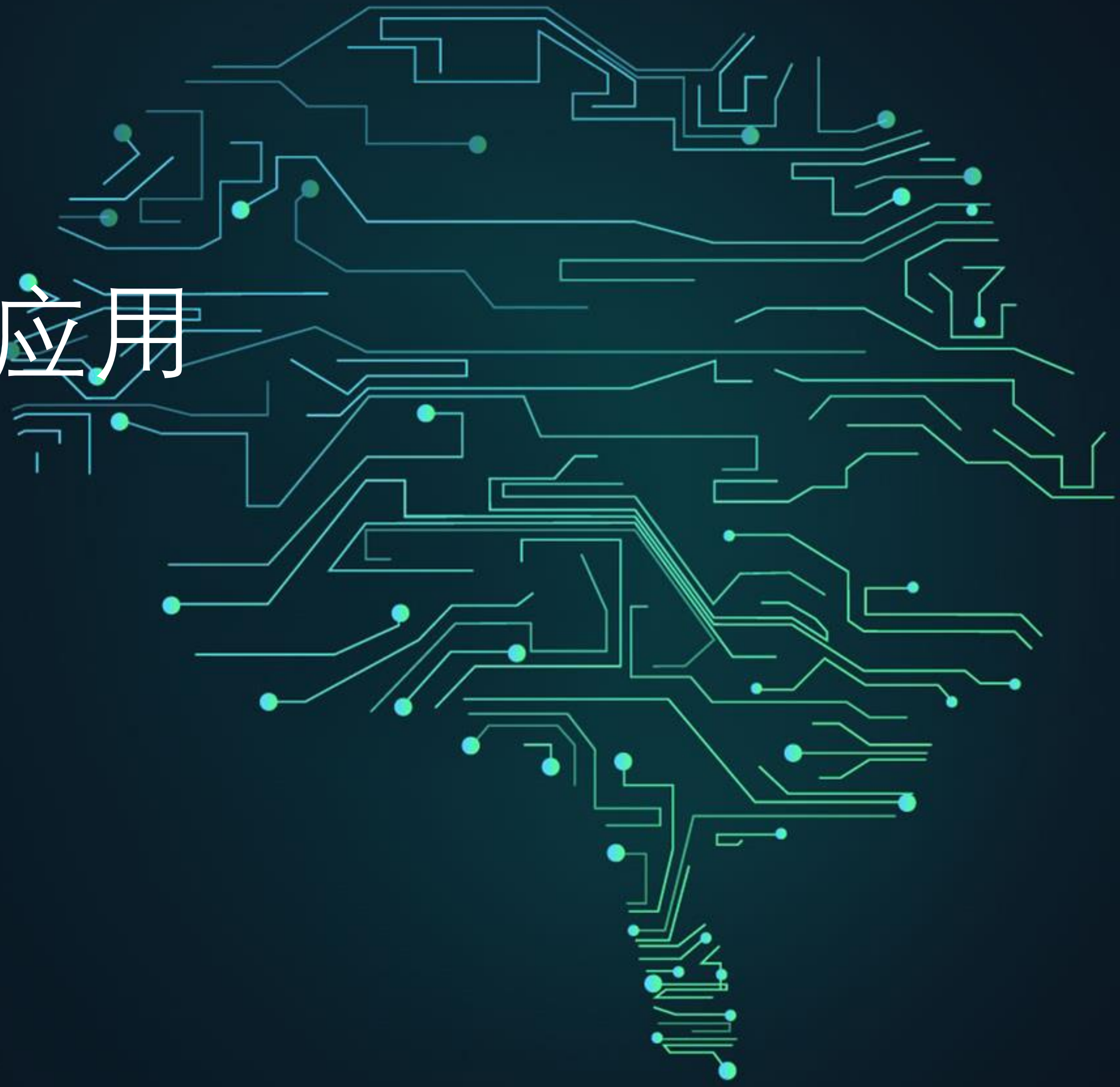
人工智能在Bug定位中的应用



主讲人：邱化峰

饿了么 新零售新餐饮 测试开发专家

NCTS 第二届 中国云测试行业峰会

Testin 助力产业智能化

目 录

Contents

01、工具介绍

02、原理和方法

03、应用介绍

01

章节 PART

工具介绍

工具介绍-Bug定位的技术

1.程序频谱

2.基于变体

3.程序切片

4.堆栈跟踪

5.上下文切换

6.信息检索

