- 阅读论文，准备报告

1. 《方法级别的细粒度软件缺陷定位方法》——MethodLocator
2. 《软件错误自动定位关键科学问题及研究进展》——“失效－错误定位－理解”模型
3. 《基于信息检索的软件缺陷定位方法综述》
4. 《软件多缺陷定位方法研究综述》——MFL
5. 《基于程序频谱的两阶段缺陷定位方法》

# 基于程序频谱的两阶段缺陷定位方法

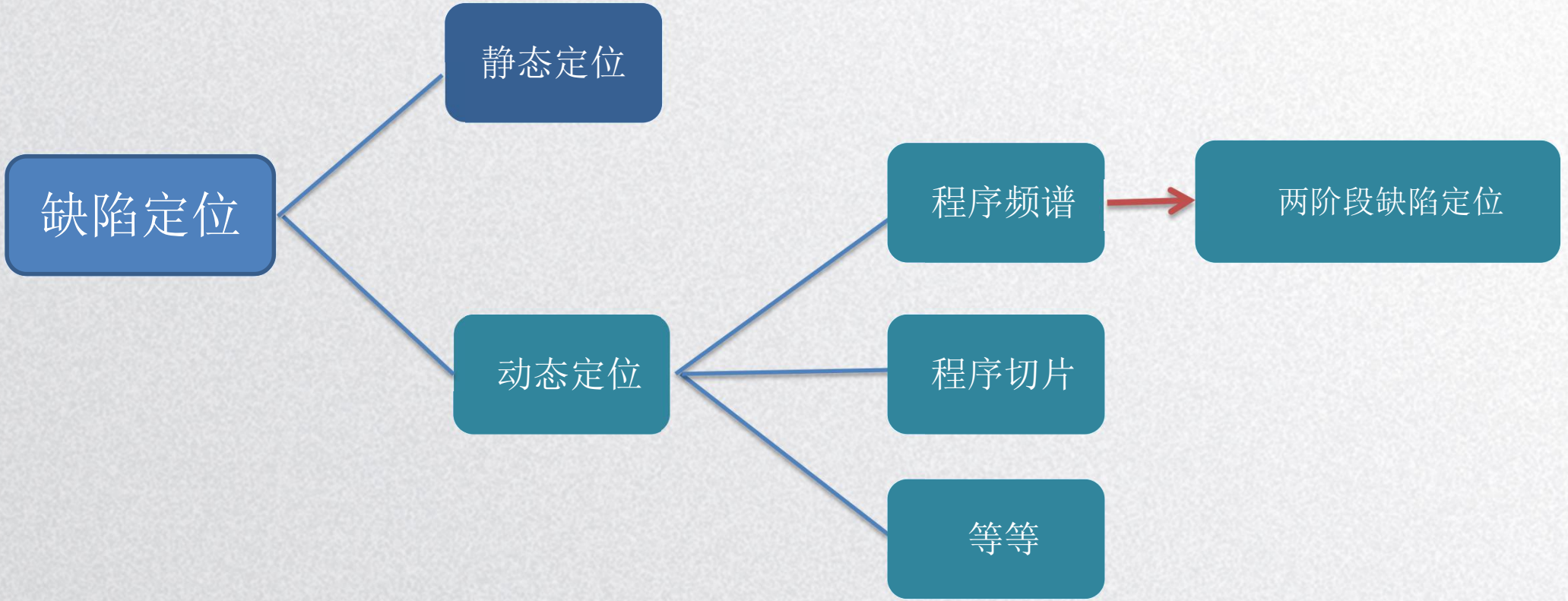
作者：伍 佳， 洪 玫， 万 莹， 邓惠心， 潘春霞

汇报人：陈冰婷　　导师：邹卫琴

# 目录

Contents

# 为什么使用两阶段缺陷定位方法



# 两阶段缺陷定位方法

## 可疑程序模块定位

1.构建程序频谱
2.生成可疑程序模块推荐列表

## 可疑语句定位

1.计算可疑分数阈值
2.构建交叉表
3.生成可疑语句推荐列表

# 1.1.构建程序频谱

测试覆盖率工具： EclEmma、Cobertura、Jacoco 等

$$S = \begin{bmatrix} T_{m \times n} \\ R \end{bmatrix} = \begin{bmatrix} t_{11} & t_{12} & \cdots & t_{1n} \\ t_{21} & t_{22} & \cdots & t_{2n} \\ \vdots & \vdots & & \vdots \\ t_{m1} & t_{m2} & \cdots & t_{mn} \\ r_1 & r_2 & \cdots & r_n \end{bmatrix}$$

# 1.2.生成可疑程序模块推荐列表

a ）程序组件的怀疑率与该程序实体被成功测试用例覆盖的次数成反比；
b ）程序组件的怀疑率与该程序实体被失败测试用 例覆盖的次数成正比；
c ）程序组件的怀疑率与该程序实体未被失败测试用例覆盖的次数成反比；
d ）在设定怀疑率公式时， 应该为假设 b ）设置更高的权重。

覆盖程序组件 m 的失败测试用例数量和
成功测试用例数量

$$\text{Tarantula}(m) = \frac{n_{cf}(m)/n_f}{n_{cf}(m)/n_f + n_{cs}(m)/n_s}$$

测试用例集中失败测试用例数量和
成功测试用例数量

# 1.2.生成可疑程序模块推荐列表

| | | |
|---|---|---|
| 1 | org.jfree.chart.plot.CategoryPlot->setRenderer->(Lorg/jfree/chart/renderer/category/CategoryItemRenderer;)V | 0.990888383 |
| 2 | org.jfree.chart.plot.CategoryPlot->setRenderer->(ILorg/jfree/chart/renderer/category/CategoryItemRenderer;Z)V | 0.984162896 |
| 3 | org.jfree.chart.plot.CategoryPlot-><init>->()V | 0.970982143 |
