北京理工大学

软件测试与质量保证

2.2 代码走读

张宇霞 副研究员

目录

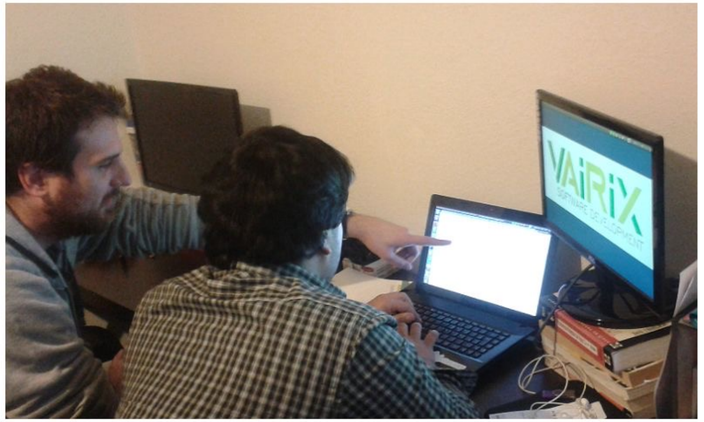
CONTENTS

目录

CONTENTS

01 代码走读

- **走读：同行评审中最自由的一种形式**
  - 主要目的：评价软件制品 (比如软件代码)。
  - 其他目的：技术的交换、参与人员的技术培训、设计思想的介绍等。
  - 评价什么：代码正确性、效率、可读性。
  - 走读成员：项目内部的其他开发人员。
  - 方式：对照检查表（checklist）检查代码。



01 代码走读

- **代码走读是代码静态测试的主要方法**
- **代码静态测试**
  - 开发者自己执行的数据流测试
  - 同行执行的代码走读

01 代码走读

■ 代码走读，查什么？

- ➢ 代码风格和规则
  - ● 独立于业务逻辑（软件的功能）
- ➢ 程序设计和结构
- ➢ 业务逻辑

01 代码风格和规则检查

- **程序风格的一致性、规范性；**

  for(i=0;i++;i<10)VS for(i=10;i--;i>0)

  for VS while VS do{} while

- **标识符定义的规范性和一致性；**

  - 保证变量命名能够见名知意，并且简洁。长度适当、命名规范、容易记忆、能够拼读。
  - 用相同的表示符代表相同功能的软件实体，不要用相同的表示符表示不同功能的软件实体。

01 代码风格和规则检查

■ 程序是否清晰、简洁、容易理解

- ➢ 冗长的程序并不一定就不清晰。

■ 注释

- ➢ 注释是否完整；
- ➢ 是否清晰简洁；
- ➢ 是否正确的反映了代码的功能；
- ➢ 是否做了多余的注释。

01 代码风格和规则检查

■ 注释文档是否完整；

- ➢ 对包、类、属性、方法、参数、返回值的注释是否正确且容易理解；
- ➢ 参数类型、参数的限定值的注释是否正确。
  - ● 特别是对于形参与返回值中关于神秘数值的注释是否清晰。
  - ● 比如：返回值 1 代表什么，2代表什 么，3 代表什么。
  - ● 对于返回结果集(Result Set)的注释。

01 代码走读

■ 代码走读，查什么?

- 代码风格和规则
- 程序设计和结构
- 业务逻辑

01

程序设计和结构的检查

■ 模块接口的正确性检查

- 确定形式参数个数、数据类型、顺序是否正确；
- 确定返回值类型及返回值的正确性；
- 以详细设计为基准。

■ 输入参数是否有正确性检查

- 缺少参数正确性检查的代码是造成软件系统不稳定的主要原因之一。

01 程序设计和结构的检查

```
void strFunction ( const char* destStr , const char* cerStr)
{
    assert( (destStr != NULL )  && ( cerStr != NULL) );

    if( NULL == destStr || NULL == cerStr )
    {
        cout << "invalid argument(s)." ;
        return ;
    }
}
```

输入参数如何做正确性检查

01

程序设计和结构的检查

■ 调用其他方法接口的正确性

- 检查实参类型正确与否、传入的参数值正确与否、参数的个数正确与否。
- 返回值正确与否，有没有误解返回值所表示的意思。
- 最好对每个被调用的方法的返回值用显式代码作正确性检查。
- 如果被调用方法出现异常或错误，程序应该给予反馈并添加适当的出错处理代码。

Try　　Catch

01

程序设计和结构的检查

■ 出错处理

- 出错的描述难以理解；
- 出错的描述不足以对错误定位，不足以确定出错的原因；
- 显示的错误信息与实际的错误原因不符；
- 对错误条件的处理不正确；
- 在对错误进行处理之前，错误条件已经引起系统的干预。

01 程序设计和结构的检查

```
char *str1 = "1", *str2 = NULL;
try {
    int num1 = parseNumber(str1);
    int num2 = parseNumber(str2);
    printf("sum is %d\n", num1 + num2);
} catch (NumberParseException) {
    printf("输入不是整数\n");
}
```

有意义的出错信息（提示）

01 程序设计和结构的检查

■ 保证表达式、SQL语句的正确性

A||B&C

A||(B&C)

- ➢ 检查所编写的SQL语句的语法、逻辑的正确性。
- ➢ 对表达式应该保证不含二义性，对于容易产生歧义的表达式或运算符优先级，

《 、=、 》、 &&、||、++、 --等，可以采用扩号（）运算符避免二义性。

■ 检查常量和全局变量使用的正确性

- ➢ 检查常量或全局变量的取值和数据类型；
- ➢ 保证常量的每次引用，是否保持数值和类型的一致性。

01 程序设计和结构的检查

Call(5);
Peter=Students[2]

- **数字是否进行了命名。**
  - 包括各种常数、数组的大小、字符位置以及程序中出现的以文字形式写出的数值。
- **检查代码是否可以优化、算法效率是否最高。**
  - 如：SQL语句是否可以优化，是否可以用1条SQL语句代替程序中的多条SQL语句的功能；
  - 如：循环是否必要，循环中的语句是否可以抽出到循环之外等。

目录

CONTENTS

02 代码走读—小结

代码走读主要包括格式检查、设计和结构检查、业务逻辑检查

代码走读是代码静态测试的主要手段

一起来走读

```python
#!/usr/bin/python
# -*- coding: UTF-8 -*-
from __future__ import division
import os
import numpy as np
import pymysql
from numpy import median

path = os.path.abspath('/Users/Yuxia/Desktop/monopoly/code/')
conn = pymysql.connect(host='127.0.0.1', port=3306, user='root', passwd='123456', db='openstack2019', charset='utf8')
cursor = conn.cursor()

# 计算每个repository总的commit数
with conn.cursor() as cursor:
    sql = 'select repository, count(distinct version) ' \
          'from commit_monopoly ' \
          'where version < 19 ' \
          'group by repository'
    cursor.execute(sql)
    repo_ver = cursor.fetchall()

dict_repo_ver = {}
for i in repo_ver:
    key = i[0]
    dict_repo_ver[key] = i[1]

analyzed_repos = np.loadtxt(path + "/data_fse/rq2_analyzed_repos.csv", delimiter=",", dtype=str, encoding='utf-8-sig')

vers = []
for i in analyzed_repos:
    print(type(i))
    ver = dict_repo_ver[i]
    vers.append(int(ver))

print(median(vers))
```

谢谢！