Software Test

Lesson 1
Overview / Introduction
v1.2

Uwe Gühl, Jittat Fakcharoenphol

Fall 2007/ 2008

Contents

- Overview / Introduction
  - Testing Definitions
  - Statements
  - Questions / Your experience
  - Goals
  - Software Quality
  - Costs for Bug fixing
  - Requirements on Testing
  - Error avoidance

Overview and Introduction

- Testing is not sexy
- When projects fail, Testing is the reason
- In Europe in ancient times bearer of bad news 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)

Overview and Introduction 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

Overview and Introduction 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)

Overview and Introduction Statements

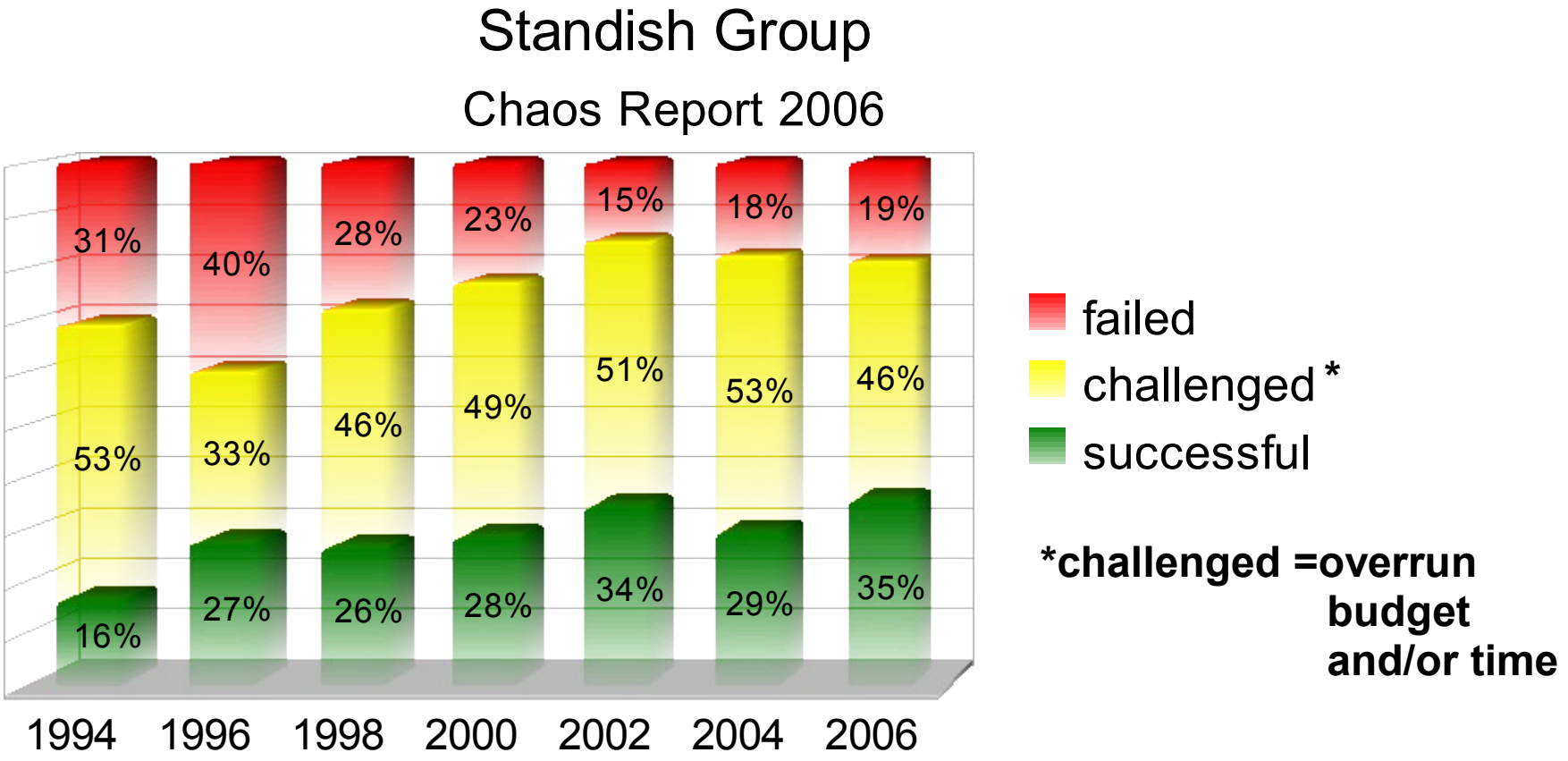
- “Program testing can be used to show the presence of bugs, but never to show their absence!” (Dijkstra 1969)
- “In most cases ‘what’ you test in a system is much more important than ‘how much’ you test” (Craig 2002)
- “Prioritise tests so that, when ever you stop testing, you have done the best testing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

Overview and Introduction Questions

- How could I influence the success of a project in a positive way with testing?