| 4 | org.jfree.chart.plot.CategoryPlot->setDataset->(Lorg/jfree/data/category/CategoryDataset;)V | 0.968819599 |
| 5 | org.jfree.chart.plot.CategoryPlot->setDataset->(ILorg/jfree/data/category/CategoryDataset;)V | 0.951859956 |
| 6 | org.jfree.chart.renderer.category.LineAndShapeRenderer-><init>->()V | 0.941558442 |
| 7 | org.jfree.chart.plot.CategoryPlot->getIndexOf->(Lorg/jfree/chart/renderer/category/CategoryItemRenderer;)I | 0.92161017 |
| 8 | org.jfree.chart.renderer.category.AbstractCategoryItemRenderer->getLegendItems->()Lorg/jfree/chart/LegendItemCollection; | 0.92161017 |
| 9 | org.jfree.data.category.DefaultCategoryDataset->addValue->(DLjava/lang/Comparable;Ljava/lang/Comparable;)V | 0.910041841 |
| 10 | org.jfree.chart.renderer.category.LineAndShapeRenderer-><init>->(ZZ)V | 0.896907217 |
| 11 | org.jfree.chart.LegendItemCollection->getItemCount->()I | 0.878787879 |
| 12 | org.jfree.chart.LegendItemCollection-><init>->()V | 0.871743487 |
| 13 | org.jfree.data.category.DefaultCategoryDataset->addValue->(Ljava/lang/Number;Ljava/lang/Comparable;Ljava/lang/Comparable;)V | 0.846303502 |
| 14 | org.jfree.data.category.AbstractCategoryDataset-><init>->()V | 0.820754717 |
| 15 | org.jfree.data.category.AbstractCategoryDataset->setSelectionState->(Lorg/jfree/data/category/CategoryDatasetSelectionState;)V | 0.820754717 |