7.基于历史

8.机器学习

工具介绍-影响Bug的因素

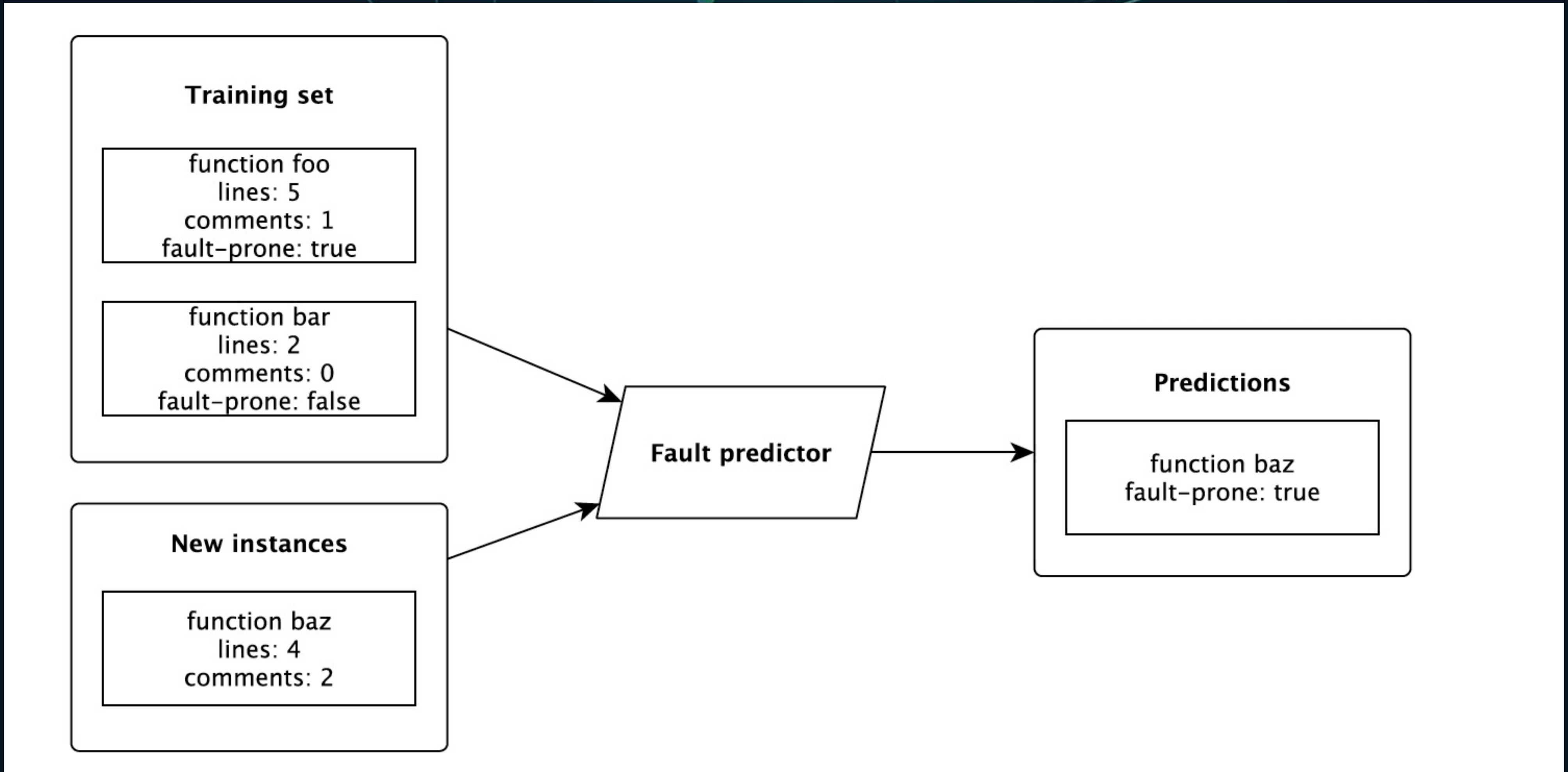
- a. JavaNCSS：有效行数，圈复杂度，是否符合javadoc的规范
- b. ckjm：WMC，DIT，NOC，CBO，RFO，RFC，LCOM
- c. Jdepend：包的设计质量
- d. JarAnalyzer：jar之间依赖关系
- e.变量个数及参数个数
- f. Dependmeter：依赖关系
- J.运算符的个数

