1. Software Testing: Introduction

Definition

"Testing is the process of executing a program with the intent of finding errors."

Glen Myers

The role(s) of the tester ?

- Find important bugs fast
- Provide a general assessment of the quality of the product
- Certify that the product meets a particular standard
- Assure that the test process meets accountability standards
- Support the programmers
- Help predict and control the costs of support
- Help the clients improve product quality and testability; educate clients how to work with testers
- Do whatever is necessary to satisfy particular clients
- . . .

Who is to do the testing?

"It is impossible to test your own program"

Myers

- Testing should always be done by an outside party
- It has to be a destructive process – psychological reasons prevent us to be destructive towards our own creations
- Software tester – a distinct role in a software company – a distinct career in Software Engineering
- Intellectually challenging and fun

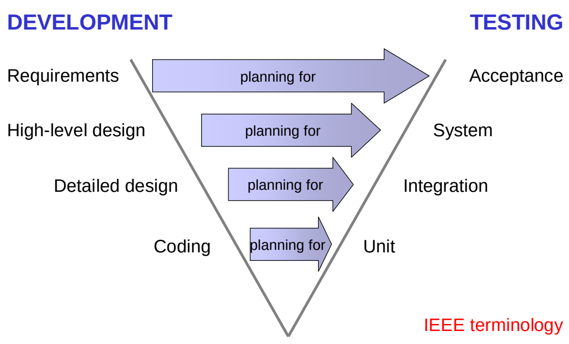
Terminology

- **Error**
  - Human mistake; bug is a coding mistake
- **Fault**
  - Consequence of error. May be categorized as
    - Fault of Commission – we enter something into representation that is incorrect
    - Fault of Omission – Designer can make error of omission, the resulting fault is that something is missing that should have been present in the representation

Terminology cont…

- **Failure**
  - Occurs when fault executes
- **Incident**
  - The symptom(s) associated with a failure that alerts user to the occurrence of a failure
- **Test**
  - The act of exercising software with test cases

Life cycle model for testing



Can we test for everything?

The fundamental problem in testing software

"In a program with just 70 branches, there are more test cases …

… than teaspoons of water …

… in the Pacific Ocean."

Bob Stahl

Software failure

Table 1 - Failure as a function of project size

Function Point	Early	On Time	Delayed	Cancelled
1 FP	14.68%	83.16%	1.92%	0.25%
10 FP	11.08%	81.25%	5.67%	2%
100 FP	6.06%	74.77%	11.83%	7.33%
1000 FP	1.24%	60.76%	17.67%	20.33%
10,000 FP	0.14%	28.03%	23.83%	48%
100,000 FP	0%	13.67%	21.33%	65%
Average	5.53%	56.94%	13.71%	23.82%

Source : Capers Jones, Software Systems Failure & Success

From A Primer on Requirements Engineering, Beaver Consulting

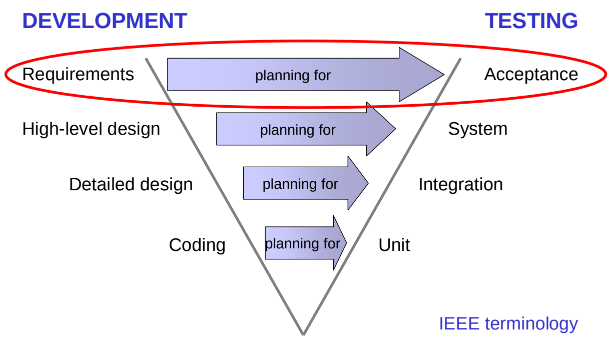
Project Impaired Factors

Factor	% Responses
Incomplete Requirements	13.1%
Lack of User Involvement	12.4%
Lack of Resources	10.6%
Unrealistic Expectations	9.9%
Lack of Executive Support	9.3%
Changing Requirements & Specifications	8.7%
Lack of Planning	8.1%
Didn't Need It Any Longer	7.5%
Lack of IT Management	6.2%
Technology Illiteracy	4.3%
Other	9.9%

Source : Standish Group (www.standish.com)

From A Primer on Requirements Engineering, Beaver Consulting

Life cycle model for testing



What is a requirement?

1. A condition or capability needed by a user to solve a problem or achieve an objective.
2. A condition or capability that must be met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed document.
3. A documented representation of a condition or capability as in 1 or 2.

IEEE Standard Glossary of Software Engineering Terminology

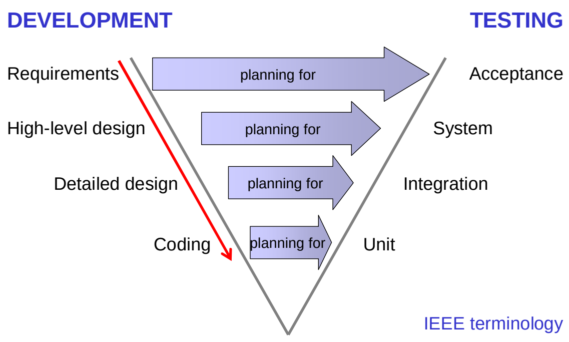
Requirements

Requirements must be

Recorded
Analysed
Agreed

particularly with the users

Life cycle model for testing



IEEE terminology

Requirement specification document

- Types of requirements
  - Functional
  - Non-functional
  - Constraint
  - Process or management
- Domain knowledge
- Definitions
- Headings
- Context

