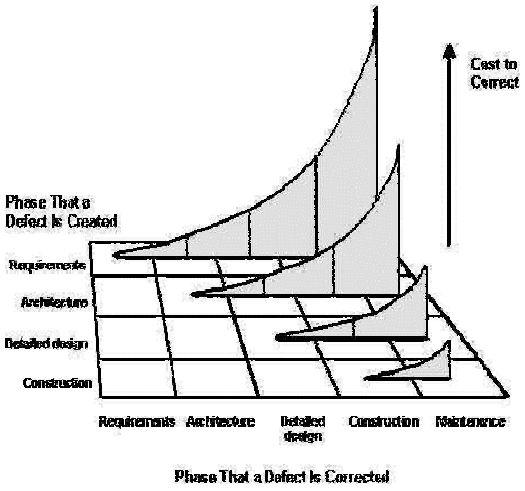
Percentage of Defects Removal Before Release

Source:http://www.construx.com/stevemcc/articles/art04.htm

6

Quality Assurance and Development Speed

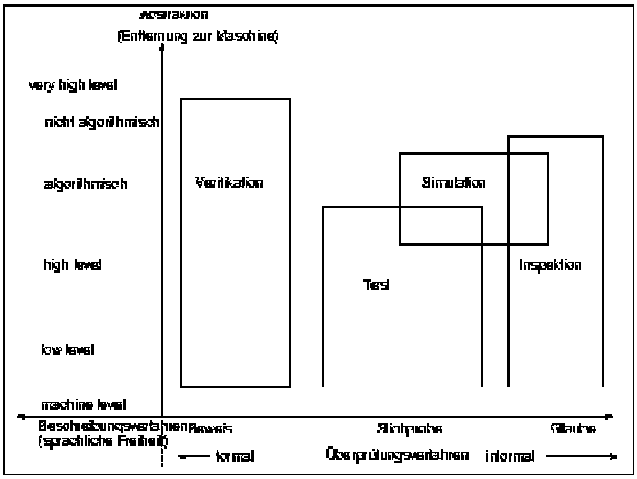
Studies have found that reworking defective requirements, design, and code typically consumes 40 to 50 percent of the total cost of software development.



Source:http://www.construx.com/stevemcc/articles/art04.htm

7

Validation Strategies

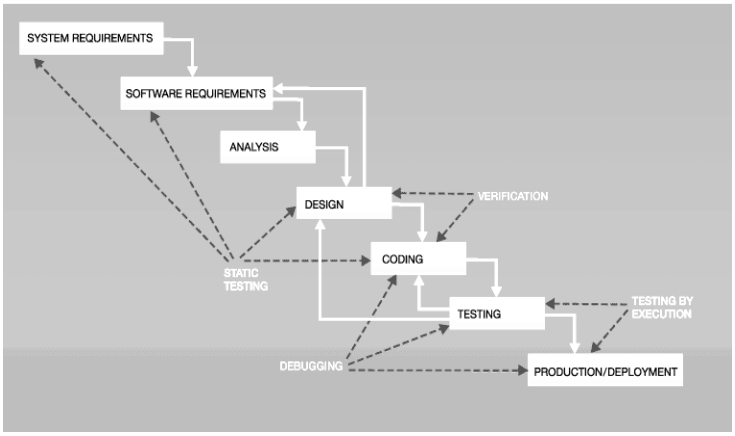


Source: Bernd Kahlbrandt, Skript zu Software Engineering

8

Activities in a Typical Software Development Process

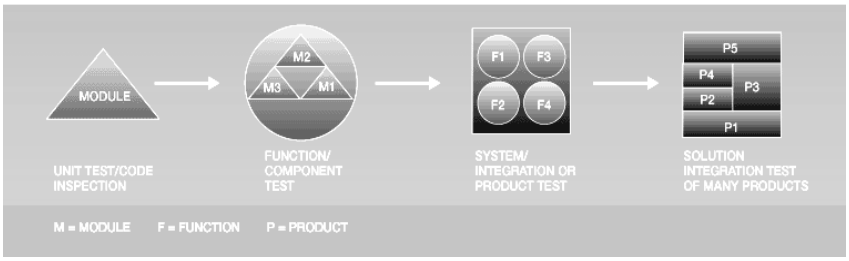
Figure 2 The activities that involve debugging, testing, and verification in a typical software development process