02

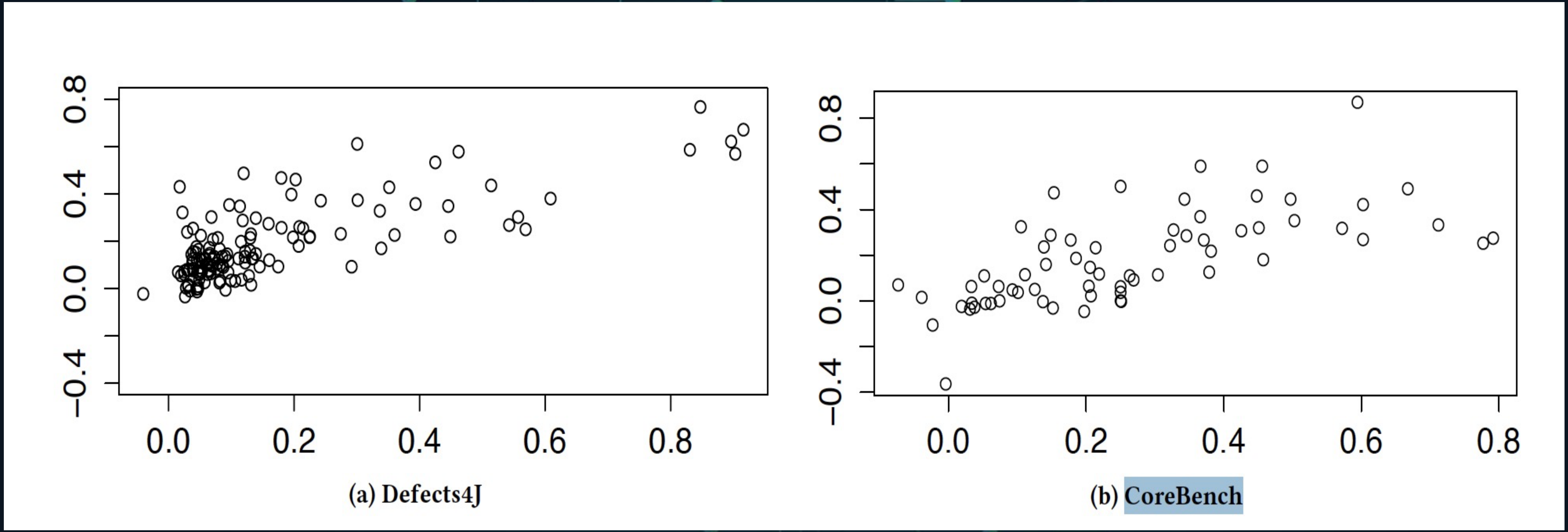
章节 PART

原理和方法

原理和方法-通过代码复杂度McCabe



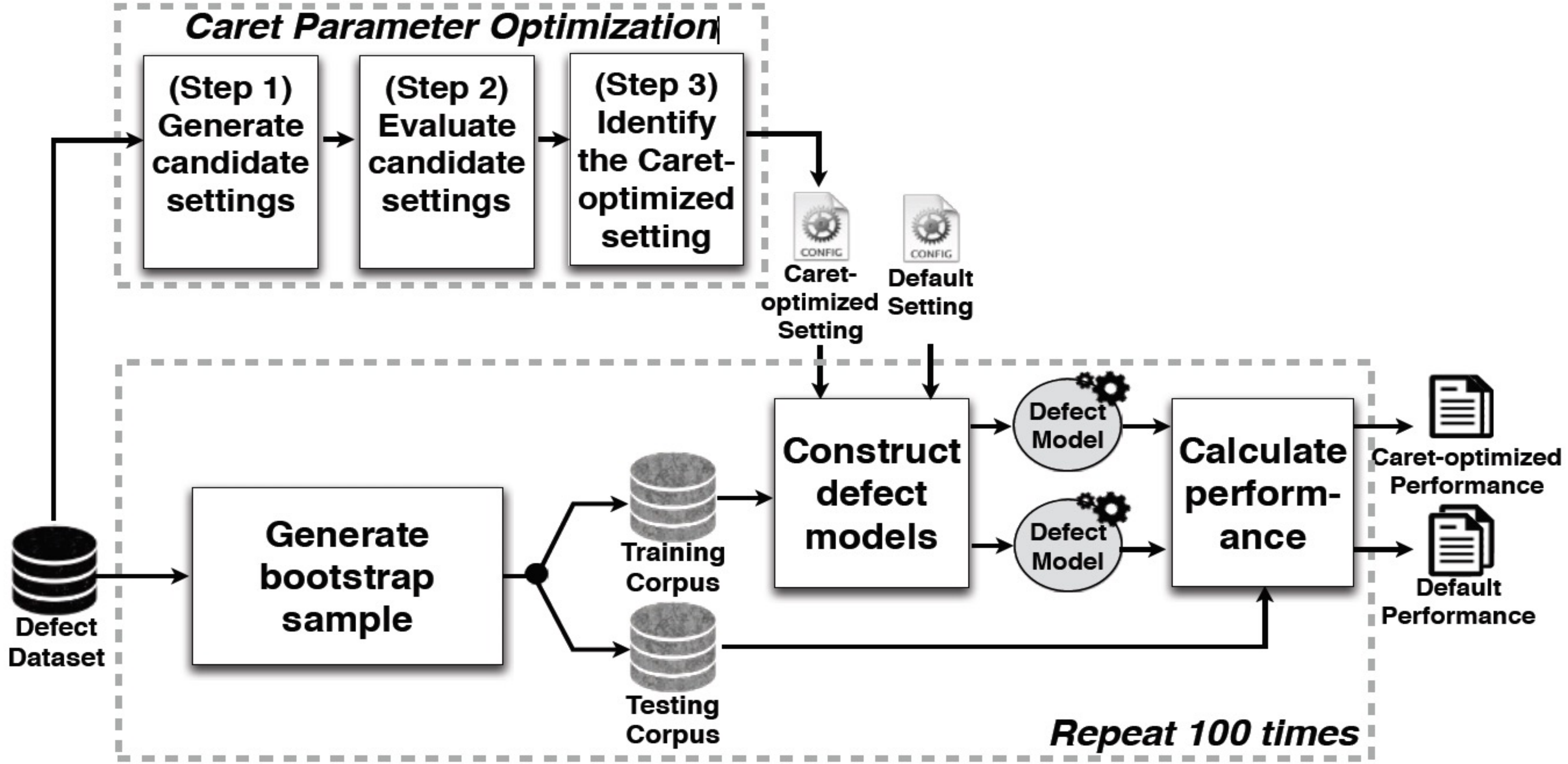
原理和方法-变异



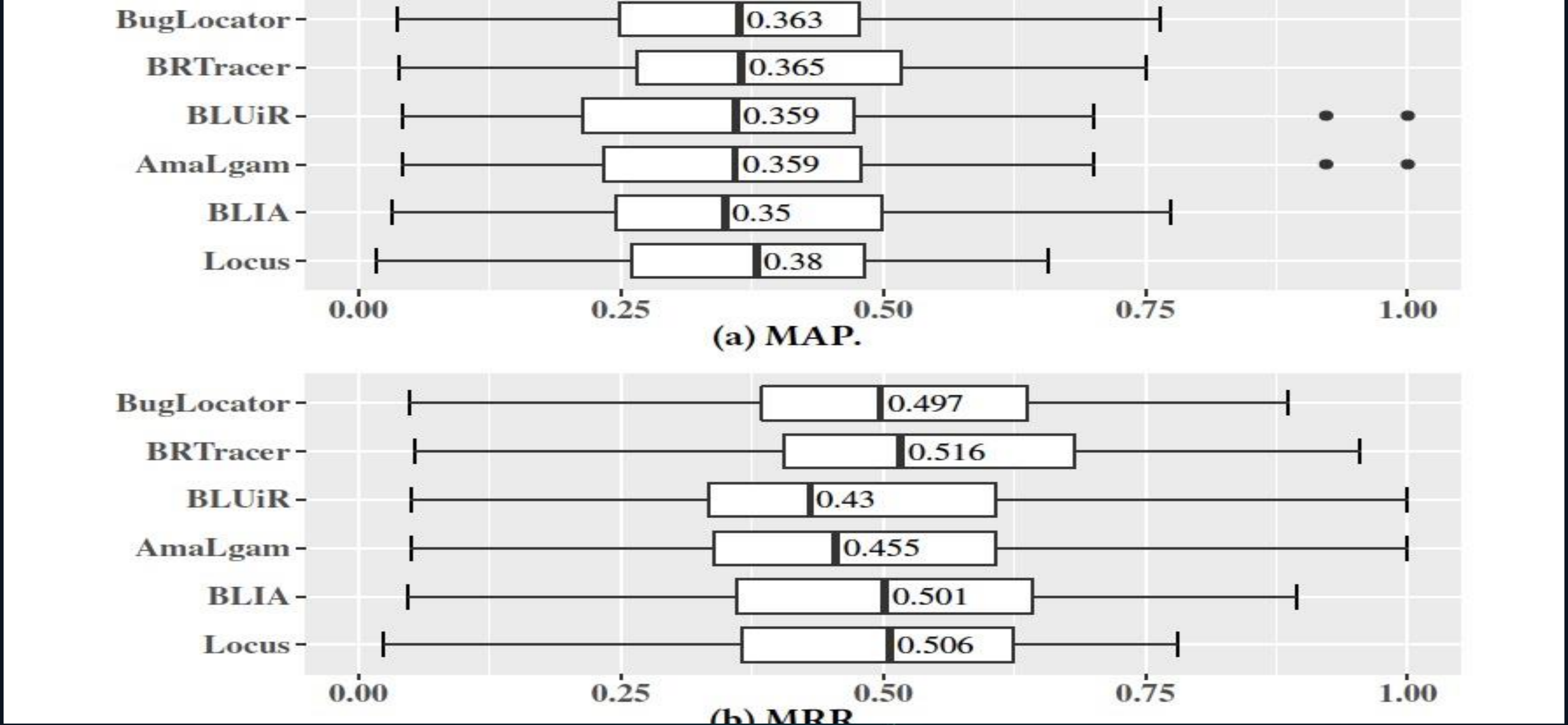
(a) Defects4J (b) CoreBench

变异分数跟错误检查存在强关联性

原理和方法

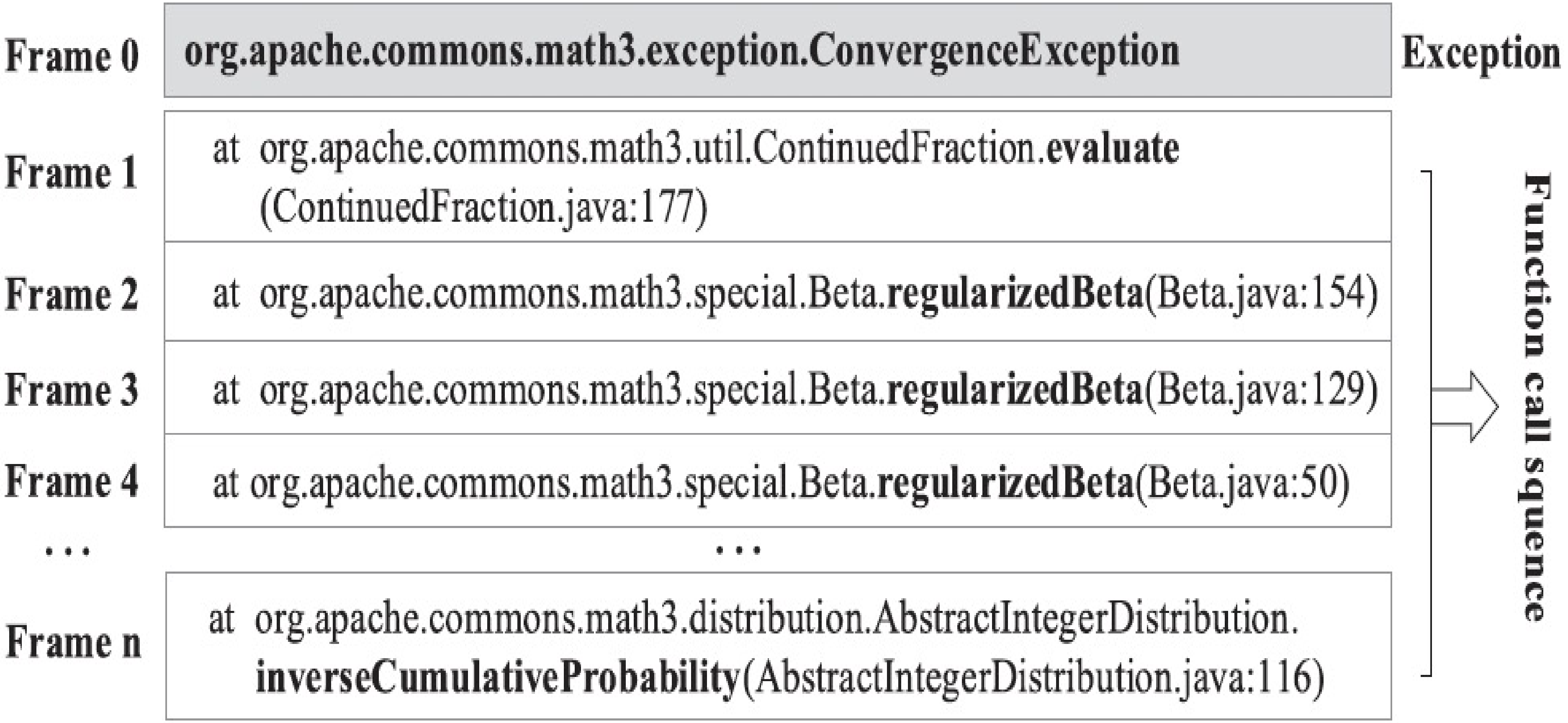

Caret Parameter Optimization
(Step 1) Generate candidate settings
(Step 2) Evaluate candidate settings
(Step 3) Identify the Caret-optimized setting
CONFIG
Caret-optimized Setting
CONFIG
Default Setting
Defect Dataset
Generate bootstrap sample
Training Corpus
Testing Corpus
Construct defect models
Defect Model
Defect Model
Calculate perform-ance
Caret-optimized Performance
Default Performance
Repeat 100 times

原理和方法-信息检索



https://github.com/exatoa/Bench4BL

原理和方法-Stack Trace



Frame 0 org.apache.commons.math3.exception.ConvergenceException

Frame 1 at org.apache.commons.math3.util.ContinuedFraction.**evaluate** (ContinuedFraction.java:177)

Frame 2 at org.apache.commons.math3.special.Beta.**regularizedBeta**(Beta.java:154)

Frame 3 at org.apache.commons.math3.special.Beta.**regularizedBeta**(Beta.java:129)

Frame 4 at org.apache.commons.math3.special.Beta.**regularizedBeta**(Beta.java:50)

. . . . . .

Frame n at org.apache.commons.math3.distribution.AbstractIntegerDistribution. **inverseCumulativeProbability**(AbstractIntegerDistribution.java:116)

Fig. 1. Stack trace of Bug 718 in the project of Apache Commons Math.

