Software Testing

Lesson 1
Introduction

Uwe Gühl

Winter 2015 / 2016

Contents

- Introduction
  - Why is Testing Necessary?
  - What is Testing?
  - Seven Testing Principles

Introduction

Rumours ...

- Testing is not sexy
- If projects fail, testing is the reason
- In Europe in ancient times bearer of bad news (sometimes) got killed
- Following legends, even bearer of good news died ...

Lowlands of Marathon

(Image source: Adam Carr, http://en.wikipedia.org/wiki/Image:Ac.marathon.jpg GNU Free Documentation License)

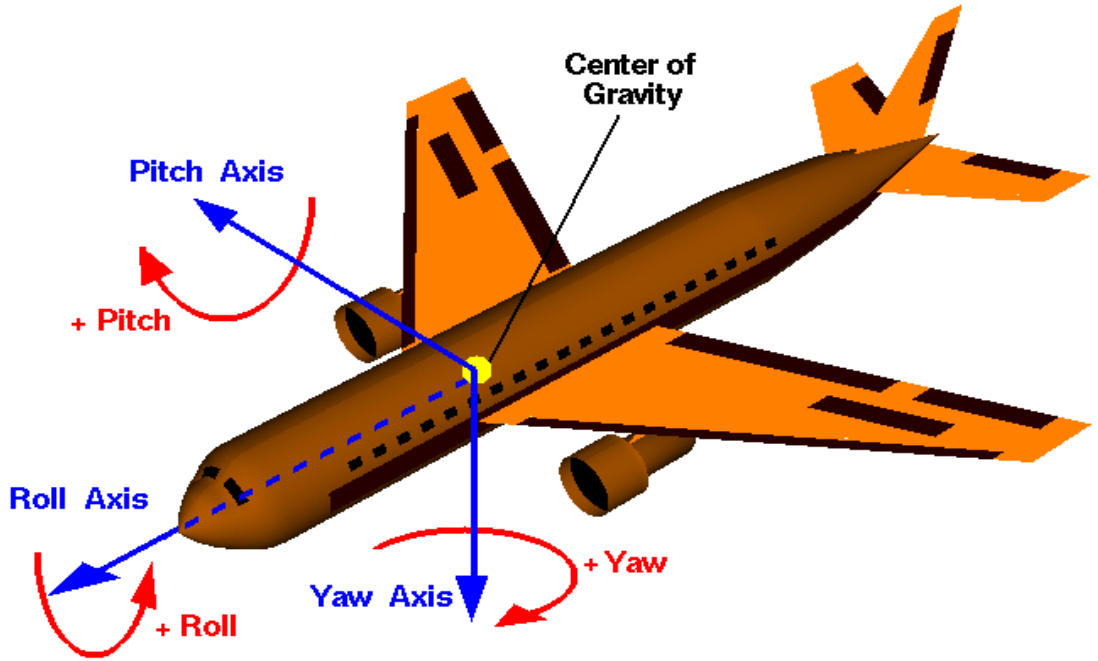
Why is Testing Necessary? (Fatal) software defects

- **Mars Climate Orbiter Loss**, September 1999
  At 2 am on September 23 1999, 5 minutes before it was due to go behind the planet, the Mars Climate Orbiter fired it's main engine to go into orbit around Mars.
  No signal was detected from the spacecraft when it was due to come out from behind the planets shadow.
  The plan was for the spacecraft to orbit at an altitude of 153 kilometres, which was far above the minimum survivable altitude of 85 kilometres However the last six to eight hours of data indicate the approach altitude was much lower at just 60 kilometres.
  So the question needing to be asked was why did the spacecraft approach so low?

- Reason:
  The likely cause of the problem related to the transfer of information between the modules of code written by 2 groups, the Mars Climate Orbiter spacecraft team in Colorado and the mission navigation team in California.
  It seems **that one team used English units** (e.g., inches, feet and pounds) while **the other used metric units** and there seems to have been no conversion between the two.

Why is Testing Necessary? (Fatal) software defects

- In 1982 there was a crash of a Lockheed F-117A Night Hawk during takeoff.
- Reason:
  The fly-by-wire system had been hooked up incorrectly ("yaw rudder" was used instead of "pitch elevator" and visa versa)



(Image source: NASA, http://en.wikipedia.org/wiki/File:Rollpitchyawplain.png Public domain)

Why is Testing Necessary? (Fatal) software defects

- In September 1994 three parking offender in Bayreuth (Germany) got a charge "Preparation of a war of aggression”
- Reason:
  Mistaken code

Why is Testing Necessary? (Fatal) software defects

- In 1985 all black cars left an assembly hall of General Motors without a windscreen.
- Reason:
  A robot in the assembly hall did not recognize the colour of black cars.

Why is Testing Necessary? (Fatal) software defects

- In an hospital therapy planning software miscalculates the proper dosage of radiation for patients undergoing radiation therapy.
  The software allows a radiation therapist to draw on a computer screen the placement of metal shields called "blocks" designed to protect healthy tissue from the radiation. But the software will only allow technicians to use four shielding blocks, and the doctors wish to use five.
  The doctors discover that they can trick the software by drawing all five blocks as a single large block with a hole in the middle. What the doctors don't realize is that the software gives different answers in this configuration depending on how the hole is drawn:
  - Draw it in one direction and the correct dose is calculated,
  - Draw in another direction and the software recommends twice the necessary exposure.
- At least eight patients die, while another 20 receive overdoses likely to cause significant health problems.

Why is Testing Necessary? (Fatal) software defects

- 1996 a prototype of the Ariane 5 rocket of the European Space Agency was destroyed one minute after the start.
- Reason:
  The code of the Ariane 4 was used.

