6.5 Testing and Test-First Programming

扩展，一级标题，二级标题，三级标题，注

目录

6.5.1 Software Testing

Definition

Software testing is an investigtion conducted to provide stakeholders with information about the quality of the product or service under test.

Why Software Testing is Hard

Exhaustive testing is infeasible:

The space of possible test cases is generally too big to cover exhaustively.

穷举+暴力=不可能

Haphazard testing makes no sence

Haphazard testing is less likely to find bugs, unless the program is so buggy that an arbitrarily-chosen input is more likely to fail than to succeed. 靠偶然测试没意义

Random or statistical testing doesn't work well for software.

Other engineering disciplines can test small random samples (e.g. 1% of hard drives manufactured) and infer the defect rate for the whole production lot. 基于样本的统计数据对软件测试意义不大—软件与产品的巨大差异.

Software behavior varies discontinuously and discretely across the space of possible inputs. 软件行为在离散输入空间中差异巨大

That's different from physical systems 无统计分布规律可循

Characteristic

1. Testing's goal runs counter to the goals of other development activities. The goal is to find errors. 测试跟其他活动的目标相反：破坏、证错、"负能量"
2. Testing can never completely prove the absence of errors.
   再好的测试也无法证明系统里不存在错误

Test case

Definition

test case = {test inputs + execution conditions+ expected results}

测试用例：输入+执行条件+期望结果

Classification by levels（由测试的分级结构进行的测试）

Unit testing（单元测试）:

范围：refers to tests that verify the functionality of a specific section of code, usually at the function level.（function and class）

Unit testing focuses verification effort on the smallest unit of software design—the software component or module. 针对软件的 最小单元模型开展测试，隔离各个模块，容易定位错误和调试

Integration testing（集成测试）:

the combined execution of two or more classes, packages, components, subsystems that have been created by multiple programmers or programming teams. （classes, packages,components,subsystems）

System testing（系统测试）:

to test a completely integrated system to verify that the system meets its requirements, which executes the software in its final configuration.（system）

acceptance testing（验收测试）:

在将产品交由客户时，所进行的测试。

Regression testing（回归测试）:

当进行某项测试后，遇到问题等因素，需要对产品进行修改，而后重新进行测试。

其他类型的测试：

Installation testing, compatibility testing, usability testing…

拓展：testing vs debugging

Testing is a means of detecting errors.（测试：发现是否有错误。）

Debugging is a means of diagnosing and correcting the root causes of errors that have already been detected.（调试：识别错误根源，消除错误。）

Classification by types（由测试类型分类）

White-box testing（白盒测试）

White-box testing tests internal structures or workings of a program by seeing the source code. （白盒测试：对程序内部代码结构的测试）

The tester chooses inputs to exercise paths through the code and determine the appropriate outputs. 根据程序执行路径设计测试用例

Black-box testing（黑盒测试）

lack-box testing treats the software as a “black box”, examining functionality without any knowledge of internal implementation, without seeing the source code. （黑盒测试：对程序外部表现出来的行为的测试）

Test cases for black-box testing are built around specifications and requirements.检查程序是否符合规约

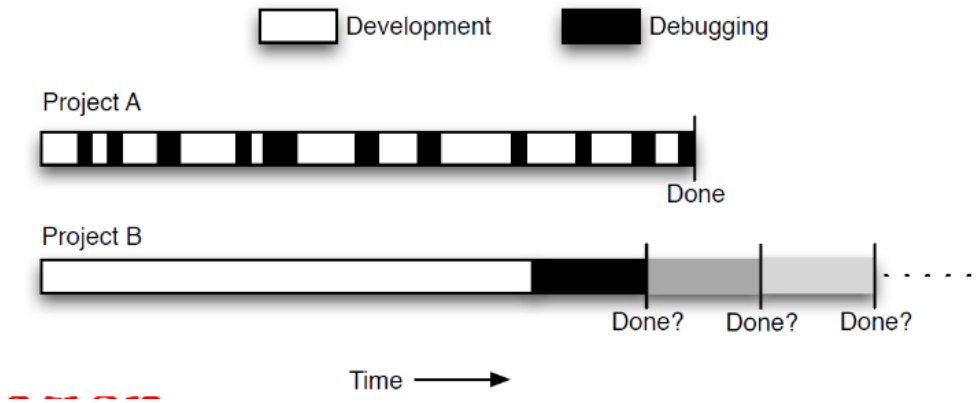
6.5.2 Test-First Programming

Definition:

Write the tests before you write the code

Process:

1. Write a specification for the function. 先写 specification
2. Write tests that exercise the specification. 再写符合 spec 的测试用例
3. Write the actual code. Once your code passes the tests you wrote, you’re done. 写代码、执行测试、有问题再改、再执行测试用例，直到通过它



6.5.3 Write a black-box testing

Two Extremes for Covering the Partition

Full Cartesian product 笛卡尔积：全覆盖

Every legal combination of the partition dimensions is covered by one test case. 多个划分维度上的多个取值，要组合起来，每个组合都要有一个用例

测试完备，但用例数量多，测试代价高。

Cover each part 覆盖每个取值：最少 1 次即可

Every part of each dimension is covered by at least one test case, but not necessarily every combination. 每个维度的每个取值至少被 1 个测试用例覆 盖一次即可

测试用例少，代价低，但测试覆盖度未必高。

Equivalence Partitioning （等价类划分）

Definition

If a set of objects can be linked by relationships that are symmetric, transitive, and reflexive, an equivalence class is present.