9

Typical Stages of Testing

Figure 3 Typical stages of testing within IBM



- **Testing is a necessary area for software validation.**
  - External Function Tests based on external specification
  - System Tests: recovery, stress, performance, hw + sw configuration
  - Integration / Production Tests based on customer acceptance criteria

10

Testing I

„Program testing can be used to show the presence of bugs, but never to show their absence” Dijkstra

- **From his point of view, any amount of testing represents only a small sampling of all possible computations and is therefore never adequate to assure the expected behavior of the program under all possible conditions.**

11

Testing II

- **Test Metric**
  - Testing is a sampling of the program execution space. Consequently, the natural question arises: when do we stop testing?
    - defect discovery rate over time
    - test progress over time (planned, attempted, actual)
    - percentage of test cases attempted, etc...
- **Static Testing**
  - can be done before an executable version of the program exists
  - Typical analyses performed will involve a compiler or parser, tied to the language of the program, that builds a representation, such as
    - a call graph,
    - a control graph, or
    - a data flow graph, of the program.

12

Testing III

- **Test Automation**
  - There are four major parts to any testing effort:
    - test case design,
    - test case creation,
    - test case execution, and
    - debugging
  - Automation of test execution
  - Automation of test case design (and hence test case creation)
    - Need of formal description of the specifications of the software behavior, resulting in a model of the software behavior.
- **Regression Testing**
  - Regression testing not only checks that earlier specifications are still valid, but also catches backward-compatibility problems.

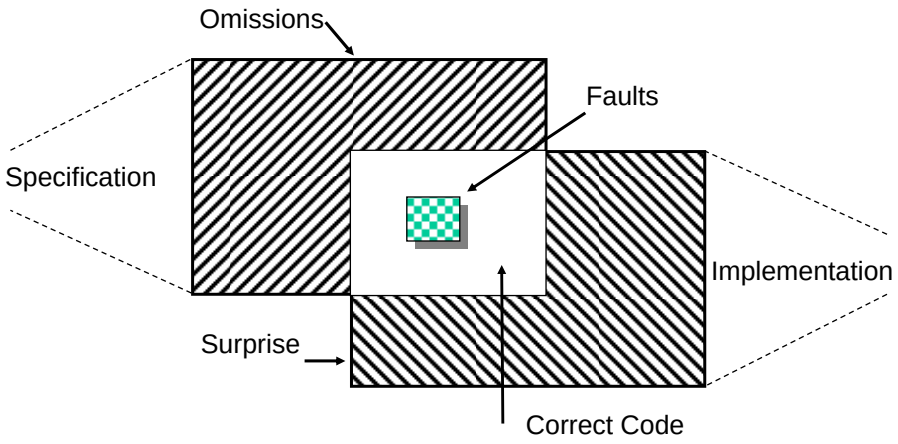
13

Defintions V

- **Test Point**
  - A test point is a specific value for test case input and state variable
- **Test Case**
  - A test case specifies the pretest state of the Implementation Under Test (IUT) and its environment, the test inputs or conditions and the expected results.
- **Test Suite**
  - A test suite is a collection of test cases, typically related by a testing goal or implementation dependency.
- **Test driver**
  - A test driver is a class or utility program that applies test cases to an IUT.
- **Test Run**
  - A test run is the execution (with results) of a test suite.
- **Test harness**
  - A test harness is a system of test drivers and other tools to support test execution.