Why is Testing Necessary? (Fatal) software defects

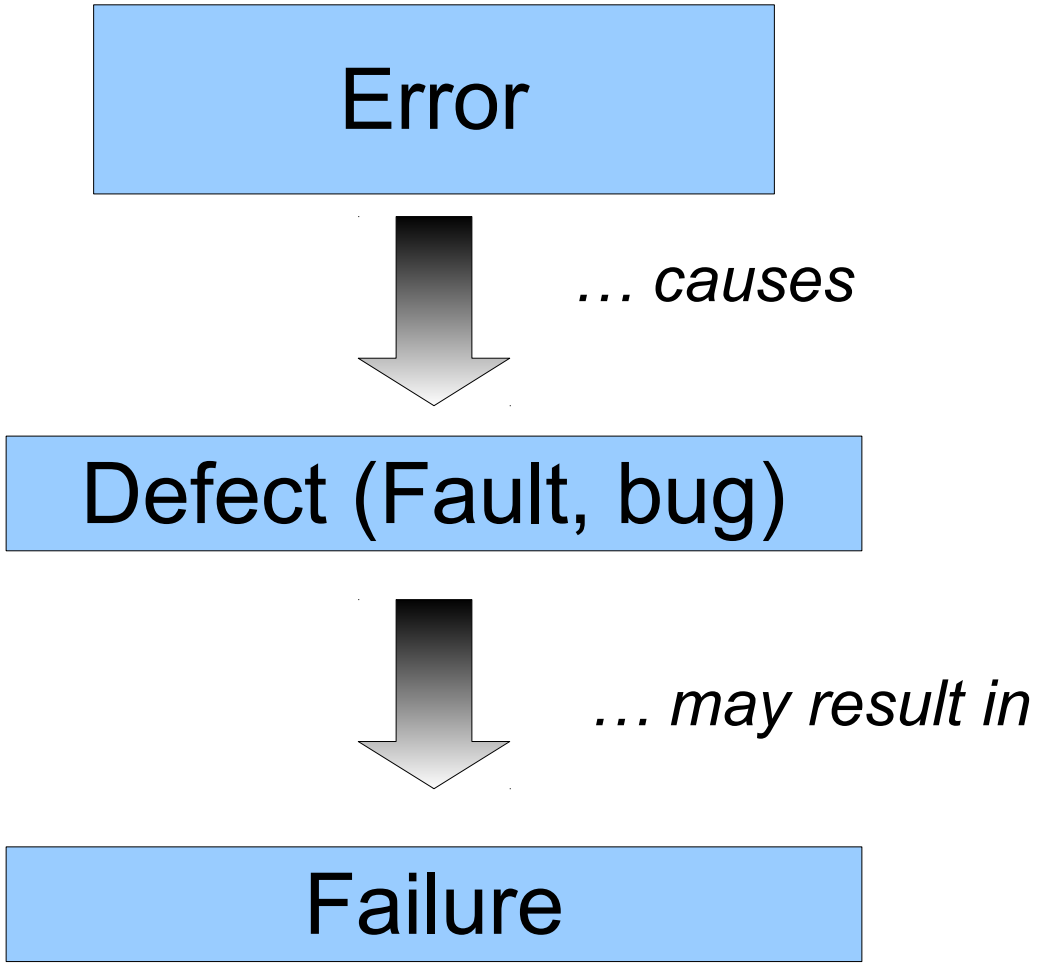
"The most expensive hyphen in history"

- 1962 the NASA lost their Venus-spacecraft Mariner 1, and so about 80 Million US-Dollar
- Reason:
  Because of a software bug caused by a missing superscript bar in $\dot{\bar{r}}_n$ in the specification

Why is Testing Necessary? (Fatal) software defects

- In Excel 2007 was a calculation defect, leading to many wrong spread sheets and accounts.
- Reason:
  In multiplication, where the result would have been 65,535, Excel calculated always 100,000

Why is Testing Necessary? Causes of Software Defects



A human action that produces an incorrect result. [After IEEE 610]

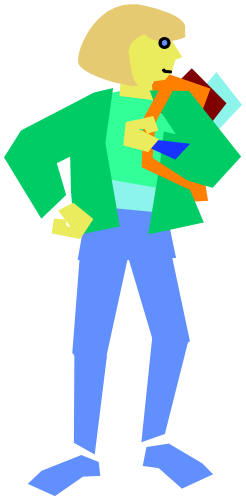
A flaw in a component or system that can cause the component or system to fail.

Deviation of the component or system from its expected delivery, service or result. [After Fenton]

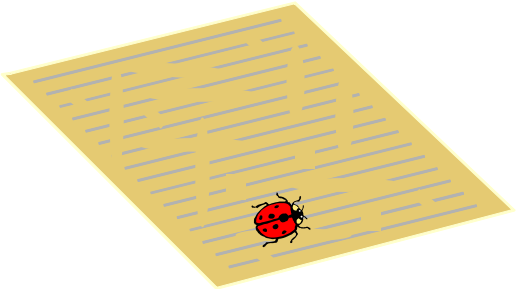
Why is Testing Necessary? Causes of Software Defects

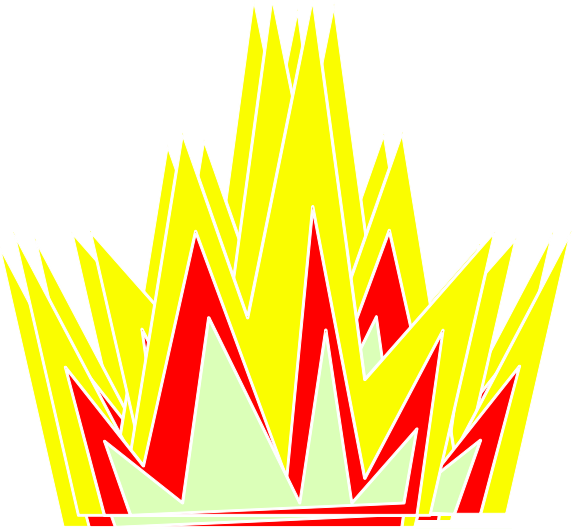
Error - Fault - Failure

A person makes an error ...