图 1　可疑程序模块推荐列表 $M$ 实例

# 2.1.计算可疑分数阈值

可疑分数阈值作用：过滤掉那些不太可能含有缺陷的程序模块，对剩余模块再进行语句粒度的定位分析。

$$\sigma = \frac{1}{k} \times \sum_{i=1}^{k} \mathrm{Tarantula}(m_i)$$

# 2.2.构建交叉表

交叉表是统计学中常用的一种分类汇总表，交叉表分析通常用于研究两个或多个分类变量之间的关系。

Tab. 1 Crosstab $Q$ of program statement $s_i$

| 运行结果 | 语句覆盖情况 | | |
|---|---|---|---|
| | 覆盖语句 $s_i$ | 未覆盖语句 $s_i$ | 合计 |
| 运行成功 | $N_{CS(s_i)}$ | $N_{US(s_i)}$ | $N_S$ |
| 运行失败 | $N_{CF(s_i)}$ | $N_{UF(s_i)}$ | $N_F$ |
| 合计 | $N_{C(s_i)}$ | $N_{U(s_i)}$ | $N$ |

# 2.3.生成可疑语句推荐列表

- $$\chi^2(s_i)=\frac{(N_{CF(s_i)}-E_{CF(s_i)})^2}{E_{CF(s_i)}}+\frac{(N_{CS(s_i)}-E_{CS(s_i)})^2}{E_{CS(s_i)}}+\frac{(N_{UF(s_i)}-E_{UF(s_i)})^2}{E_{UF(s_i)}}+\frac{(N_{US(s_i)}-E_{US(s_i)})^2}{E_{US(s_i)}}$$

- $$\phi(s_i)=\frac{N_{CF(s_i)}/N_F}{N_{CS(s_i)}/N_S}$$

- $$\text{crosstab}(s_1)=\begin{cases}\chi^2(s_i) & \phi(s_i)>1\\ 0 & \phi(s_i)=1\\ -\chi^2(s_i) & \phi(s_i)<1\end{cases}$$

# 2.3.生成可疑语句推荐列表

| | | | |
|---|---|---|---|
| 1 | org.jfree.chart.renderer.category.AbstractCategoryItemRenderer->getLegendItems->()Lorg/jfree/chart/LegendItemCollection; | 1793 | 436 |
| 2 | org.jfree.chart.plot.CategoryPlot->setRenderer->(Lorg/jfree/chart/renderer/category/CategoryItemRenderer;)V | 1613 | 86.39816092 |
| 3 | org.jfree.chart.plot.CategoryPlot->setRenderer->(Lorg/jfree/chart/renderer/category/CategoryItemRenderer;)V | 1614 | 86.39816092 |
| 4 | org.jfree.chart.plot.CategoryPlot->setRenderer->(ILorg/jfree/chart/renderer/category/CategoryItemRenderer;Z)V | 1665 | 53.62298851 |
| 5 | org.jfree.chart.plot.CategoryPlot->setRenderer->(ILorg/jfree/chart/renderer/category/CategoryItemRenderer;Z)V | 1667 | 53.62298851 |
| 6 | org.jfree.chart.plot.CategoryPlot->setRenderer->(ILorg/jfree/chart/renderer/category/CategoryItemRenderer;Z)V | 1672 | 53.62298851 |
| 7 | org.jfree.chart.plot.CategoryPlot-><init>->()V | 567 | 30.21215107 |
| 8 | org.jfree.chart.plot.CategoryPlot-><init>->()V | 568 | 30.21215107 |
| 9 | org.jfree.chart.plot.CategoryPlot->setDataset->(Lorg/jfree/data/category/CategoryDataset;)V | 1339 | 28.13118774 |
| 10 | org.jfree.chart.plot.CategoryPlot->setDataset->(Lorg/jfree/data/category/CategoryDataset;)V | 1340 | 28.13118774 |
| 11 | org.jfree.chart.plot.CategoryPlot->setDataset->(ILorg/jfree/data/category/CategoryDataset;)V | 1358 | 18.86144201 |
| 12 | org.jfree.chart.plot.CategoryPlot->setDataset->(ILorg/jfree/data/category/CategoryDataset;)V | 1367 | 17.9978011 |
| 13 | org.jfree.chart.renderer.category.LineAndShapeRenderer-><init>->()V | 201 | 14.60492611 |
| 14 | org.jfree.chart.renderer.category.LineAndShapeRenderer-><init>->()V | 202 | 14.60492611 |
| 15 | org.jfree.chart.plot.CategoryPlot->getIndexOf->(Lorg/jfree/chart/renderer/category/CategoryItemRenderer;)I | 1727 | 10.49776165 |

