AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

AI驱动软件研发
全面进入数字化时代

中国·深圳 11.24-25

AI+
software
Development
Digital
summit

华为云智能单元测试用例生成实践

周建祎 华为云PaaS技术创新Lab

科技生态圈峰会 + 深度研习

——1000 + 技术团队的共同选择

KEYLINK ing

K+峰会

K+峰会 深圳站

K+ 全球软件研发行业创新峰会

会议时间：2024.05.24-25

K+峰会 上海站

K+ 全球软件研发行业创新峰会

会议时间：2024.09.20-21

AiDD 峰会

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2023.11.24-25

AiDD峰会 北京站

AI+ 软件研发数字峰会

会议时间：2024.07.19-20

AiDD峰会 深圳站

AI+ 软件研发数字峰会

会议时间：2024.11.15-16

演讲嘉宾

周建祎

高级工程师

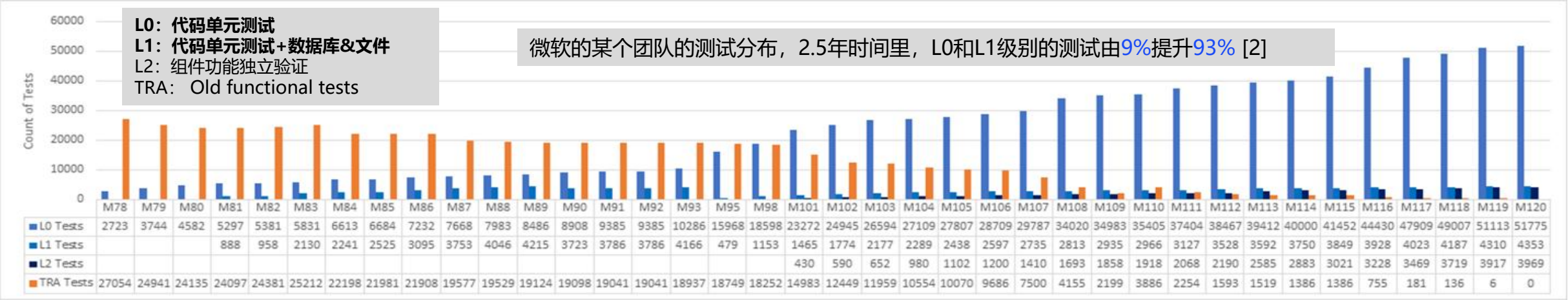
华为云PaaS技术创新Lab算法工程师，本硕毕业于北京航空航天大学，博士毕业于北京大学计算机学院。博士阶段的研究方向包括测试执行优化和深度学习系统测试，在CCF-A类会议或期刊共发表4篇论文。22年获得博士学位入职华为。入职以后围绕单元测试用例自动生成和自动演化等方向展开工作。近期带领团队探索基于大模型生成单元测试用例的实践。

目录
CONTENTS

PART 01
背景

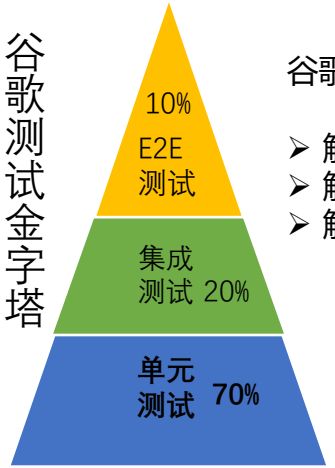
单元测试用例自动生成

单元测试的重要性：在软件开发阶段，越早发现缺陷，修复缺陷和维护系统的代价越低[1]。单元测试旨在开发阶段通过测试用例发现软件缺陷，可以有效降低系统维护的代价。业界也逐渐关注测试左移。



	M78	M79	M80	M81	M82	M83	M84	M85	M86	M87	M88	M89	M90	M91	M92	M93	M95	M98	M101	M102	M103	M104	M105	M106	M107	M108	M109	M110	M111	M112	M113	M114	M115	M116	M117	M118	M119	M120
L0 Tests	2723	3744	4582	5297	5381	5831	6613	6684	7232	7668	7983	8486	8908	9385	9385	10286	15968	18598	23272	24945	26594	27109	27807	28709	29787	34020	34983	35405	37404	38467	39412	40000	41452	44430	47909	49007	51113	51775
L1 Tests				888	958	2130	2241	2525	3095	3753	4046	4215	3723	3786	3786	4166	479	1153	1465	1774	2177	2289	2438	2597	2735	2813	2935	2966	3127	3528	3592	3750	3849	3928	4023	4187	4310	4353
L2 Tests																			430	590	652	980	1102	1200	1410	1693	1858	1918	2068	2190	2585	2883	3021	3228	3469	3719	3917	3969
TRA Tests	27054	24941	24135	24097	24381	25212	22198	21981	21908	19577	19529	19124	19098	19041	19041	18937	18749	18252	14983	12449	11959	10554	10070	9686	7500	4155	2199	3886	2254	1593	1519	1386	1386	755	181	136	6	0

谷歌测试金字塔



谷歌的实践发现，解决不同阶段缺陷的代价存在差异：

- 解决单个单元测试的缺陷约**5美元**
- 解决单个集成测试的缺陷约500美元
- 解决单个系统测试的缺陷高达5000美元

如何将单元测试变得更加智能？

人工编写/更新单元测试

自动生成
单元测试

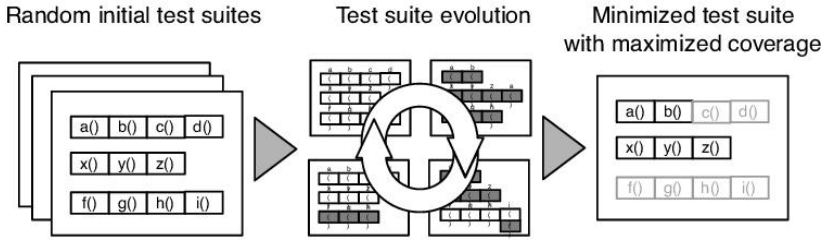
[2] https://learn.microsoft.com/en-us/devops/develop/shift-left-make-testing-fast-reliable