… that creates a fault in the software ...





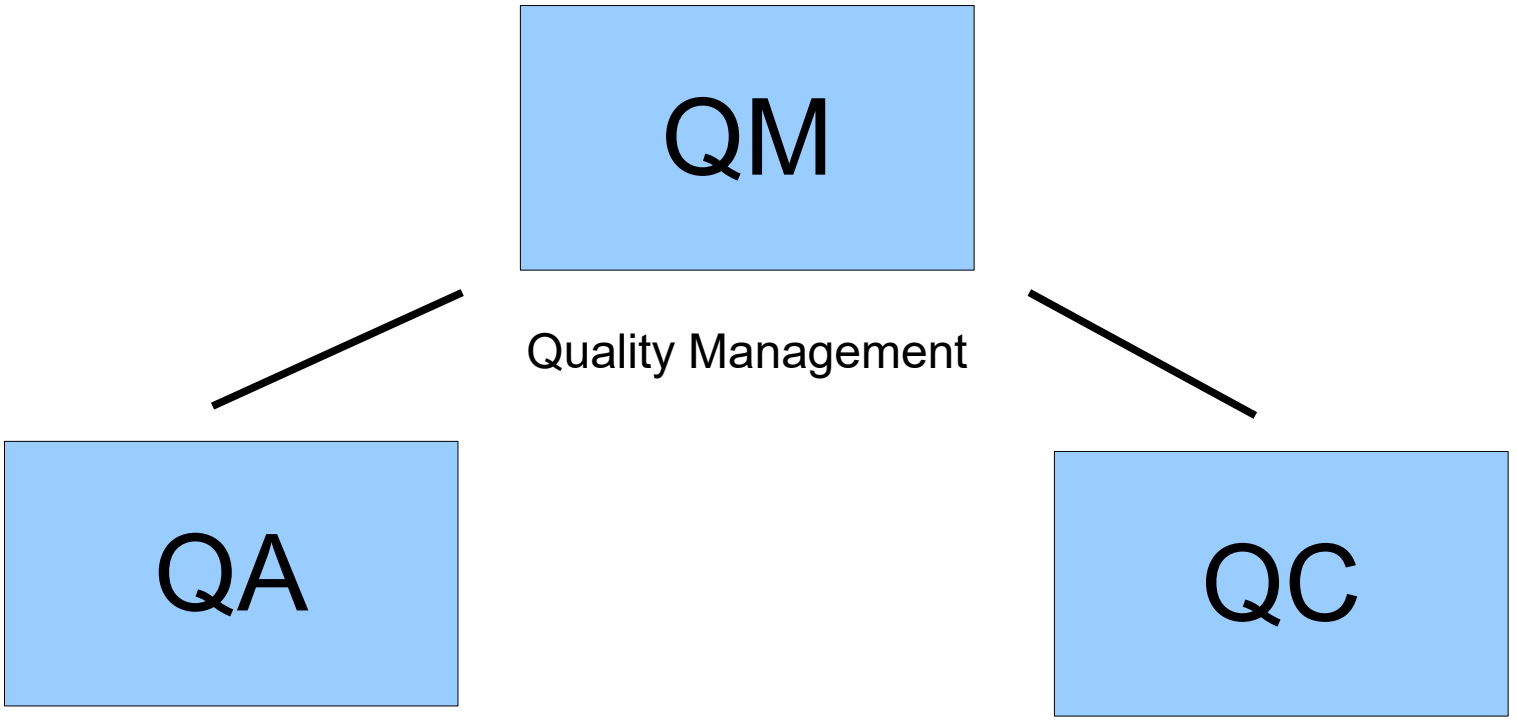
… that can cause a failure in operation

http://www.softwaretestinggenius.com

Why is Testing Necessary? Role of testing

- Testing of systems and documentation can
  - help to reduce the risk of problems occurring during operation
  - contribute to the quality of the software system, if the defects found are corrected before the system is released for operational use.
- Software testing may also be required to meet
  - contractual requirements,
  - legal requirements, or
  - industry-specific standards.

Why is Testing Necessary? Testing and Quality



Quality Assurance

Are we building the product right?

Prevention of faults
by inspecting and testing the **process**

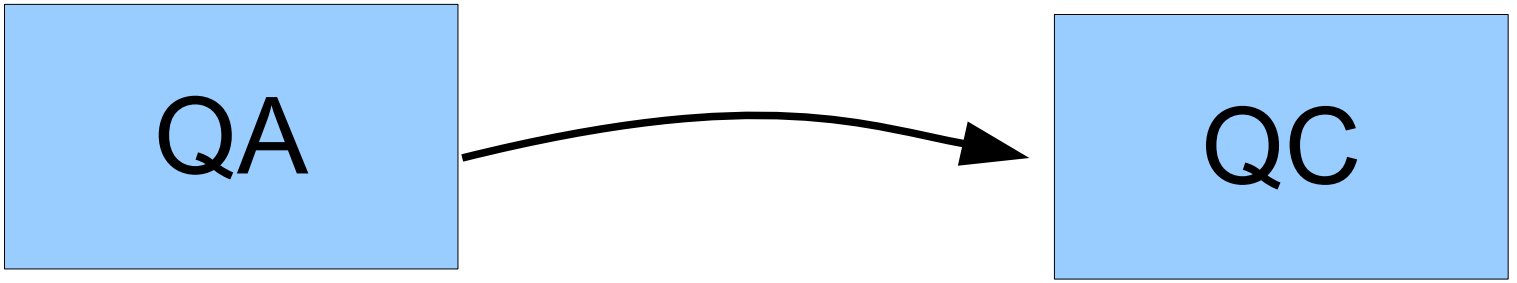
Quality Control

Are we building the right product?