图 2　可疑语句推荐列表 $T$ 实例

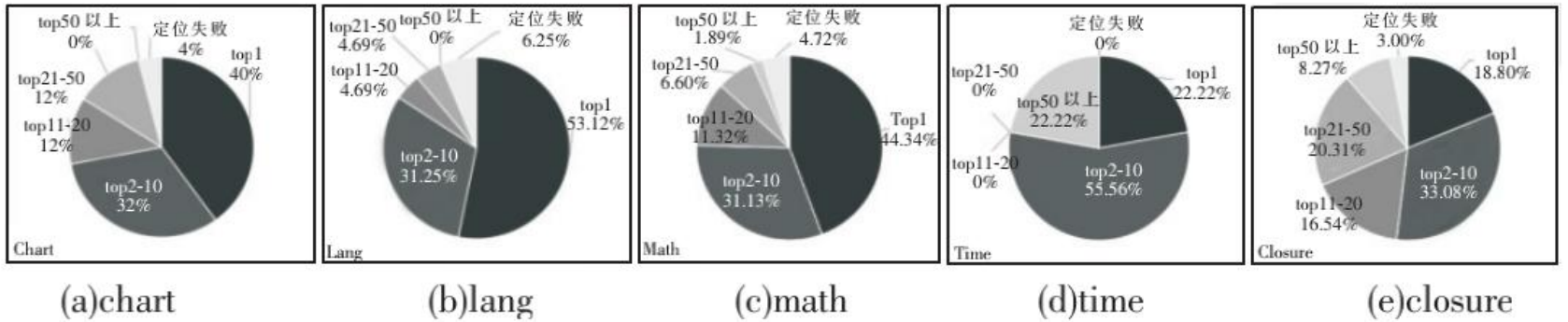
# 实验

实验设计 | 评价指标 | 实验结果 | 结果分析

**Defects4J**：真实缺陷库

**Cobertura**：测试覆盖率工具

**Top N**




(a)chart (b)lang (c)math (d)time (e)closure

图 3　Defects4J 中各项目定位结果分布

# 优点

有效地减少缺陷定位的时间花销

Tab. 3 Comparison of fault localization time

| 项目名称（平均代码行/平均测试用例数） | 本文方案与其他算法相比，定位时间增加/减少 | | | | |
|---|---|---|---|---|---|
| | Jasscard | Ochiai | DStar$^2$ | Op2 | EP$^3$ |
| chart（4187/230） | ↓ 10.44% | ↓ 10.78% | ↓ 10.30% | ↓ 11.15% | ↓ 10.65% |
| lang（814/176） | ↓ 7.25% | ↓ 5.83% | ↓ 8.30% | ↓ 9.65% | ↓ 8.37% |
| math（2266/215） | ↓ 9.22% | ↓ 8.42% | ↓ 9.99% | ↓ 10.74% | ↓ 10.35% |
| time（5218/2606） | ↓ 10.61% | ↓ 10.97% | ↓ 10.77% | ↓ 11.36% | ↓ 11.22% |
| closure（16483/1540） | ↓ 11.75% | ↓ 11.74% | ↓ 11.93% | ↓ 11.84% | ↓ 12.25% |

# 缺点

a）主要针对单缺陷项目，未来可以进一步地研究多缺陷定位问题；

b）采用轻量级程序频谱构造方法，在未来研究中可考虑采用重量级程序频谱构造方法，从而提高定位效果。

THANK YOU FOR YOUR LISTENING.

# 谢谢您的聆听