PART 02

基于软件分析的单测自动生成技术

基于软件分析技术的单测自动生成技术

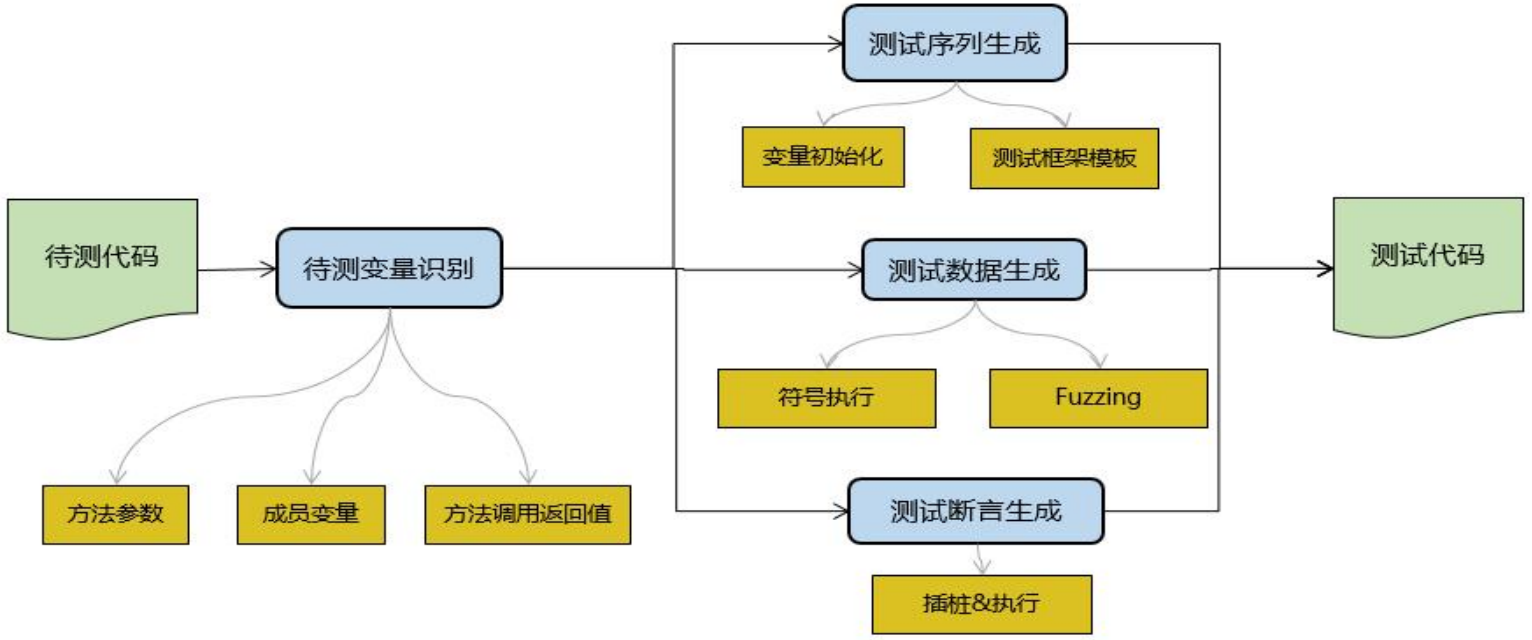
- 基于搜索的测试用例生成技术，如Evosuite [1]
  - 目标为生成测试序列，能够生成基本类型的测试输入
- 基于符号执行的测试用例生成技术
  - 存在路径爆炸或依赖于动态分析
- 基于模糊测试的测试用例生成技术



Evosuite 流程图

[1] Gordon Fraser and Andrea Arcuri. 2011. EvoSuite: automatic test suite generation for object-oriented software. ESEC/FSE '11

基于对象构造分析的单测生成技术

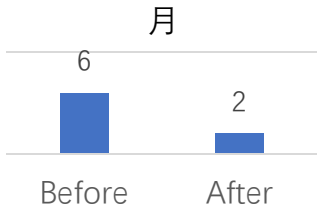


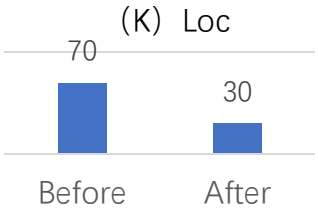
- 支持编程语言：Java
- 支持2种构建工具：maven，gradle
- 支持4种测试框架：JUnit4, JUnit5, spring-boot-starter-test，spock
- 支持2种mock框架：mockito，powermock
- 辅助生成单元测试用例代码，支持3种生成策略

效果总结

工具能力落地覆盖全域100+代码仓，80+开发者，涉及xxx、xxx、xxx、xxx等特性。2个月时间，团队代码覆盖率目标达成xx%，其中单测生成工具自生成代码覆盖率40%，员工手工编码量减少57%。