Detection of faults
by inspecting and testing the **product**

Why is Testing Necessary? Testing and Quality

- Relationship QA – QC
  As QA inspects the processes, it investigates in test processes as well, test process improvements e. g. with TPI [Sog16] or TMMI [TMMI16]



Quality Assurance

Quality Control

Examples for test processes and test work products

- Defect Management Process
- Test Case Creation Process
- Test Cases
- Test Reports

Why is Testing Necessary? Testing and Quality

- Measuring the quality of systems
  - Number of defects
  - Characteristics, e.g. following IEC/ISO 9126 (functionality, reliability, usability, efficiency, maintainability and portability)
- Good designed test that passes reduces risk in a system.
- Quality of the software system increases, if defects found by testing get fixed.
- Testing should be part of quality assurance (like standards, training, defect analysis).

Why is Testing Necessary? How much testing is enough?

- Deciding how much testing is enough should take account of
  - the level of risk, including
    - technical,
    - safety, and
    - business risks,
  - project constraints such as time and budget.
- Testing should provide sufficient information to stakeholders to make informed decisions:
  - Release of the software could be delivered? … to next development step or to customer

What is Testing? Thoughts

- How many testers does it take to change a light bulb?
  - None.
  - Testers just noticed that the room was dark.
    Testers don’t fix the problems, they just find them.

- Testing is not accurate science!

What is Testing? Definitions

- The British Standards Institution, in their standard BS7925-1 from 1998, define testing as “the process of exercising software to verify that it satisfies specified requirements and to detect faults; the measurement of software quality” [STW07]
- The IEEE* offers a couple of standards:
  - IEEE 1008 – "IEEE Standard for Software Unit Testing"
  - IEEE 610 – "IEEE Standard Glossary of Software Engineering Terminology"
  - IEEE 829 – "IEEE Standard for Software Test Documentation"

* Institute of Electrical and Electronic Engineers

What is Testing? Definitions

- "Testing is the process of establishing confidence that a program or system does what it is supposed to." (Hetzel, 1973)
- "Testing is demonstrating that a system is fit for purpose." (Evans et al. 1996)
- "Testing is the process of executing a program or system with the intent of finding errors." (Myers, 1979)
- "Testing is the process consisting of all life cycle activities concerned with checking software and software-related work products." (Gelperin and Hetzel, 1988)

What is Testing? Statements

- “Program testing can be used to show the presence of bugs, but never to show their absence!” (Dijkstra 1969)
- “In most cases ‘what’ you test in a system is much more important than ‘how much’ you test” (Craig 2002)
- “Prioritise tests so that, when ever you stop testing, you have done the best testing in the time available” (ISEB testing foundation course material 2003)

What is Testing? Goals

- Goal of Testing is to establish a base for the acceptance of the software by the customer based on the specification through
  1. High test coverage
  2. No / Low number of non critical defects left
     → There should be no critical defect
     → Depends on acceptance criteria defined in advance
  3. Statements concerning software quality

What is Testing?
Goals

1. High test coverage

- Completeness
  Ensure all requirements are implemented
  → Total scope must be tested at least once (of course hopefully successful, means test passed)
- Critical scope
  Ensure that critical requirements are implemented and work fine
  → All high prioritized requirements must be tested successfully, means test passed

What is Testing? Goals

2. No / Low number of defects left
   - At the end the final version of the application
     - ✗ should have no critical defects any more
     - ✗ should have only a small number of tolerable defects
   - Demand on testing is therefore, to detect as much critical defects as soon as possible – idea is to fix them during the testing cycles
   - The acceptance criteria should determine, what the customer expects. A contract could content acceptance criteria: How many defects with which severity are finally acceptable? Necessary: Definition of criteria for
     - ✗ Severity Level
     - ✗ Priority Level

What is Testing?
Goals

3. Statements concerning software quality

- Is it possible to install the software?
- Is it possible to operate the software, is it compatible?
- Fulfils the software the expected functionality?
- Do the interfaces work?
- Is it possible to use the software optimal (Software ergonomics, usability, end user needs)
- Does the software run steadily, with high performance, fail proof?
- Fulfils the software special cultural features (Multilingualism, English / metric system, weight units)?
- Is the software safe / secure?

What is Testing? Debugging and testing

- Debugging
  - Development activity that finds, analyses and removes the cause of the failure.
  - Responsible: Developer
- Testing
  - Testing can show failures that are caused by defects.
  - Responsible: Tester

Seven Testing Principles
Principle 1: Presence of defects

- Testing can show that defects are present, but cannot prove that there are no defects.
- Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.

Seven Testing Principles
Principle 2: No exhaustive testing

- Exhaustive testing is impossible
- Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases.
- Instead of exhaustive testing, risk analysis and priorities should be used to focus testing efforts.