如果一组对象具有对称、传递、自反的关系，那么他们就是一个等价类。

roles

Equivalence partitioning is a testing method that divides the input domain of a program into classes of data from which test cases can be derived.

基于等价类划分：将被测函数的输入域划分为等价类，从等价类中导出测试用例

reasons

The idea behind equivalence classes is to partition the input space into sets of similar inputs on which the program has similar behavior, then use one representative of each set.

基于的假设：相似的输入，将会展示相似的行为。故可从每个等价类中选一个代表作为测试用例即可

Boundary Value Analysis（边界值分析方法：对等价类划分方法的补充）

Definition

the boundaries of the input domain rather than in the “center”

选取测试用例在可能的边界值处，而不是中央随机。

测试函数耦合处理

These tests inevitably interact with each other 测试的众多方法可能存在相互依赖的关系。

如果被依赖的其他方法有错误，可能导致被测试方法的测试结果失效。

Methods

A. 临时修改原类的访问属性，直接访问原类的数据成员，测试完成后修改回原值。

B. 额外做一些方法进行辅助测试，测试完成后进行注释或删除。

目的是改变双向依赖关系为单向依赖关系或不依赖。

An example

```
/**
 * Reverses the end of a string.
 *
 * For example:
 *   reverseEnd("Hello, world", 5)
 *   returns "Hellodlrow ,"
 *
 * With start == 0, reverses the entire text.
 * With start == text.length(), reverses nothing.
 *
 * @param text    non-null String that will have
 *                its end reversed
 * @param start   the index at which the
 *                remainder of the input is
 *                reversed, requires 0 <=
 *                start <= text.length()
 * @return input text with the substring from
 *               start to the end of the string
 *               reversed
 */
static String reverseEnd(String text, int start)
```

Document the strategy at the top of the test class:

```
/*
 * Testing strategy
 *
 * Partition the inputs as follows:
 * text.length(): 0, 1, > 1
 * start:         0, 1, 1 < start < text.length(),
 *                text.length() - 1, text.length()
 * text.length()-start: 0, 1, even > 1, odd > 1
 *
 * Include even- and odd-length reversals because
 * only odd has a middle element that doesn't move.
 *
 * Exhaustive Cartesian coverage of partitions.
 */
```

Each test method should have a comment above it saying how its test case was chosen, i.e. which parts of the partitions it covers:

```
// covers test.length() = 0,
//        start = 0 = text.length(),
//        text.length()-start = 0
@Test public void testEmpty() {
    assertEquals("", reverseEnd("", 0));
}
```

6.5.3 Write a white-box testing

goal

1. Guarantee that all independent paths within a module have been exercised at least once 保证每条独立的路径至少运行一次
2. Exercise all logical decisions on their true and false sides,每一个判断真假各一次
3. Execute all loops at their boundaries and within their operational bounds, 运行循环到边界处以及在其之内
4. Exercise internal data structures to ensure their validity.确保每个数据结构都有效

拓展 independent/basis path testing（独立/基本路径测试）

对程序所有执行路径进行等价类划分，找出有代表性的最简单的路径(例如循环只需执行 1 次)，设计测试 用例使每一条基本路径被至少覆盖 1 次。

Code coverage（代码覆盖度）

Definition

a measure used to describe the degree to which the source code of a program is executed when a particular test suite runs.

Kinds（种类）

a) Function coverage（函数覆盖）：所有函数都调用一遍
b) Statement coverage（语句覆盖）：每个可执行语句都走一遍
c) Branch coverage: for every if or while or switch-case or for statement in the program, are both the true and the false direction taken by some test case? 分支覆盖

   分支覆盖又称判定覆盖：使得程序中每个判断的取真分支和取假分支至少经历一次，即判断的真假均曾被满足。只看**最终**判断条件真假。
d) Condition coverage: for every condition in if/while/for/switch-case statement, are both the true/false direction taken by some test case? 条件覆盖

   条件覆盖：要使得每个判断中的每个条件的可能取值至少满足一次。观察判断条件的各个选项取值取值。
e) Path coverage 路径覆盖：每一个可能的路径都走一遍

拓展：各种白盒测试的结果比较

测试难度：路径覆盖>分支覆盖>语句覆盖
测试效果：路径覆盖>分支覆盖>语句覆盖