注：UT 恒定xxK代码量，工具生成代码量57%，人工编码43%





- **目标达成时间缩短67%**
- **员工UT编码量减少57%**
- **≈40Kloc**

单元测试用例演化

➢相似代码模式

> 概述：变更代码与已有代码存在相似性，对应测试用例的修改也存在相似性

➢调用演化

> 概述：被测代码中的调用发生变化（比如，调用当前项目其他方法、调用其他三方库API等），对应测试用例调用需要演化

➢废弃方法演化

> 概述：被测代码的某个函数被废弃，测试用例中对应的废弃函数调用需要进行演化

➢泛型参数演化

> 概述：被测代码中的参数被修改为泛型，测试用例中对应的参数声明需要进行演化

单元测试用例演化

- Commons-lang

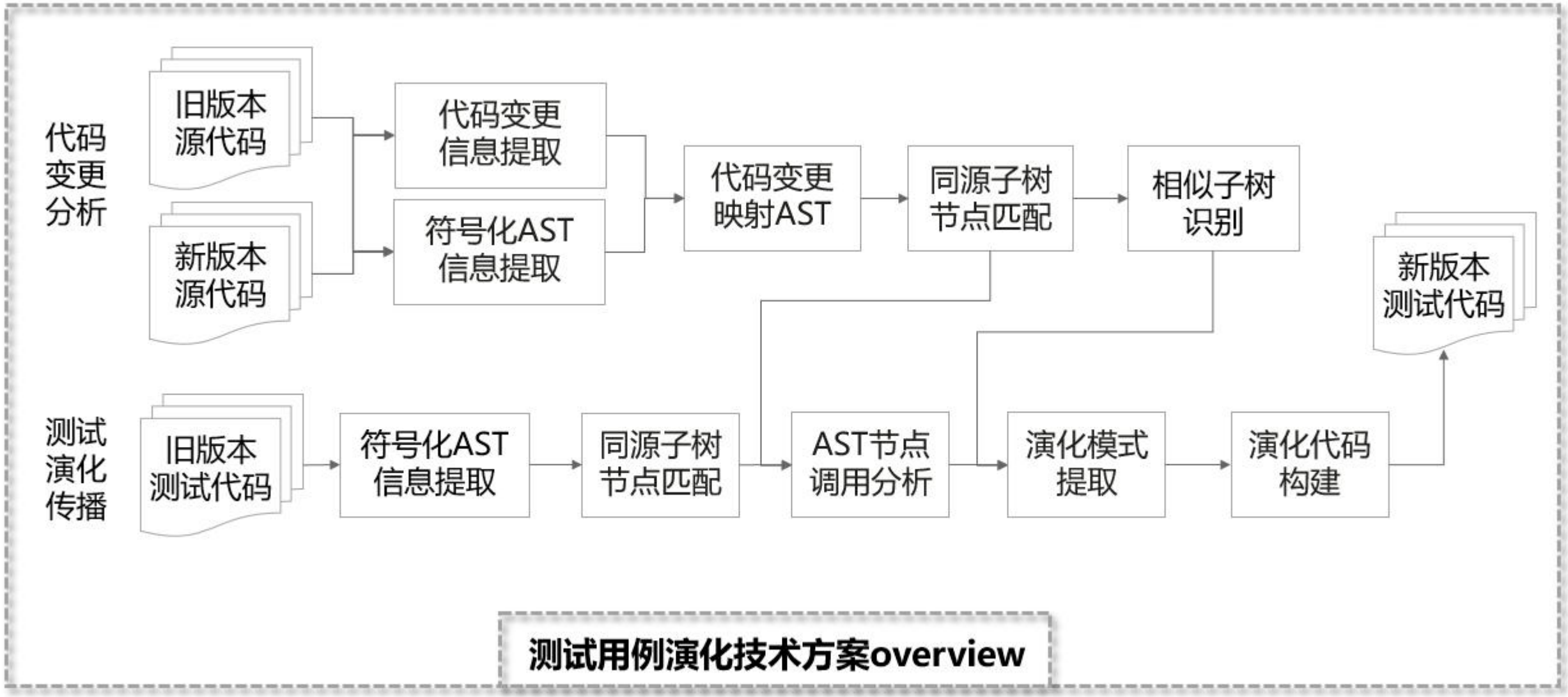
```
public static final boolean IS_JAVA_16 = getJavaVersionMatches("16");

/**
 * Is {@code true} if this is Java version 17 (also 17.x versions).
 * <p>
 * The field will return {@code false} if {@link #JAVA_VERSION} is {@code null}.
 * </p>
 *
 * @since 3.13.0
 */
public static final boolean IS_JAVA_17 = getJavaVersionMatches("17");

/**
 * Is {@code true} if this is Java version 18 (also 18.x versions).
 * <p>
 * The field will return {@code false} if {@link #JAVA_VERSION} is {@code null}.
 * </p>
 *
 * @since 3.13.0
 */
public static final boolean IS_JAVA_18 = getJavaVersionMatches("18");
```

```
public void test_IS_JAVA() {
    final String javaVersion = SystemUtils.JAVA_VER
    if (javaVersion == null) {
        assertFalse(SystemUtils.IS_JAVA_1_1);
        assertFalse(SystemUtils.IS_JAVA_1_2);
        assertFalse(SystemUtils.IS_JAVA_1_3);
        assertFalse(SystemUtils.IS_JAVA_1_4);
        assertFalse(SystemUtils.IS_JAVA_1_5);
        assertFalse(SystemUtils.IS_JAVA_1_6);
        assertFalse(SystemUtils.IS_JAVA_1_7);
        assertFalse(SystemUtils.IS_JAVA_1_8);
        assertFalse(SystemUtils.IS_JAVA_1_9);
        assertFalse(SystemUtils.IS_JAVA_9);
        assertFalse(SystemUtils.IS_JAVA_10);
        assertFalse(SystemUtils.IS_JAVA_11);
        assertFalse(SystemUtils.IS_JAVA_12);
        assertFalse(SystemUtils.IS_JAVA_13);
        assertFalse(SystemUtils.IS_JAVA_14);
        assertFalse(SystemUtils.IS_JAVA_15);
        assertFalse(SystemUtils.IS_JAVA_16);
        assertFalse(SystemUtils.IS_JAVA_17);
        assertFalse(SystemUtils.IS_JAVA_18);
    } else if (javaVersion.startsWith("1.8")) {
        assertFalse(SystemUtils.IS_JAVA_1_1);
        assertFalse(SystemUtils.IS_JAVA_1_2);
        assertFalse(SystemUtils.IS_JAVA_1_3);
        assertFalse(SystemUtils.IS_JAVA_1_4);
        assertFalse(SystemUtils.IS_JAVA_1_5);
        assertFalse(SystemUtils.IS_JAVA_1_6);
        assertFalse(SystemUtils.IS_JAVA_1_7);
        assertTrue(SystemUtils.IS_JAVA_1_8);
        assertFalse(SystemUtils.IS_JAVA_1_9);
        assertFalse(SystemUtils.IS_JAVA_9);
        assertFalse(SystemUtils.IS_JAVA_10);
        assertFalse(SystemUtils.IS_JAVA_11);
        assertFalse(SystemUtils.IS_JAVA_12);
        assertFalse(SystemUtils.IS_JAVA_13);
        assertFalse(SystemUtils.IS_JAVA_14);
        assertFalse(SystemUtils.IS_JAVA_15);
        assertFalse(SystemUtils.IS_JAVA_16);
        assertFalse(SystemUtils.IS_JAVA_17);
        assertFalse(SystemUtils.IS_JAVA_18);
    } else if (javaVersion.startsWith("9")) {
        assertFalse(SystemUtils.IS_JAVA_1_1);
```

基于相似代码模式的单元测试用例演化技术



测试用例演化技术方案overview

PART 03
现有技术痛点

现有技术面临的痛点

新增测试用例生成

- 可读性
  - 传统技术生成的单测可读性差，解释能力欠缺
- 复杂数据类型输入
  - 实际业务中涉及许多复杂类型数据（e.g., 网络通讯数据、半结构化数据），传统技术难以生成这些复杂类型的有效测试输入。
- 多语言&多框架拓展
  - 不同语言、开发框架、测试框架差异较大，传统单测生成技术拓展至其他场景代价大。
- 性能
  - 在复杂工程项目和超大型项目中应用符号执行技术、搜索技术存在性能瓶颈。
- 断言
  - 现有技术生成测试断言效果有待提升。