Seven Testing Principles
Principle 2: No exhaustive testing

Task:

- Testing of a simple program with three integers, up to 16 Bit
- Every combination should be tested
- Duration with assumption 100.000 tests / second

Solution:

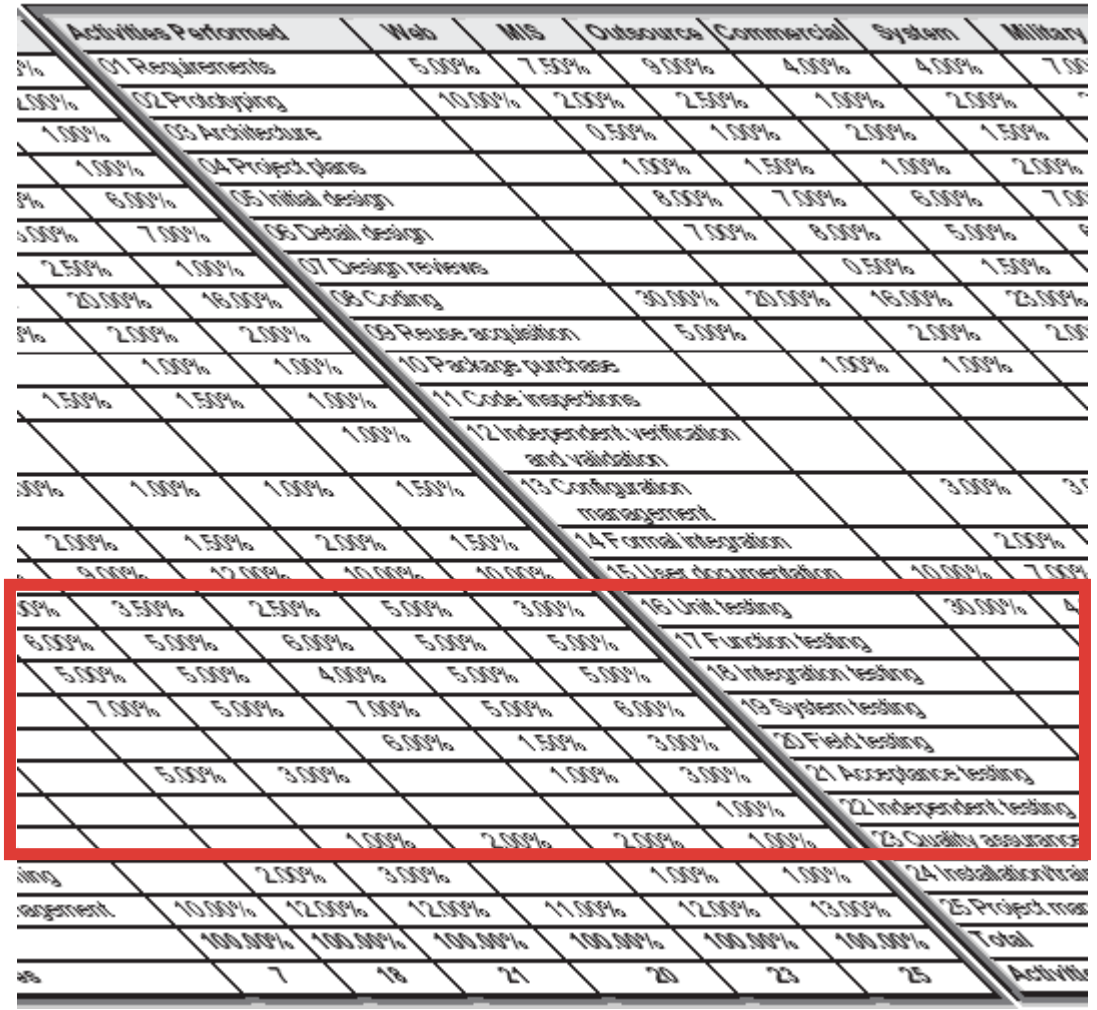
- $2^{16} * 2^{16} * 2^{16} = 2^{48}$ combinations
  = 281.474.976.710.656 combinations
- Duration: About 90 years

Seven Testing Principles
Principle 3 – Early testing

- To find defects early ...
  ⇨ start testing activities as early as possible in the software or system development life cycle,
  ⇨ focus on defined objectives.

Seven Testing Principles
Principle 3 – Early testing

- Costs for testing *Software Development Activities – percentage of work effort by activities concerning test:* 22.5 % up to 30 % [Jon05]



Activities Performed	Web	MIS	Outsource	Commercial	System	Military
01 Requirements	5.00%	7.50%	9.00%	4.00%	4.00%	7.00%
02 Prototyping	10.00%	2.00%	2.50%	1.00%	2.00%	
03 Architecture			0.50%	1.00%	2.00%	1.50%
04 Project plans			1.00%	1.50%	1.00%	2.00%
05 Initial design			8.00%	7.00%	6.00%	7.00%
06 Detail design			7.00%	8.00%	5.00%	
07 Design reviews					0.50%	1.50%
08 Coding			30.00%	20.00%	16.00%	23.00%
09 Reuse acquisition			5.00%		2.00%	
10 Package purchase					1.00%	1.00%
11 Code inspections						
12 Independent verification and validation						
13 Configuration management					3.00%	
14 Formal integration						2.00%
15 User documentation					10.00%	7.00%
16 Unit testing	3.50%	2.50%	5.00%	3.00%	30.00%	
17 Function testing	6.00%	5.00%	6.00%	5.00%	5.00%	
18 Integration testing	5.00%	5.00%	4.00%	5.00%	5.00%	
19 System testing	7.00%	5.00%	7.00%	5.00%	6.00%	
20 Field testing			6.00%	1.50%	3.00%	
21 Acceptance testing		5.00%	3.00%	1.00%	3.00%	
22 Independent testing					1.00%	
23 Quality assurance			1.00%	2.00%	2.00%	1.00%
24 Installation/training			2.00%	3.00%		1.00%
25 Project management	10.00%	12.00%	12.00%	11.00%	12.00%	13.00%
Total	100.00%	100.00%	100.00%	100.00%	100.00%	100.00%
Activities	7	18	21	20	23	25