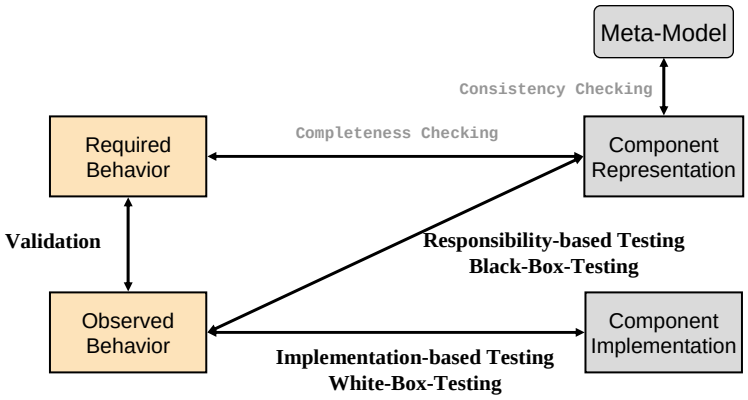
14

Faults, Omissions, and Surprises



15

Modelling – Testing - Validation



16

Black-Box Testing I

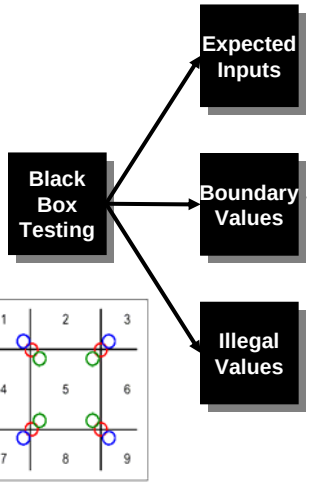
- **Black–Box Testing**
  - Also known as *functional testing*. A software testing technique whereby the internal workings of the item being tested are not known by the tester.
  - For example, in a black box test on a software design the tester only knows the inputs and what the expected outcomes should be and not how the program arrives at those outputs.
  - The tester does not ever examine the programming code and does not need any further knowledge of the program other than its specifications.

17

Black-Box Testing II

- When you perform black box testing you should execute your procedure with test cases from each of the following categories:
  - Expected inputs
    These include the values that you expect your procedure to receive most of the time.
  - Boundary values
    If your procedure expects an input value from 1 to 999, use 1 and 999 as test cases to make sure that your procedure returns the expected results for the boundary cases.
  - Illegal values
    Using the boundary values example, what happens if your procedure receives as input a value that is less than 1 or greater than 999?
    - Does the user receive a useful error message?
    - Does the procedure simply stop, or does it attempt to use values outside its limitations and simply return an incorrect answer?

    It is essential that you run the procedure using illegal input values to determine the answers to these questions.



18

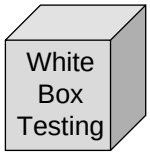
Black-Box Testing

- **The** advantages **of this type of testing include:**
  - The test is unbiased because the designer and the tester are independent of each other.
  - The tester does not need knowledge of any specific programming languages.
  - The test is done from the point of view of the user, not the designer.
  - Test cases can be designed as soon as the specifications are complete.
- **The** disadvantages **of this type of testing include:**
  - The test can be redundant if the software designer has already run a test case.
  - The test cases are difficult to design.
  - Testing every possible input stream is unrealistic because it would take a inordinate amount of time; therefore, many program paths will go untested.

19

White-Box Testing I

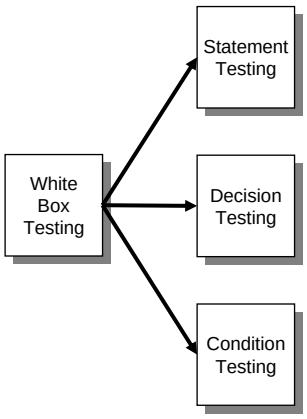
- **White-Box Testing**
  - A software testing technique whereby explicit knowledge of the internal workings of the item being tested are used to select the test data.
  - Unlike black box testing, white box testing uses specific knowledge of programming code to examine outputs. The test is accurate only if the tester knows what the program is supposed to do. He or she can then see if the program diverges from its intended goal.
  - White box testing does not account for errors caused by omission, and all visible code must also be readable.



20

White-Box Testing II

- **To perform white box testing, do the following:**
  - Test each statement.
    - You provide sets of test values to ensure that every statement in the procedure is executed at least once. This includes all statements --- even those executed only when optional arguments, user-supplied arguments, subroutines, user-action routines, or specific error codes are present.
  - Test each decision.
    - You provide test cases to ensure that each branch of a decision is executed at least once. In the case of a standard Boolean decision, this typically requires providing two values; however, this number may be much greater in the case of compound or nested decisions.
  - Test each condition.
    - Condition testing requires writing test cases that ensure each condition in a decision takes all possible outcomes at least once, and each point of entry to the program or subroutine is invoked at least once. You must supply multiple test values in cases of compound and nested loops. In testing the entry points, remember to invoke any optional routines (either internal or external), as well as error handlers. If your procedure contains a JSB entry point, that entry point should also be tested.



21

Result-Oriented Testing

- **Hybrid approach to testing:**
  - **Result Oriented Testing** orchestrates test techniques for effectiveness. Instead of being cast as compatible opposites or narrow technical specialties, scope-specific test design and execution techniques are organized into a coherent whole.

Binder

„It doesn‘t matter whether a cat is black or white so long as it catches mice.“

Deng Xiaoping

22