已有测试用例演化

- 编译错误
  - 软件迭代过程中变量类型和接口发生变化，导致原有单测存在编译错误
- 运行时错误
  - 业务逻辑发生变化导致断言失效
- 测试用例冗余
  - 业务逻辑发生变化导致原有单测没有实际覆盖语句

示例：接口发生变化导致单测失效

```
static MediaType createImageType(String subtype) {
  return create(IMAGE_TYPE, subtype);
}
```
Existing method in v_i

```
static MediaType createFontType(String subtype) {
  return create(FONT_TYPE, subtype);
}
```
Added Method in v_{i+1}

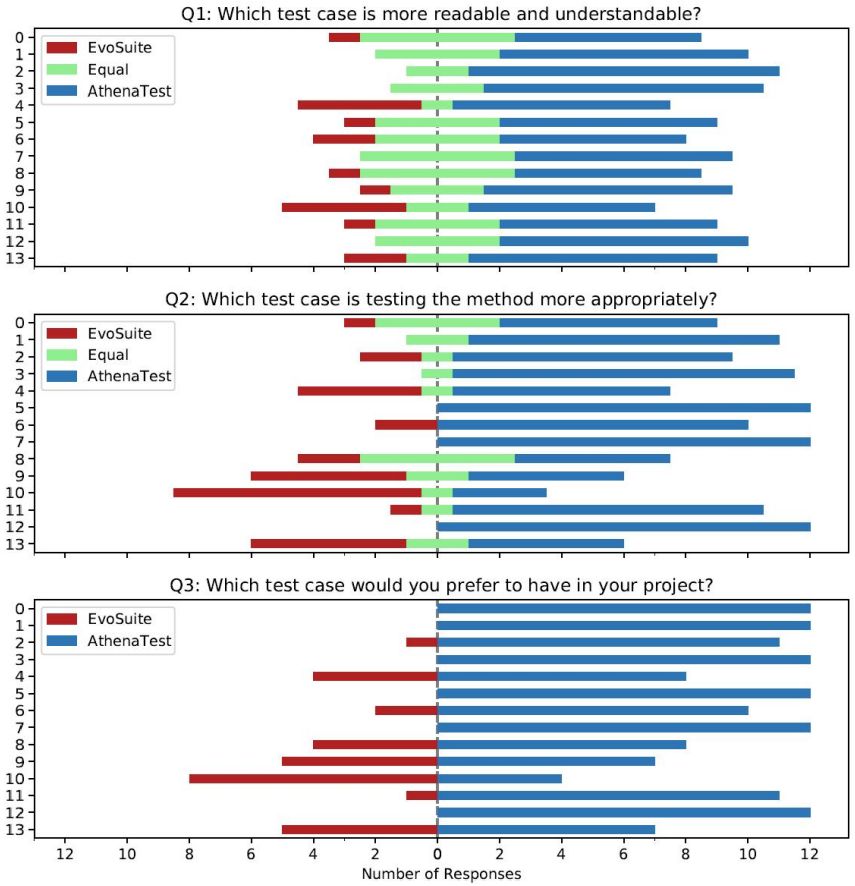
```
public void testCreateImageType() {
  MediaType newType = MediaType.createImageType("yams");
  assertEquals( expected: "image", newType.type());
  assertEquals( expected: "yams", newType.subtype());
}
```
Existing Testcase in v_i

```
public void testCreateFontType() {
  MediaType newType = MediaType.createFontType("yams");
  assertEquals( expected: "font", newType.type());
  assertEquals( expected: "yams", newType.subtype());
}
```
Modified Testcase in v_{i+1}

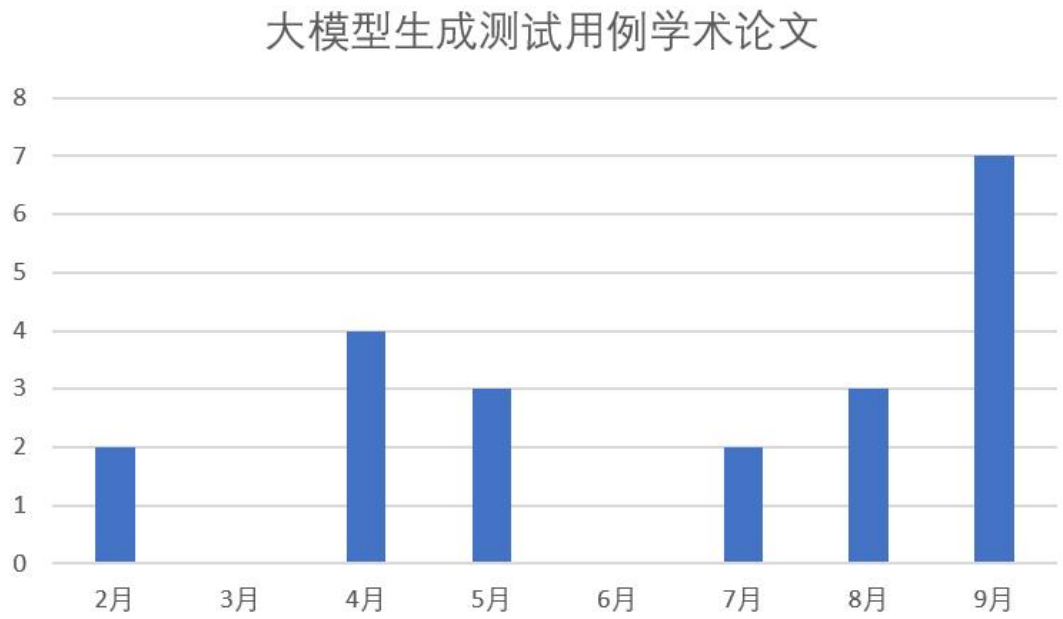
[1] Shimmi, Samiha, and Mona Rahimi. "Patterns of Code-to-Test Co-evolution for Automated Test Suite Maintenance." ICST'2022.

LLM开启单测生成的新热潮

开发者更加认可 LLM生成的单元测试用例



学术界热潮，23年已发表论文20余篇



Tufano, Michele, et al. “Unit test case generation with transformers and focal context.” *MSRA, 2020*.