Seven Testing Principles
Principle 3 – Early testing

- Costs for defects
  - Based on Elfriede Dustin [Dus03]
    Source: B. Littlewood, ed., Software Reliability, Achievement and Assesment

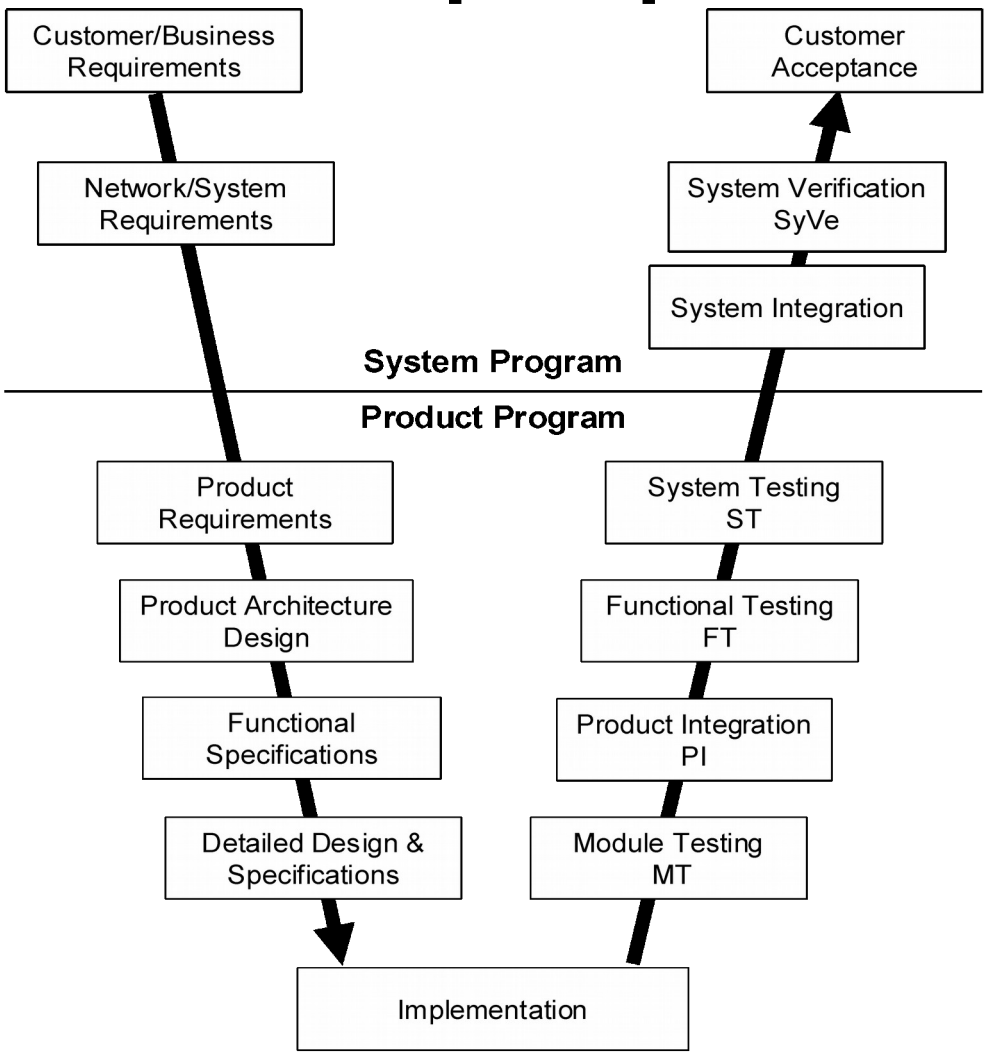
Prevention is Cheaper Than Cure

Phase	Relative Cost to Correct
Definition	1 $
High-Level Design	2 $
Low-Level Design	5 $
Code	10 $
Unit Test	15 $
Integration Test	22 $
System Test	50 $
Post-Delivery	100 $

  - (see following page) based on Jorma Tuominen [Tuo06] with differentiation:
    - Standard Software
    - Individual Software

Seven Testing Principles
Principle 3 – Early testing

Costs for defects [Tuo06]