See “A Primer on Requirements Engineering”, http://www.cs.bham.ac.uk/~exc/Teaching/STesting/Requirement_Primer.pdf

Requirement specification - attributes

- Individual requirement must be
  - Atomic
  - Realisable
  - Testable
  - Unambiguous, specific and complete
  - High-level (it does not prematurely dictate the design)
- Document and a set of requirements must be
  - Non-conflicting
  - Non-overlapping
  - Not duplicated
  - Traceable, complete and appropriate
  - Consistent in the use of terminology

See “A Primer on Requirements Engineering”, http://www.cs.bham.ac.uk/~exc/Teaching/STesting/Requirement_Primer.pdf

Software testing

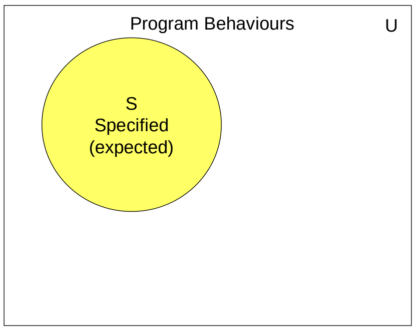
Until all the information demanded by the Requirement Specification document is not recorded

TESTING IS NOT POSSIBLE

Program structure and function

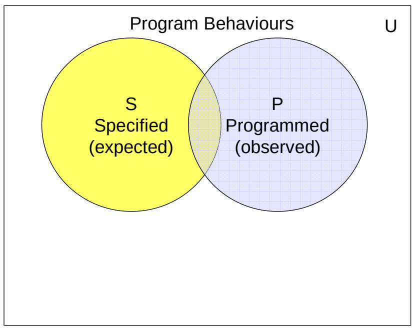
- Which of the two is the main focus of testing:
  - Structure (code)? What it is
  - Function (behaviour)? √ What it does

Program function testing



U=Universe of program behaviours

Program function testing

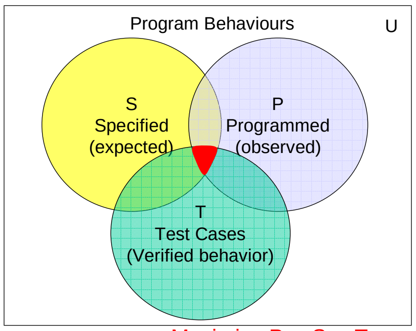


Fault Of Omission
S - P

Fault Of Commission
P - S

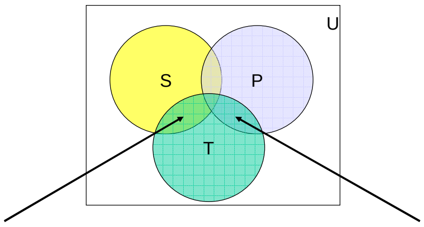
Correct portion
P ∩ S

Program function testing



Maximise P ∩ S ∩ T

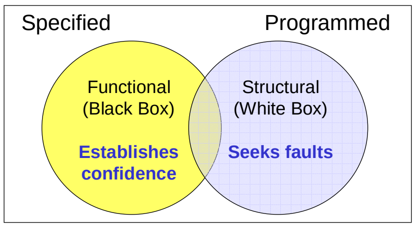
Test methodologies



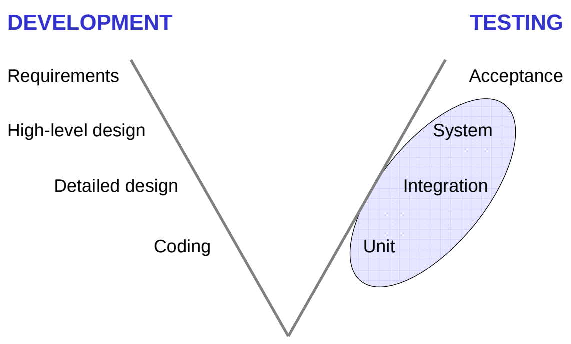
Functional (Black box) inspects specified behaviour

Structural (White box) inspects programmed behaviour

Test function



Levels of testing



The fundamental problem of testing software

- We cannot test for everything
- No system can be completely tested

- The need to have a clever testing methodology

Next lecture

Requirements specification (practical exercise)

Further reading

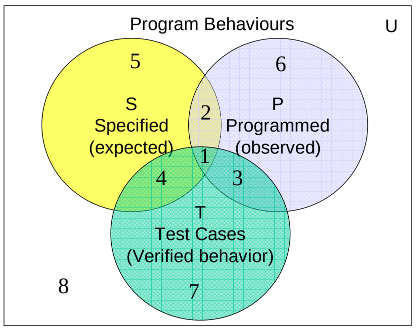
- Study in detail "Types of information and Requirements", especially sections 4.1 – 4.4 of "A Primer on Requirements Engineering", http://www.cs.bham.ac.uk/~exc/Teaching/STesting/Requirement_Primer.pdf

- Ensure that you have good understanding of the terms listed in the slides "Requirement specification – attributes" and "Requirement specification document"

Homework

In the Venn diagram below identify regions which correspond to

- Incomplete testing
- Incomplete specification



Homework

In preparation for the next lecture (Software Requirements Specification) study the document "IEEE Standard 830-1998 Recommended Practice for Software Requirements Specification"

http://www.cs.bham.ac.uk/~exc/Teaching/STesting/Standards/ieee_830.pdf
(local access only)