LLM自动生成单测的实证研究

CORRECTNESS OF GENERATED TESTS

Metrics (%)	ChatGPT	AthenaTest	Evosuite
Syntactical correct	≈ 100.0	54.8	100.0
Success compilation	42.1	18.8	66.8
Passing execution	24.8	14.4	59.7

EFFECTIVENESS OF CHATTESTER

Metrics (%)	ChatGPT	CHATTESTER-	CHATTESTER
Syntactical correct	100.0	100.0	100.0
Success compilation	39.0	50.7	73.3
Passing execution	22.3	29.7	41.0

Category	Detailed Errors	Frequency
Symbol Resolution Error	Cannot find symbol class	1,934
	Cannot find symbol method	471
	Cannot find symbol variable	466
	Cannot find symbol	169
Type Error	Incompatible types	73
	Constructor cannot be applied to given types	46
	Methods cannot be applied to given types	11
Access Error	Private access	75
Abstract Class Initiation Error	Abstract class cannot be instantiated	33
Unsupported Operator	Diamond operator is not supported	15

Category	Detailed Errors	Frequency
Assertion Error	java.lang.AssertionError/ org.opentest4j.AssertionFailedError/ org.junit.ComparisonFailure/ org.junit.internal.ArrayComparisonFailure	148
Runtime Error	java.lang.IllegalArgumentException	6
	java.lang.RuntimeException	5
	java.lang.NullPointerException	3
	others	11

StarCoder 生成单测 缺陷类型

缺陷类型	案例数	比例
缺失调用目标方法	65	21.3%
代码被截断	64	21.0%
参数不匹配	50	16.4%
括号不匹配	35	11.5%
生成有效代码与无效代码	32	10.5%
含未声明非法部分	18	5.9%
生成重复的有效代码	18	5.9%
生成空方法	14	4.6%
生成错误变量名	7	2.3%
引用包名错误	1	0.3%
结尾出现多余符号	1	0.3%

[1] 复旦大学, 23年4月, “ChatTester: No More Manual Tests? Evaluating and Improving ChatGPT for Unit Test Generation”

PART 04

基于大模型的单元测试自动生成实践

基于大模型生成的单元测试用例实例 – GPT4

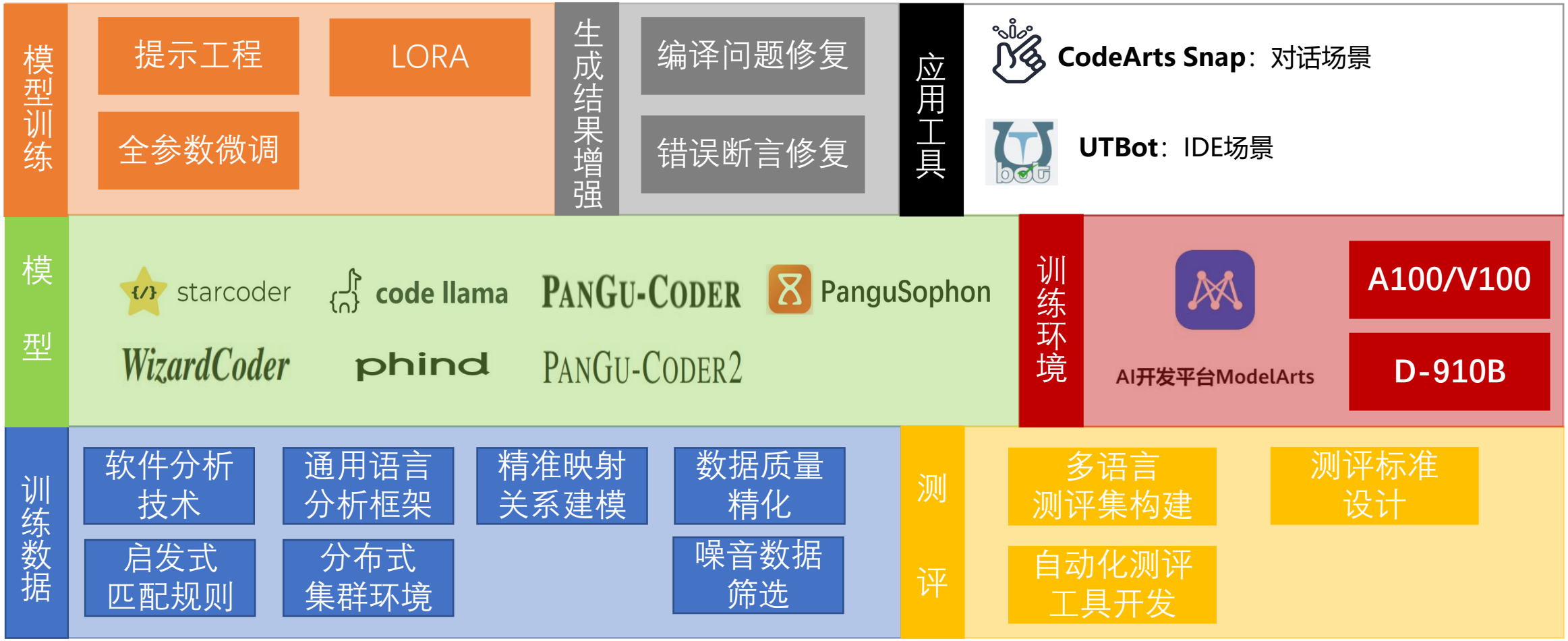
```java
public int findIndex(int[] sortedArray, int target) {
    if (sortedArray == null) {
        throw new RuntimeException("array is null");
    }
    for (int i = 0; i < sortedArray.length; i++) {
        if (sortedArray[i] == target) {
            return i;
        }
    }
    return -1;
}
```

大模型生成单测的优势：

多样的测试场景（包括许多边界情况）
良好的可读性和代码解释
较为准确的断言
充分的测试覆盖

```java
public class ArrayExamplesTest {
    4 usages
    private ArrayExamples arrayExamples = new ArrayExamples();
    no usages
    @Test
    void testFindIndex_withNullArray() {
        // Test Explanation: We need to test when the array is null.
        Exception ex = Assertions.assertThrows(RuntimeException.class, ()
            -> arrayExamples.findIndex( sortedArray: null,  target: 1));
        Assertions.assertEquals( expected: "array is null", ex.getMessage());
    }
    no usages
    @Test
    void testFindIndex_ExistingElement() {
        // Test Explanation: We need to test when the array contains the target element.
        int[] array = {1, 2, 3, 4, 5};
        int target = 3;
        int expected = 2;
        Assertions.assertEquals(expected, arrayExamples.findIndex(array, target));
    }
    no usages
    @Test
    void testFindIndex_NonExistingElement() {
        // Test Explanation: We need to test when the array doesn't contain the target element.
        int[] array = {1, 2, 3, 4, 5};
        int target = 6;
        int expected = -1;
        Assertions.assertEquals(expected, arrayExamples.findIndex(array, target));
    }
    no usages
    @Test    void testFindIndex_EmptyArray() {
        // Test Explanation: We need to test when the array is empty.
        int[] array = {};
        int target = 1;
        int expected = -1;
        Assertions.assertEquals(expected, arrayExamples.findIndex(array, target));
    }
}
```