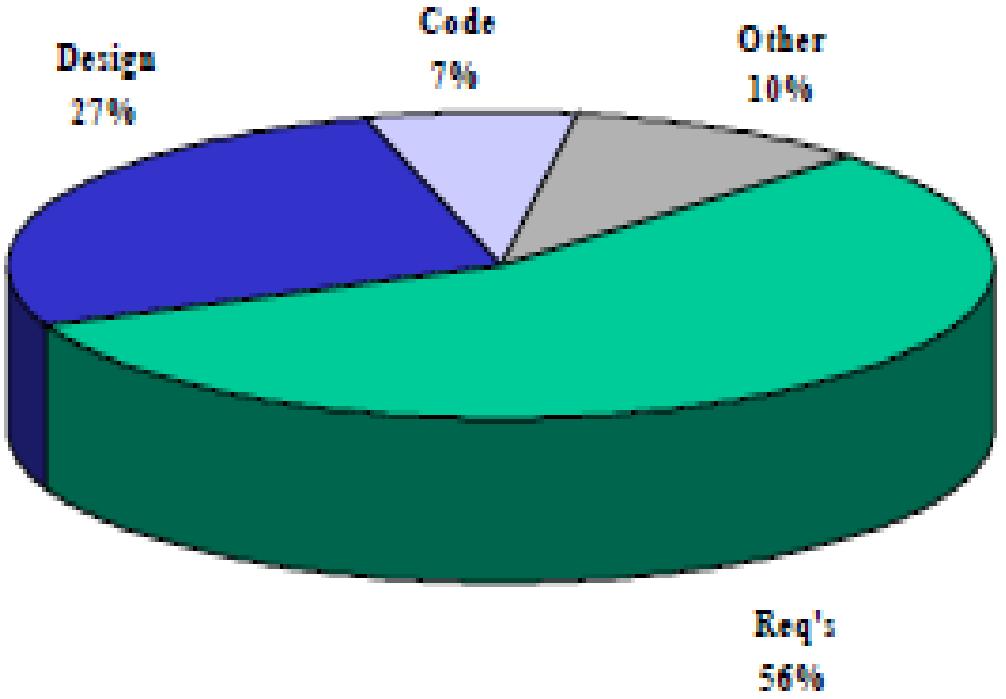


Phase where defect is discovered	Relative cost to correct a defect
Requirements	1
Design	3-6
Coding	10
Development testing	15-40
Acceptance testing	30-70
Production	40-1000

Phase where defect is discovered	Relative cost to correct a defect
Definition	1
High-level design	2
Low-level design	5
Code	10
Unit test	15
Integration test	22
System test	50
Post delivery	100+

Seven Testing Principles
Principle 3 – Early testing

What is the source of defects? [Ric05]



⇨ **Requirements play a central role in IT projects**

Seven Testing Principles
Principle 3 – Early testing

Example: Costs for defects in Germany [LOT01]

- Guessed loss because of software defects for medium and big companies in Germany:
  About 84,4 Billion Euro per year
- Productivity loss because of computer outfalls because of incorrect software about 2,6% of business volume:
  About 70 Billion Euro per year

Seven Testing Principles
Principle 3 – Early testing

Error avoidance (1/4)

- Prevention
  ... not cure
- The earlier a defect is detected, the cheaper is the correction
- More cheaper are defects, that don't occur at all
- Idea: Increasing quality „from the scratch“ with early (code) reviews ...

Seven Testing Principles
Principle 3 – Early testing

Error avoidance (2/4)

- „Peer reviews“ - capable experts review the work
  Use: *will detect about 31 % up to 93 % of all defects, average: 60 %*
- “Perspective review” – evaluators use the work for own tasks (For example specification: Generation of test cases, or a manual out of it)
  Use: *35 % more defects are detected compared to non-purposeful reviews*

Seven Testing Principles
Principle 3 – Early testing

Error avoidance (3/4)

- Own structured working, e.g. desk checks (Humphrey's Personal Software Process) including development of a theoretical solution, writing of pseudo code, then implementation
  Use: *up to 75 % less defects*
- Structured Walk through
  Programmer presents his work as moderator to a group, which tries to find defects.
  Yet in preparation he detects defects himself.

Seven Testing Principles
Principle 3 – Early testing

Error avoidance (4/4)

- Pair programming
  Quality is rising when doing pair programming [TDD05]
  TDD research studies in industry „ … showed that programmers using TDD produced code that passed 18 percent to 50 percent more external test cases than code produced by corresponding control groups“ with minimal impact to productivity

Study	Type	Number of companies	Number of programmers	Quality effects	Productivity effects
George[8]	Controlled experiment	3	24	TDD passed 18% more tests	TDD took 16% longer
Maximilien[9]	Case study	1	9	50% reduction in defect density	Minimal impact
Williams[10]	Case study	1	9	40% reduction in defect density	No change

Seven Testing Principles
Principle 4 – Defect clustering

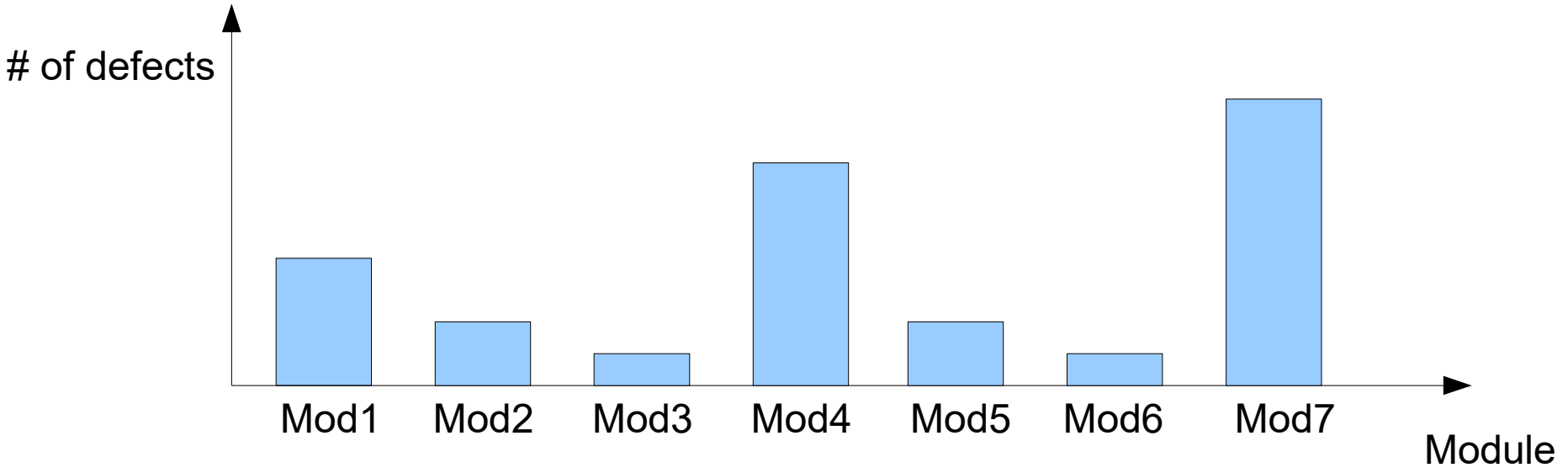
- Focus testing effort proportionally to the expected and later observed defect density of modules.
- A small number of modules usually contains most of the defects discovered during prerelease testing, or is responsible for most of the operational failures.