原理和方法-Stack Trace

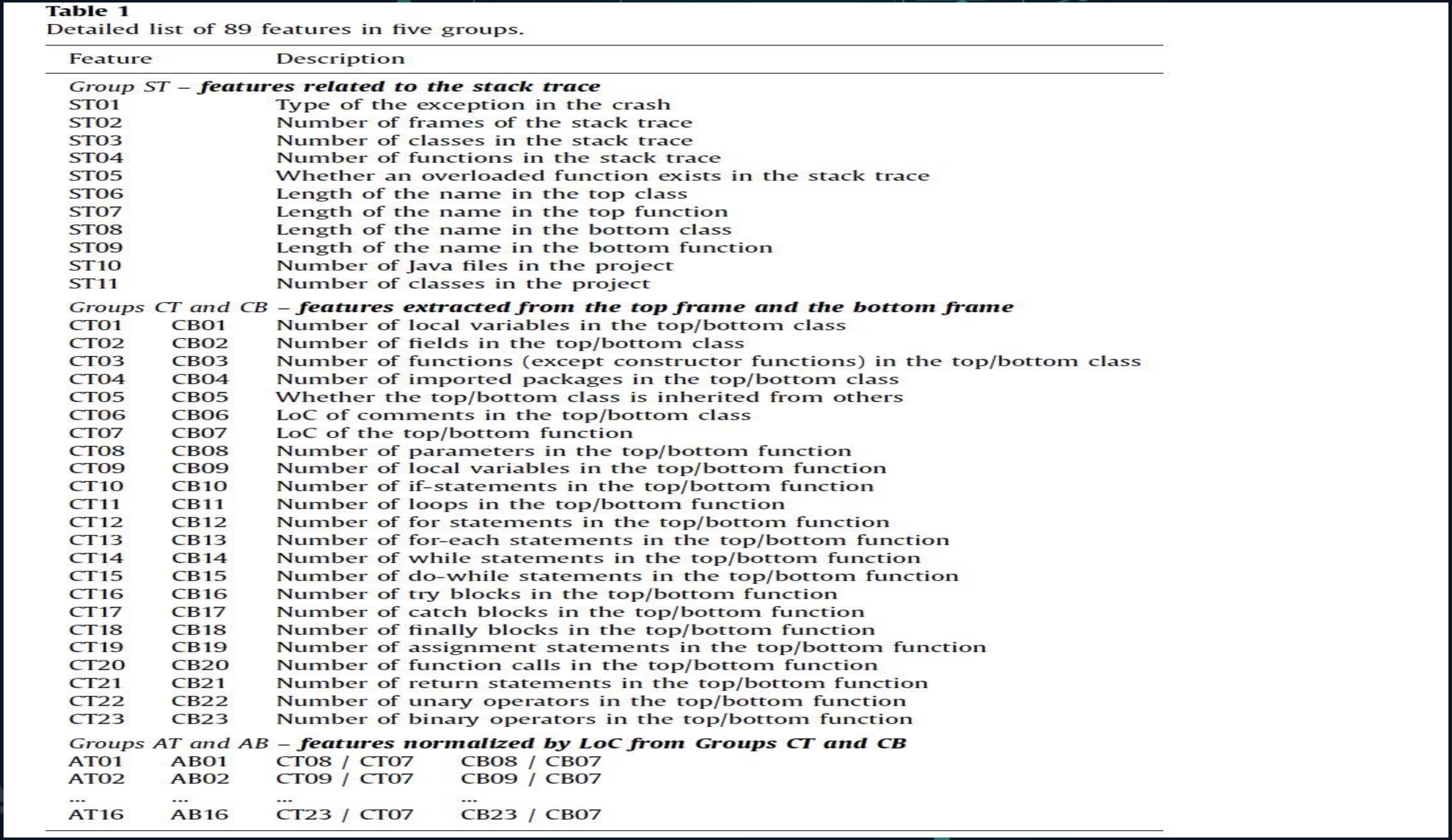


Table 1
Detailed list of 89 features in five groups.