大模型自动生成单元测试用例： UTCoder



训练数据准备

训练语料收集

Golang

百万级单测训练数据

FossMiner分布式集群 + 软件分析技术

GitHub开源代码仓

训练语料清洗

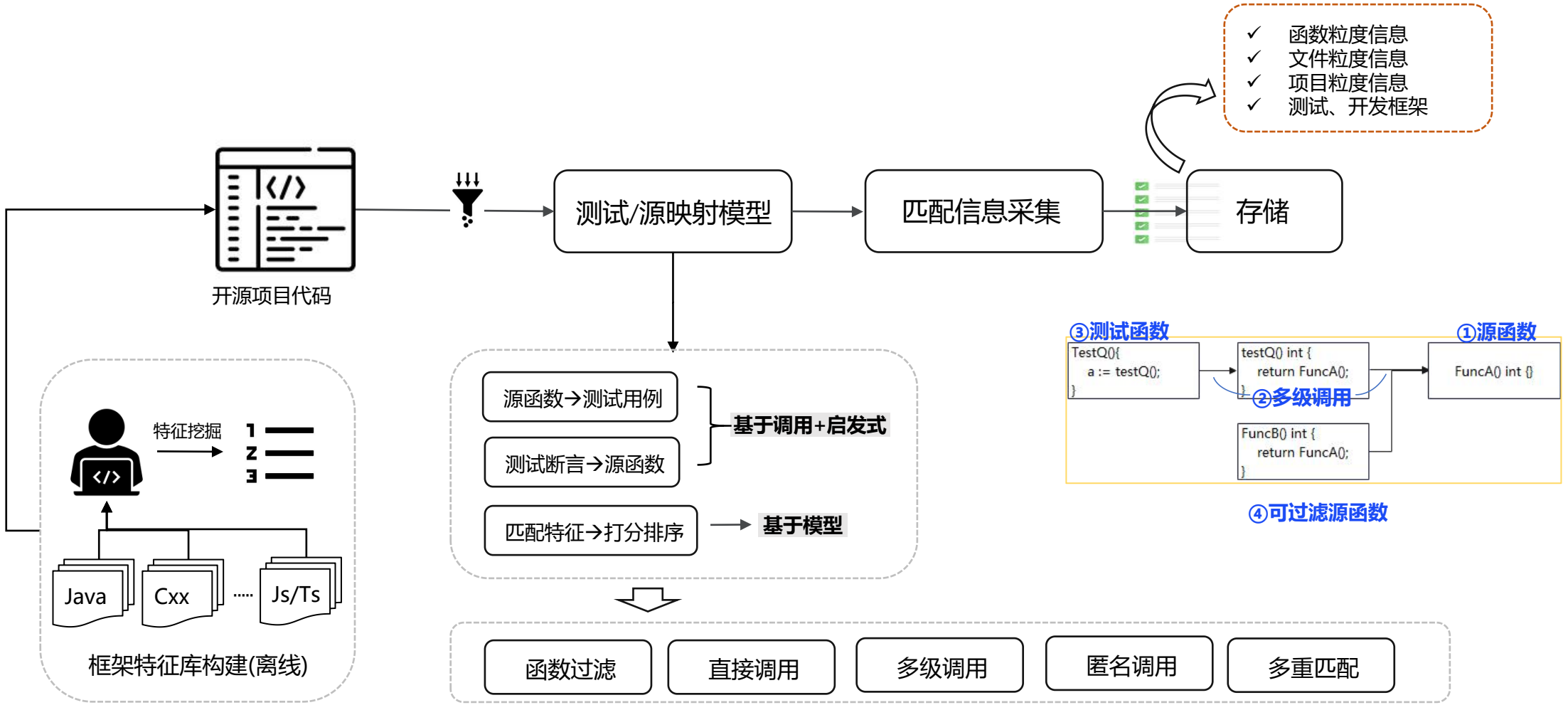
数据质量精化

编码格式、特殊字符占比、十六进制占比、字符长度等。

噪音数据筛选

不好的单测代码&断言风格
单测数量太多/太少
…

统一的多语言数据处理框架



提示词设计

- 被测方法

```
public String saveData(User user, String id, String content) throws IOException {
    if (!verify(user)) {
        return INVALID_USER;
    }
    Data data = redisDao.query( type: "data", id, Data.class);
    if (!verify(data)) {
        return INVALID_DATA;
    }
    data.setEncrypted(StaticUtil.encrypt(content));
    ResultEnum result = save(data);
    return callback(result);
}
```

通过依赖分析技术，自动提取被测函数上下文信息，用于构建prompt

- 上下文
  - 被测类、测试类

```
public class SampleService {
```

  - 参数类

```
@Data
public class User {
    no usages
    private String id;
    no usages
    private boolean isValid;
}
```

  - 局部变量类

```
@lombok.Data
public class Data {
    no usages
    private String id;
    no usages
    private boolean isExpired;
    no usages
    private String encrypted;
    no usages
    private long time;
}
```

  - 成员变量、调用的本类其他方法

- Prompt

```
Please help me create unit test cases. Please just write the test cases code. The tes
The source method under test is :
 public int getSingleton(LazyDoubleCheckSingleton instance) {
    if (instance == null) {
        return -1;
    } else {
        return instance.showCount();
    }
}
The return type of the method is: []
This method is located in the class: org.utbench.objects.singleton.SingletonFactory
The parameters in this method are: ['private LazyDoubleCheckSingleton();']
The fields in this class are: []
Given the test class:
public class SingletonFactoryTest {

    private SingletonFactory singletonFactory;

    @BeforeEach
    public void setUp() throws Exception {
        singletonFactory = new SingletonFactory();
    }

    @AfterEach
    public void tearDown() throws Exception {
    }



}
Please write some unit test methods for the method getSingleton
The test cases codes for the source method are:
```