Overview and Introduction Questions

- An analysis of more than 9000 IT projects says [CW07]:



Overview and Introduction Questions

- Testing and You
  - Let's hear from Your experience
  - Group discussion (3 to 4 people in one group)
    - Have You ever done software testing?
    - How long did you do testing?
    - Was there a process, did you like it?
    - Which bugs did you find, how many?
    - What was the craziest, funniest, most stupid bug you found?
  - Results to the class

Overview and Introduction Questions

- How much money should I spend for testing?
- How should I ensure quality?
- How do I minimize the risks?
- How do I integrate the topic „Testing“ in the project?
- When I am happy with Testing?
- How do I avoid follow up costs?
- What is the benefit of Testing?
- How do I plan / control Testing?
- Who is going to test, which specialists?
- Which standards do exist?

Overview and Introduction Goals

- Thought
  - 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.

Overview and Introduction Goals

- Goal of Testing is to establish a fundament for the acceptance of the software by the customer based on the specification through
  1. High test coverage
  2. Low number of non critical defects
     → There should be no critical defect
  3. Statements concerning software quality

Overview and Introduction Goals

1. High test coverage

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

Overview and Introduction Goals

2. Low number of non critical defects
   - 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 phase
   - The acceptance criteria should determine, what the customer expects. A contract could content acceptance criteria concerning
     - Severity Level
     - Priority Level

Overview and Introduction 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?

Overview and Introduction Goals

- To achieve this goals, a release and configuration management is necessary
  - Expectation: Defect will be detected
  - The software vendor would like to fix the detected defects
  - ISO 10007:2003: "configuration management coordinated activities to direct and control configuration”
  - Release management defines the scope and the point in time of software deliveries
  - Discussion: Fixing of defects – changed scope
  - Discussion: Retest
  - The last delivered version should be in such high quality so that an acceptance is possible

Overview and Introduction Software quality

- Basic for Testing is the needed software quality
- Compare the quality requirements of a medical software with a web application

Overview and Introduction Software quality

- (Fatal) software defects [STW07]

  Mars Climate Orbiter Loss, September 1999
  At 2am 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?
  A week later the preliminary findings of a review team found that 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.

Overview and Introduction Software quality

- (Fatal) software defects [Web07]
  - Which defect was responsible for a crash of a Lockheed F-117A Night Hawk in 1982?

    The fly-by-wire system had been hooked up incorrectly (pitch was yaw and visa versa)

Overview and Introduction Software quality

- (Fatal) software defects [Web07]
  - In September 1994 three parking offender in Bayreuth (Germany) got a wrong charge because of a mistaken code. What was the contents of the charge?

    "Preparation of a war of aggression”

Overview and Introduction Software quality

- (Fatal) software defects [Web07]
  - 1985 a robot in the assembly hall of General Motors did not recognize the colour of black cars. What happened?

    All black cars left the assembly hall without a windscreen

Overview and Introduction Software quality

- (Fatal) software defects [Web07]
  - What was the problem with the mobile S65 from Siemens?

    The shut-down melody, when the battery was too low, was too loud, hearing damage for some people was the effect.

Overview and Introduction Software quality

- (Fatal) software defects [Web07]
  - 1996 a prototype of the Ariane 5 rocket of the European Space Agency was destroyed one minute after the start. Why?

    The code of the Ariane 4 was used.

Overview and Introduction Software quality

- (Fatal) software defects [Web07]
  - When did the NASA loose their Venus-spacecraft Mariner 1, and so about 80 Million US-Dollar, because of a software bug caused by a missing superscript bar in $\dot{r}_n$ in the specification?

    1962

Overview and Introduction Software quality

- (Fatal) software defects [Web07]
  - In Excel 2007 was a calculation defect, leading to many wrong spread sheets and accounts.
    What was the problem?

    In multiplication, where the result would have been 65,535, Excel calculated always 100,000

Overview and Introduction Costs

- Costs for testing *Software Development Activities – percentage of work effort by activities concerning test:* 18.5 % 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%	2.00%
03 Architecture		0.50%	1.00%	2.00%	1.50%	1.00%
04 Project plans		1.00%	1.50%	1.00%	2.00%	1.00%
05 Initial design		8.00%	7.00%	6.00%	7.00%	6.00%
06 Detail design		7.00%	8.00%	5.00%	6.00%	7.00%
07 Design reviews			0.50%	1.50%	2.50%	1.00%
08 Coding	30.00%	20.00%	16.00%	23.00%	20.00%	16.00%
09 Reuse acquisition	5.00%		2.00%	2.00%	2.00%	2.00%
10 Package purchase		1.00%	1.00%		1.00%	1.00%
11 Code inspections				1.50%	1.50%	1.00%
12 Independent verification and validation						1.00%
13 Configuration management		3.00%	3.00%	1.00%	1.00%	1.50%
14 Formal integration		2.00%	2.00%	1.50%	2.00%	1.50%
15 User documentation	10.00%	7.00%	9.00%	12.00%	10.00%	10.00%
16 Unit testing	30.00%	4.00%	3.50%	2.50%	5.00%	3.00%
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%	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**

