Software Engineering: Mining Software Repositories

Paper Notes

Zhouyiyang

2025-12-01

目录

原文背景与动机

重构是提升代码质量的重要手段，但大多数研究集中在源代码重构，对测试代码重构的研究较少。测试代码与源代码之间存在紧密联系，当源代码

被重构时，测试代码往往也需要相应调整。

第一阶段：实证分析 (回答 RQ1)

这一阶段的目标是揭示现象，搞清楚 “是什么” 和 “有多频繁”。

```
From Ben Kenobi
```

1. 实验对象与数据准备

数据源：SmartSHARK 数据集，包含 77 个 Apache 开源 Java 项目的完整开发历史。工具：使用 RefactoringMiner 2.2（高精度重构识别工具）来识别每个提交中的重构操作。关键步骤：由于数据集中的重构位置信息不完整，作者重新运行了 RefactoringMiner 以确保能获取每个重构操作所修改的具体文件。

2. 测试文件识别方法

为了判断一个重构是作用于测试代码还是源代码，需要先识别出测试文件。作者采用了一个两级过滤策略：

文件名过滤：筛选出路径名中包含 “test” 的.java 文件。内容分析：使用 JavaParser 解析这些 Java 文件，检测其中是否包含 JUnit 测试方法。这一步能有效减少误报（例如，将名为 Test.java 的普通源文件误判为测试文件）。

3. 提交分类规则

根据一个提交中所有重构操作所触及的文件类型，将其分为三类：

仅测试重构：所有重构都只修改了测试文件。仅源代码重构：所有重构都只修改了源文件（非测试文件）。共现重构：提交中同时存在对测试文件和源文件的重构操作。

4. 数据分析方法

比例计算：对每个项目，计算三类提交各自占所有重构提交的百分比，然后汇总所有项目的分布情况（使用箱线图展示）。重构类型分析：仅在共现重构提交中，提取所有应用于测试代码的重构类型，并统计它们的出现频率，找出 Top 10。