  - 测试框架、mock框架

如何评测大模型生成单测效果

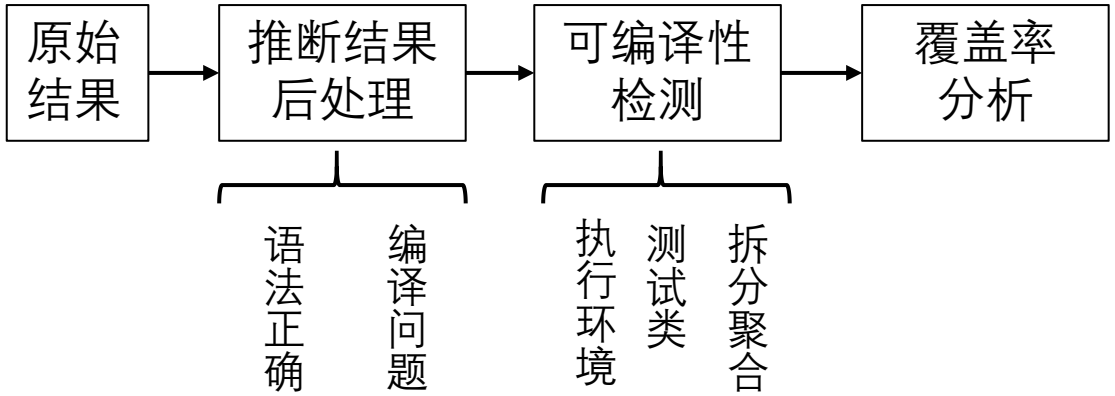
人工构建测评项目

JavaBasic：基本语法、数据结构、常见算法等

MicroService：开源微服务项目

Mall：开源商城项目

自动化测评工具



实际工程项目

业务部门专门构建了用于测评大模型生成结果的工程项目进行测评

测评关注指标

生成代码行

修改代码行、修改占比

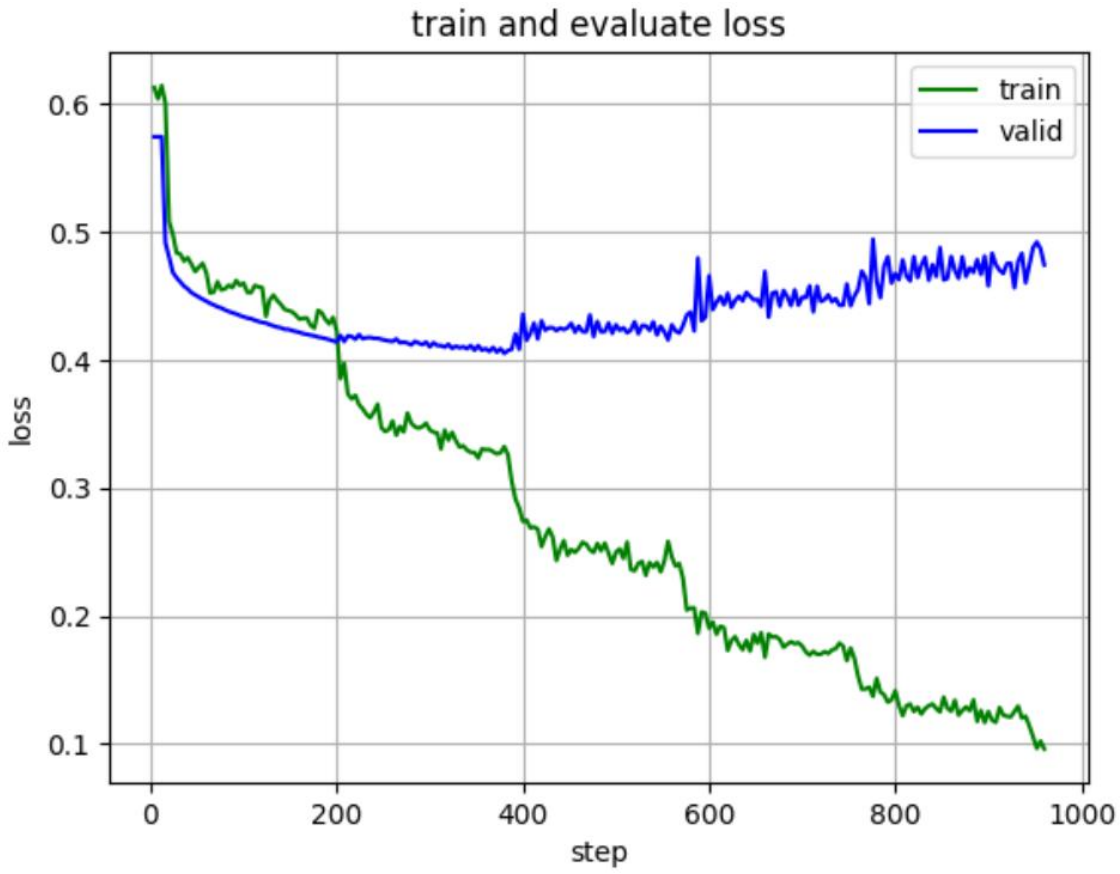
语法正确率

编译通过率

行覆盖

分支覆盖

模型精调




```
min train loss: 0.0961, step: 960
min eval loss: 0.405029296875, step: 380
```

经过筛选，使用10w条单测训练数据微调

训练时间：A100，约160h（4*40h）

结果分析 – 覆盖率&编译

测评项目	指标	StarCoder	WizardCoder	Phind	UTCoder	UTCoder+
JavaBasic	修改占比	10%	2%	3.21%	13.32%	13.32%
	行覆盖	51%	82%	81.80%	**98.04%**	**98.04%**
	分支覆盖	40%	75%	69.77%	**88.63%**	**88.63%**
MicroService	修改占比	-	-	12.22%	**4.49%**	**4.49%**
	行覆盖	-	-	55.60%	**73.60%**	**85.60%**
	分支覆盖	-	-	38.60%	**51.00%**	**60.00%**
Mall	修改占比	-	-	15.70%	**5.67%**	**5.67%**
	行覆盖	-	-	33.20%	**69.80%**	**93.20%**
	分支覆盖	-	-	24.00%	**58.00%**	**69.80%**
平均	修改占比	-	-	6.37%	**10.89%**	**10.89%**
	行覆盖	-	-	77.06%	**89.59%**	**95.50%**
	分支覆盖	-	-	65.09%	**78.59%**	**81.65%**

UTCoder生成单测代码示例

```java
public int findIndex(int[] sortedArray, int target) {
    if (sortedArray == null) {
        throw new RuntimeException("array is null");
    }
    for (int i = 0; i < sortedArray.length; i++) {
        if (sortedArray[i] == target) {
            return i;
        }
    }
    return -1;
}
```