Feature		Description	
*Group ST – **features related to the stack trace***			
ST01		Type of the exception in the crash	
ST02		Number of frames of the stack trace	
ST03		Number of classes in the stack trace	
ST04		Number of functions in the stack trace	
ST05		Whether an overloaded function exists in the stack trace	
ST06		Length of the name in the top class	
ST07		Length of the name in the top function	
ST08		Length of the name in the bottom class	
ST09		Length of the name in the bottom function	
ST10		Number of Java files in the project	
ST11		Number of classes in the project	
*Groups CT and CB – **features extracted from the top frame and the bottom frame***			
CT01	CB01	Number of local variables in the top/bottom class	
CT02	CB02	Number of fields in the top/bottom class	
CT03	CB03	Number of functions (except constructor functions) in the top/bottom class	
CT04	CB04	Number of imported packages in the top/bottom class	
CT05	CB05	Whether the top/bottom class is inherited from others	
CT06	CB06	LoC of comments in the top/bottom class	
CT07	CB07	LoC of the top/bottom function	
CT08	CB08	Number of parameters in the top/bottom function	
CT09	CB09	Number of local variables in the top/bottom function	
CT10	CB10	Number of if-statements in the top/bottom function	
CT11	CB11	Number of loops in the top/bottom function	
CT12	CB12	Number of for statements in the top/bottom function	
CT13	CB13	Number of for-each statements in the top/bottom function	
CT14	CB14	Number of while statements in the top/bottom function	
CT15	CB15	Number of do-while statements in the top/bottom function	
CT16	CB16	Number of try blocks in the top/bottom function	
CT17	CB17	Number of catch blocks in the top/bottom function	
CT18	CB18	Number of finally blocks in the top/bottom function	
CT19	CB19	Number of assignment statements in the top/bottom function	
CT20	CB20	Number of function calls in the top/bottom function	
CT21	CB21	Number of return statements in the top/bottom function	
CT22	CB22	Number of unary operators in the top/bottom function	
CT23	CB23	Number of binary operators in the top/bottom function	
*Groups AT and AB – **features normalized by LoC from Groups CT and CB***			
AT01	AB01	CT08 / CT07	CB08 / CB07
AT02	AB02	CT09 / CT07	CB09 / CB07
...	...	...	...
AT16	AB16	CT23 / CT07	CB23 / CB07

原理和方法-Stack Trace

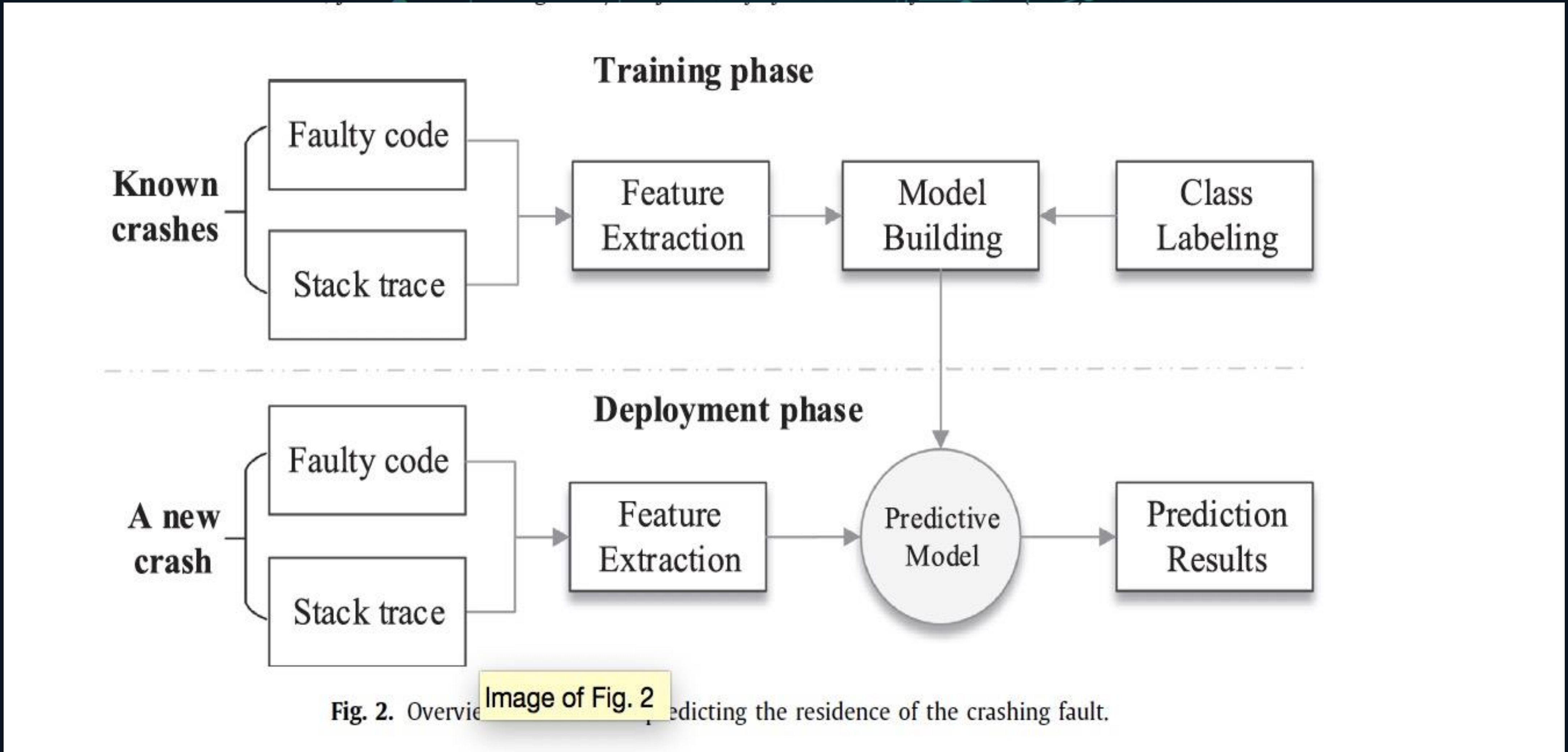


Fig. 2. Overvie Image of Fig. 2 edicting the residence of the crashing fault.