Seven Testing Principles
Principle 4 – Defect clustering

Pareto principle

- Defect clustering is based on the Pareto principle – the 80-20 rule.

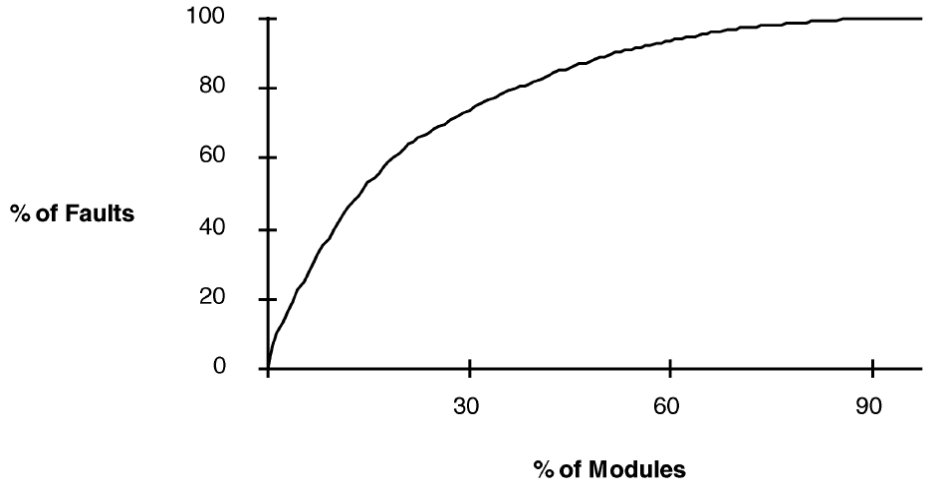
Approximately 80 per cent of the problems are caused by 20 per cent of the modules [Jaw13].



Seven Testing Principles
Principle 4 – Defect clustering

Pareto principle

- Fenton and Ohlsen detected in empirical investigations, that 20 % of the modules (equals to about 30 % of the code) are source of 60 % of the defects [FO00].



Pareto diagram showing % of modules versus % of faults for release n [FO00]

Seven Testing Principles
Principle 5 – Pesticide paradox

- If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new defects.
- To overcome this “pesticide paradox”:
  - Regularly review and revise test cases
  - Write new and different tests to exercise different parts of the software or system to find potentially more defects.

Seven Testing Principles
Principle 6 – Context dependence

Testing is context dependent

- Basic for Testing is the needed software quality.
- Testing is done differently in different contexts.
- Compare
  - Quality requirements of
    a medical software to a web application
  - Testing of
    a safety-critical software to an e-commerce site.
- Balance
  Effort for testing must be related to expected quality

Seven Testing Principles

Principle 7 – Absence-of-errors fallacy

- Finding and fixing defects does not help if the system built is unusable and does not fulfill the users' needs and expectations.

Sources (1/2)

- [Dus03] Elfriede Dustin: Book Excerpt: Software Testing Starts When Projects Begin, CSC, Effective Software Testing – 50 Ways to Improve Your Software Testing, 2003
- [FO00] Norman E. Fenton, Niclas Ohlsen: Quantitative Analysis of Faults and Failures in a Complex Software System; IEEE Transactions on Software Engineering, Vol. 26, No. 7, July 2000; fenton_ohlsson_published.pdf
- [ISTQB-CTFLS11] International Software Testing Qualifications Board: Certified Tester Foundation Level Syllabus, Released Version 2011, http://www.istqb.org/downloads/syllabi/foundation-level-syllabus.html
- [ISTQB-GWP15] Glossary Working Party of International Software Testing Qualifications Board: Standard Glossary of Terms Used in Software Testing, Version 3.01, 2015, http://www.istqb.org/downloads/glossary.html
- [Jaw13] Ranjeet Jawale: Defect clustering & Pesticide paradox, 2013 http://www.softwaretestingclub.com/profiles/blogs/defect-clustering-pesticide-paradox

Sources (2/2)

- [Jon05] Capers Jones: Software Cost Estimating Methods for Large Projects,
  © Software Productivity Research, LLC, Apr 2005
- [Lot01] Study of LOT Consulting Karlsruhe, page 31, 2001, IT-Services 3/2001
- [Ric05] Randall W. Rice: STBC The Economics of Testing, 2005, http://www.riceconsulting.com/public_pdf/STBC-WM.pdf
- [Sog16] Sogeti: TPI® - Test Process Improvement, 2016, http://www.sogeti.com/solutions/testing/tpi/
- [TDD05] Test-Driven Development: Concepts, Taxonomy, and Future Direction, IEEE Sep 2005
- [TMMI16] TMMI© foundation, 2016, tmmi.org
- [Tuo06] Jorma Tuominen: Test process analysis of Gateway GPRS Support Node, Nokia Networks, http://www.netlab.tkk.fi/opetus/s383310/05-06/Kalvot%2005-06/Tuominen_170106.ppt, Jan 2006