行覆盖 ： 100%
Pass@1： 100%

```java
@Test
public void testFindIndex_empty() {...}

@Test
public void testFindIndex_singleElementMatchingTarget() {...}

@Test
public void testFindIndex_singleElementNotMatchingTarget() {...}

@Test
public void testFindIndex_multipleElementsNoMatch() {
    // Test on multiple elements but none matches the target
    int[] input = {1, 2, 3, 4, 5};
    assertEquals( expected: -1, new ArrayExamples().findIndex(input,  target: 6));
}

@Test
public void testFindIndex_multipleElementsOneMatchAtStart() {...}

@Test
public void testFindIndex_multipleElementsOneMatchInMiddle() {...}

@Test
public void testFindIndex_multipleElementsOneMatchAtEnd() {...}

@Test
public void testFindIndex_nullInput() {...}
```

如何提升大模型生成的单测编译率，解决幻觉问题?

```
// compile failed
新*
@Test
public void testSplitArrayWithCountLessThanListSize() {
    List<String> list = new ArrayList<~>();

    examples.addToList(list, "a");

    examples.addToList(list, "b");

    examples.addToList(list, "c");

    List<List<String>> res = examples.splitArray(list, 2);

    Assertions.assertEquals( expected: 2, res.size());

    Assertions.assertEquals( expected: 2, res.get(0).size());

    Assertions.assertEquals( expected: "a", res.get(0).get(0));

    Assertions.assertEquals( expected: "b", res.get(0).get(1));

    Assertions.assertEquals( expected: 1, res.get(1).size());

    Assertions.assertEquals( expected: "c", res.get(1).get(0));
}

// compile failed
新*
@Test
public void testSplitArrayWithNullList() {
    List<List<String>> res = examples.splitArray(null, 3);

    Assertions.assertNull(res);
}
```

```
public class ListExamplesTest {
    1个用法
    private ListExamples listExamples;

    @BeforeEach
    public void setUp() throws Exception {
        listExamples = new ListExamples();
    }

    @AfterEach
    public void tearDown() throws Exception {
    }

@Test
public void testSplitArray2() {
    List<Integer> list = new ArrayList<~>();

    for (int i = 0; i < 10; i++) {

        list.add(i);

    }

    List<List<Integer>> result = listExamples.splitArray(list, count: 1);

    Assertions.assertEquals( expected: 10, result.get(0).size());

}

// successfully compiled!
新*
@Test
public void testSplitArray3() {
    List<Integer> list = new ArrayList<~>();

    for (int i = 0; i < 10; i++) {

        list.add(i);

    }

    List<List<Integer>> result = listExamples.splitArray(list, count: 2);

    Assertions.assertEquals( expected: 5, result.get(0).size());

    Assertions.assertEquals( expected: 5, result.get(1).size());

}
```

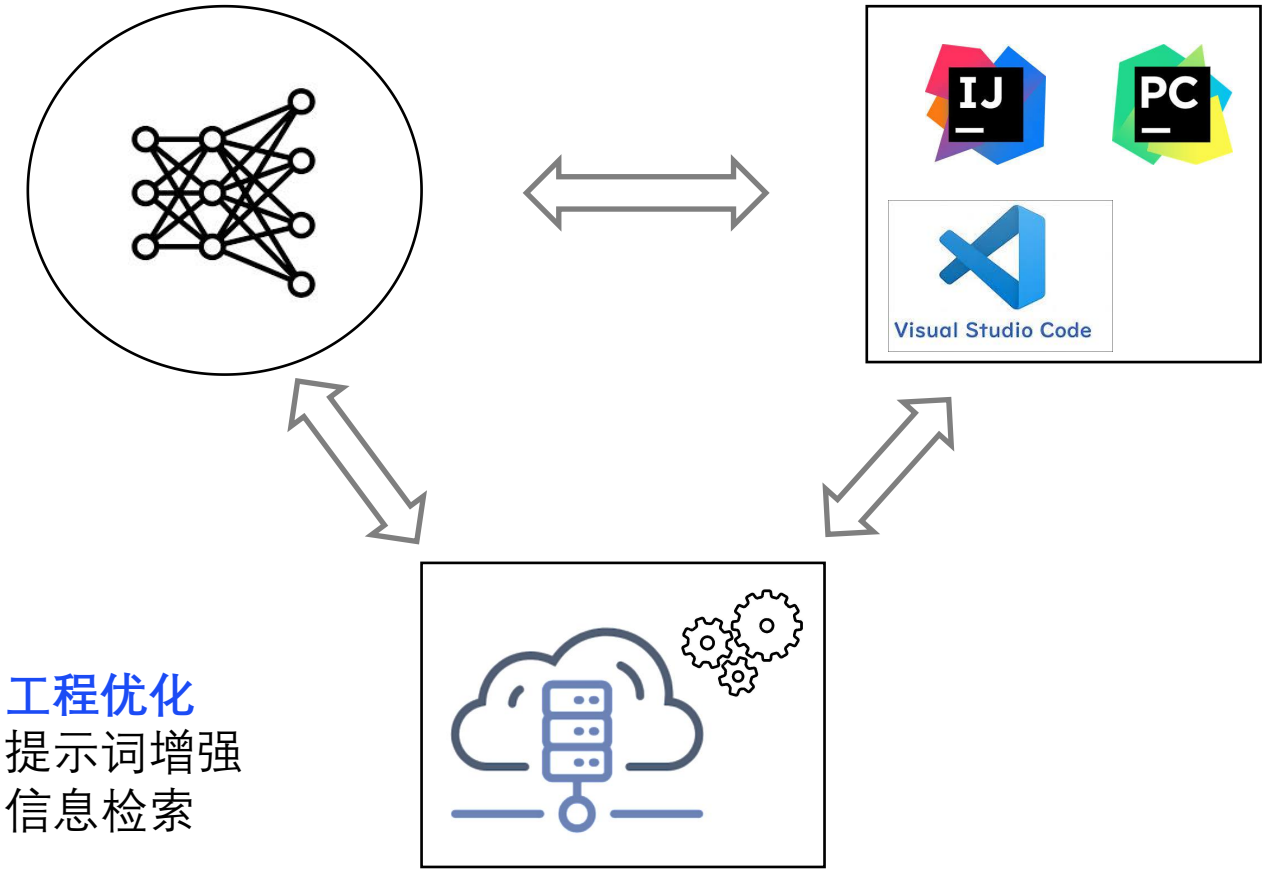
微调数据和提示词中加入更多相关信息

结合软件分析技术

PART 05
总结与展望

未来展望

模型能力
微调数据质量
多语言单测能力增强
测试用例演化



产品体验
用户体验优化
丰富交互模式
拓展使用场景

工程优化
提示词增强
信息检索

AiDD AI+ 软件研发数字峰会
AI+ software Development Digital summit

THANKS