原理和方法

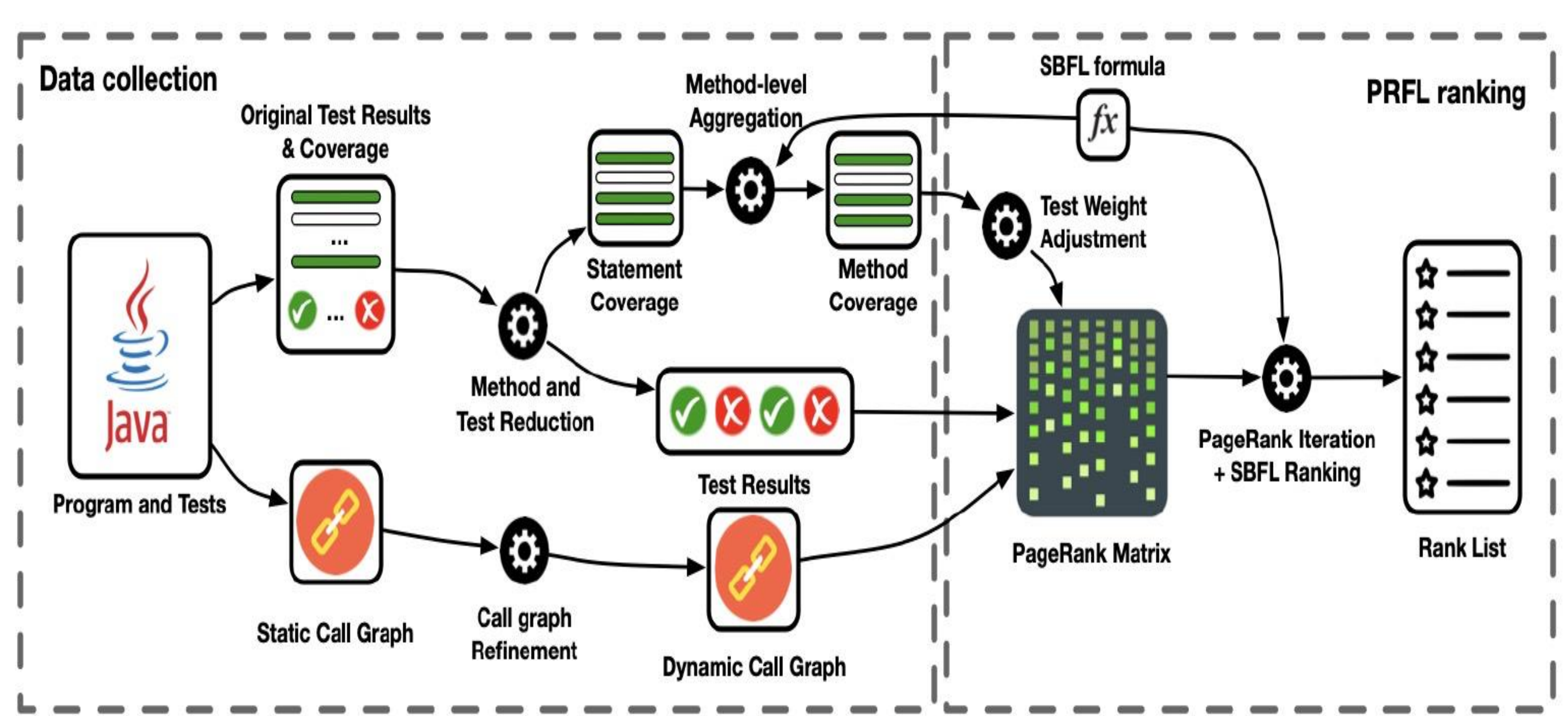


Fig. 5: Framework of the Proposed Approach

03

章节 PART

应用介绍

应用介绍-发现了什么

1.程序频谱是一种有效的Bug定位的方法

2.Stack Trace对于Crash的定位是很有效的

3.错误的样本集目前仍然不够大

4.参考的因素和样本在持续的增加

5.算法的多样性

应用介绍-场景应用

1.辅助开发和测试定位Bug，比如通过Bug Report 定位Bug在某个函数体内

2.Bug 的重复检测

3.Bug的自动assign

4.Case的等级分类

应用介绍-参考链接

- 1. An Empirical Study of Fault Localization Families and Their Combinations

2. "Does the Fault Reside in a Stack Trace?

- 3. Bench4BL: Reproducibility Study on the Performance of IR-Based Bug Localization

- 4. Bench4BL: Reproducibility Study on the Performance of IR-Based Bug Localization

THANK YOU

感谢聆听

Testin
助力产业智能化

NCTS 第二届 中国云测试行业峰会