VERIFICATION
VALIDATION
TESTING
A set of inputs = test case

Examples:
- values of input variables
- loginname, password, click on mail icon, check mail, logout
- infinitely many more possible sequences

A test case exercises the program

Where do the test cases come from?
- from the (natural language) requirements
  - called BLACK-BOX testing
- from the code
  - called WHITE-BOX testing
Black & white

- **Black-box testing**
  - end of s/w lifecycle (can be component level)
  - ‘sanity check’: make sure you’ve satisfied the reqts!
  - essential always

- **White-box testing**
  - test coding, while coding
  - ignores the reqts: focus is bug removal
  - essential for mission-critical apps
A test case exercises some path through the code

Many strategies:

- **Statement coverage**
  - Each statement is executed at least once

- **Branch coverage**
  - Each branch in flow-of-control is executed at least once

- **Predicate coverage**
  - The truth values of every predicate is tested

- **Multiple Predicate coverage**
  - Every combination of truth value of predicates is tested
Example

If \((a<b \land b\neq c)\) print(“1”) else print(“0”)

<p>| | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>out</td>
<td>a&lt;b</td>
<td>b\neq c</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>1</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Who cares

Who cares

Who cares
loops

- A branch is taken every time loop-condition is true

- Do you test for:
  - 0 times through the loop?
  - 1 time?
  - pick any number times?
  - a really large number times?

- loop-condition known only at run-time, so what do you do?
  - possible strategy: 3 testcases for 0, 1, 2 times
White-box woes

- White-box testing:
  - is never finished (just run out of time to test any more)
  - is a hit-and-miss technique in bug hunting
  - must repeat all tests (gasp) after even just one code patch
    - change code anywhere can break the system unpredictably
    - called regression testing (talk about inefficiency)
  - is very very laborious and time consuming
  - statement coverage is the industry standard
    - other forms of coverage are too hard, too difficult to implement
    - it is surprisingly effective (80% bugs found)
Validate and verify

What’s the difference?

**Validation** is the process of checking whether the specification captures the customer's needs.

**Verification** is the process of checking that the software meets the specification.  

- black-box testing is validation
  - Is the behaviour correct?

- white-box-testing is verification
  - Is the code correct?

from Wikipedia
V&V

- Verification examples
  - white-box testing, code inspections, design analysis, specification analysis, static code analysis, proof of correctness
    - it is a largely objective process, even clinical

- Validation examples
  - black-box testing, requirements modelling, prototyping, user evaluation
    - it is more subjective
      - Does the (proposed) system satisfy the customer’s needs?
      - Does the customer really know what he needs?
      - Are the customer’s needs changing in the course of the development?
The big picture

Diagram courtesy of Steve Easterbrook
Behavioural vs code correctness

- **Validation correctness, a difficult act**
  - customers are fallible, changing, ‘unreliable’
  - ‘behaviour’ is such a real-world, fuzzy concept

- **Verification correctness, what is that actually?**
  - correct with respect to a **specification** (the definition says)
  - what is a spec?
    - A document
    - A clear/consistent/concise statement of the problem
    - Constructed by the design team to ‘formalise’ the requirements
      - Could involve natural language descriptions
      - Could involve a mathematical model
Correctness continued

Proof of correctness techniques

- Pre- and post-conditions are a specification
  - Define correctness
  - Form part of a design (fit together)

- Code driven
  - The effect of each line of code (on state) is determined

- Continuum:
  - Handles infinity
  - Code is correct, result is correct for continuous range: e.g. \( n \geq 0 \)

- Difficult, subtle tedious work
White-box testing

- Correctness is defined by the output of testcases (only)
- Discrete: individual input values must be well-chosen
  - Cannot handle infinity
  - Coverage of actual realistic input can be infinitesimally small
- Data-driven:
  - particular input values must generate particular output values
  - primitive/non-existent code checking (e.g. statement coverage)
- Tedious work (not really subtle or difficult)