Test Case Specifications and Test adequacy
Test Case Selection

• How do we create tests?
• Test are defined in terms of their adequacy against certain criteria
Test completeness and adequacy

• It is impossible to find a set of tests that ensures the correctness of a product
• We can only determine whether test sets are not adequate for a given criterion we set. Examples. A test suite is inadequate to guard against faults in:
  • ... specifications. If in the specifications, we give different permissions to different actors of a system and a test suite does not check that in fact the permissions are different
  • ... statements. If we consider executable statements and a test suite does not cover all the executable statements (except infeasible statements)
Test Case

• A test case is a choice of Inputs, Execution Conditions and Pass / Fail Criterion

• Example “Permission”.
  • I: {read, write},
  • EC: under domain environment,
  • PFC: {when owner -> TRUE, when guest -> FALSE}
Test Case

• Input: all kind of stimuli that contribute to a specific behaviour
• Output oracle: given against expected output or other peculiar way to determine that an output is correct (e.g., 100% coverage)
Test Case Specification

• A specification to be satisfied by one or more test cases.

• Examples:
  
  • **TC specification**: A system has multiple actors. **TC Input**: \{owner, guest, administrator or user, group, other\}
  
  • **TC specification**: Word processor must open one or more files; **TC 1: Input**: one file; **TC2: Input**: 2 files

• It may also describe some aspects of input and output. Example:
  
  • **TC specification**: Word processor requires some recovery policy while opening files
Types of Testing

• Functional testing (Black-box testing).
• Structural testing (White-box)
• Fault based testing
• Fault-seeding testing
Functional Testing (Black-box testing)

- Test Case specification can be derived from **product specification**, which in turn can include description of **input and output**, or from any system **observable behaviour** (often reported as deviation from the expected one). Example:
  - **Observation**: a DB system requires robust failure recovery in case of power loss
  - **TC specification**: Removing power at certain critical point in processing queries
  - Example:
    - Finite State Machine. If the system is is described as a control flow graph, a **test case specification can be a selection of feasible execution paths**
Structural Testing (White-box Testing)

• Test cases are derived from the structure of the code

1. public static void String collapseSpaces(String argStr)f
2. char last = argSrt.charAt(0);
3. StringBuffer argBuf = new StringBuffer();
4. for(int cldx=0; cldx<argStr.length(), clds++){
5.     char ch = argStr.charAt(cldx);
6.     if(ch!=' ' || last!=' '){
7.         argBuf.append(ch)
8.         last=ch;
9.     }
10. }
11.}
Structural Testing (White-box Testing)

• Test case specification for general rules: Example: Empty string must be tested

• Test case specification for conditions: Example: test the two conditions of the if clause separately
Fault-base Testing

• Test cases are derived from reported fault

• Example

  • Reported: Race condition experienced in multi threads

  • Test case specification: test for synchronisation in multi threads
Example of fault-base Testing

- Fault seeding testing
- Mutation Testing
Fault Seeding Testing

• Fault-seeding testing is fault-base testing that deliberately seeds faults and define test case specifications to test them
Fault Seeding Testing

• Let $S$ be the total number of seeded faults, and $s(t)$ is the number of seeded faults that have been discovered at time $t$.
• $s(t)/S$ is the seed-discovery effectiveness of testing to time $t$. 
Issues

• Inserting faults into software involves the obvious risk of leaving them there
• Faults injected are "typical" faults
Mutation Testing

- Fault mutation is a **fault-base testing** technique that mutates the original code.
- Each atomic change is called **mutant**. Each mutant injects one fault.
  - It creates a test case per mutant.
  - If a mutant fails any test, then it is said to be **killed**.
  - All mutants that are not killed are said to remain **live at this point**.
## Example of Mutation Operators

<table>
<thead>
<tr>
<th>Mutation Operator</th>
<th>Original Code</th>
<th>Mutated Code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Add 1</strong></td>
<td>q=0</td>
<td>q=1</td>
</tr>
<tr>
<td><strong>Replace Variable</strong></td>
<td>r=x</td>
<td>r=y</td>
</tr>
<tr>
<td><strong>Replace Operator</strong></td>
<td>q=q+1</td>
<td>q=q-1</td>
</tr>
</tbody>
</table>
Test Case Adequacy
Test Case Adequacy

• Ideally, adequacy means that test cases show correctness of a product.

• In practice, we can only approximate the problem by setting a set of adequacy criteria and we can only discuss whether a set of test cases is inadequate against such criteria.

• If this happens, we can extend the test cases.
Test Case Adequacy

- Even if we are able to prove that all test cases satisfy all adequacy criteria, we cannot say that the product is correct.
Test Case Adequacy

• **Adequacy criterion**: a predicate that is TRUE/FALSE on a pair <Program, Test Suite>
  • Example: the test suite exercises all executable statements
Test obligation

- **Test obligation**: a partial test specifications that checks an adequacy criterion
  - Example: Execute an executable statement
- An adequacy criterion can be checked by one or more test obligations
  - Example: Execute all executable statements
Test Suite

• A test suite (i.e. a set of test cases) satisfies an adequacy criterion if all its test cases succeed and for every test obligation checking the adequacy criterion, there exists at least a test case that satisfies it
Adequacy degree

• The adequacy degree is the level of adequacy a test suite achieves against an adequacy criterion:
  • Example: \textit{percentage} of statement coverage for a pair \(<\text{myProgram, myTestSuite}>\)
  • Example: \textit{ratio} of killed total mutants (K/M) measures the adequacy degree of a test suite in mutation testing
Test Subsumption

• An adequacy criterion A subsumes an adequacy criterion B if every test suite X that satisfies A contains some test suite Y that satisfies B (i.e. X also satisfies B)

• Example: Branch coverage subsumes executable statement coverage

• We tend to discard adequacy criteria that are subsumed
Test Subsumption

• Stronger adequacy criteria can potentially reveal more faults
Structural Testing (White-box Testing)

- Test cases are derived from the structure of the code

```java
public static void String collapseSpaces(String argStr)
{
    char last = argStr.charAt(0);
    StringBuffer argBuf = new StringBuffer();
    for(int cldx=0; cldx<argStr.length(), clds++;
    char ch = argStr.charAt(cldx);
    if(ch!='' || last!=''){
        argBuf.append(ch);
        last=ch;
    }
}
```
Example

• In the code example above, we can satisfy the statement coverage (obligation: execute statements) by a test case with any string with no spaces, but if we want to satisfy the if condition (branch coverage) we need to create another test case with a string containing empty characters (to test the true and the false of the branch)
Example

• If we want to test the for loop at different counter lengths, we might need to add new test case obligations.

• For example, in the above code example, testing for `argStr.length()=0` will include a new test case for an empty string. If it is not checked it can cause future failures.