Overview and Introduction Costs

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

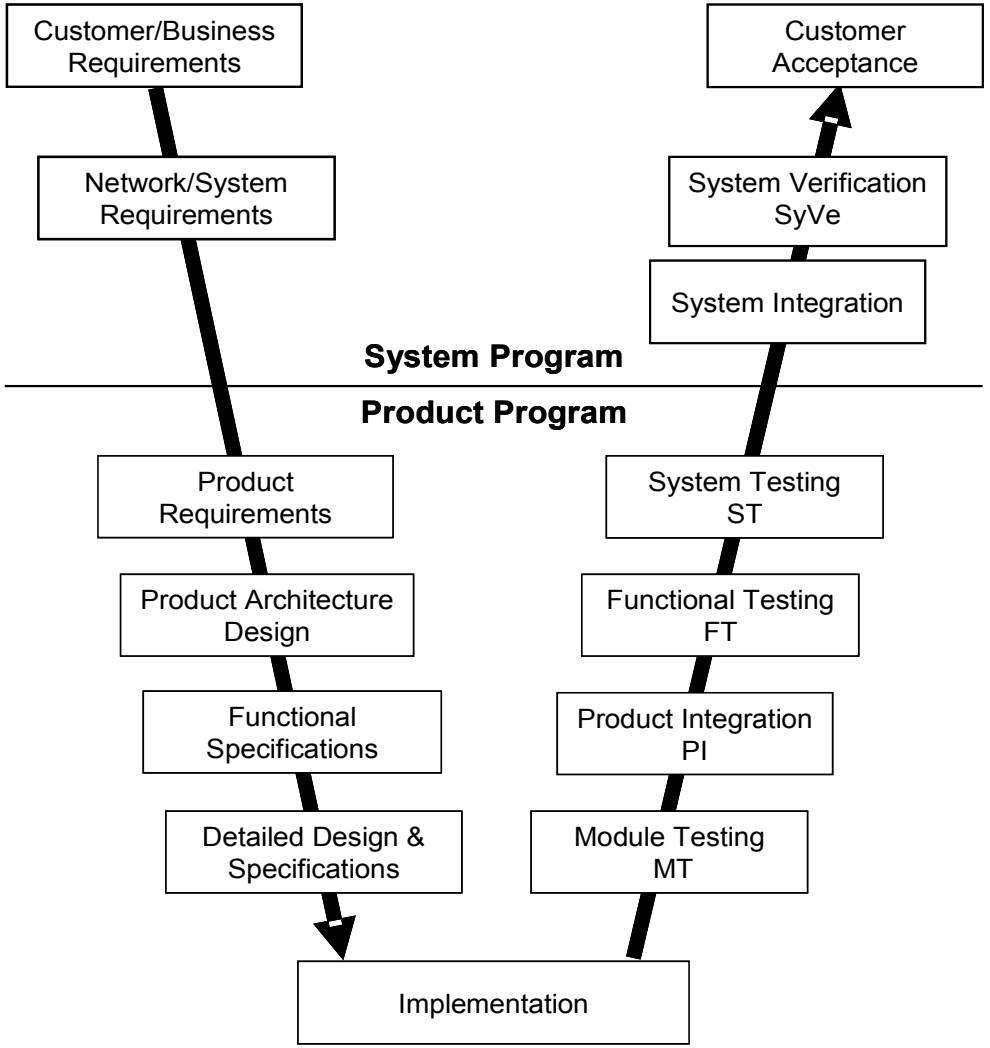
Prevention is Cheaper Than Cure

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

Overview and Introduction Costs

- 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+

Overview and Introduction Requirements

- Requirements on testing
  - Balance
    Effort for testing must be related to expected quality
  - Traceability
    - Defects must be reproducible!
    - How did they occur?
      - Which test cases?
      - Which test steps?
      - Which test data?
      - Which environment? etc.
    - Without reproducibility bug fixing is difficult!
      → Difference to „informal testing“
  - Testing is not accurate science!

Overview and Introduction Error avoidance (1/3)

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

Overview and Introduction Error avoidance (2/3)

- Suggestions for reviews (1/2) [BB01]
  - „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*

Overview and Introduction Error avoidance (3/3)

- Suggestions for reviews (2/2) [BB01]
  - 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*
- Additional suggestion
  - Structured Walk through
    Programmer presents his work as moderator to a group, which tries to find defects.
    Yet in preparation he detects defects himself.