文章编号：2095-2163(2022)08-0070-07 中图分类号：TP311 文献标志码：A

Android 应用程序的代码异味检测工具与方法综述

王露颖，边奕心，赵 松，朱 晓，涂 杰
（哈尔滨师范大学 计算机科学与信息工程学院，哈尔滨 150025）

摘 要：在 Android 应用程序中存在大量的代码异味，良好的代码异味检测工具和方法可以帮助程序开发者和维护者快速、高效、准确地在大量的程序中找到异味，是软件开发和维护强有力的技术支持方式。本文对 Android 应用程序中的代码异味检测工具和方法进行了总结，从应用环境、功能、支持语言、检测出的异味种类及检测精度等方面展开分析和比较；最后，针对目前工具存在的不足及未来的研究方向加以讨论和分析，为 Android 应用程序中代码异味检测及重构的研究提供参考。

关键词：Android 应用程序；代码异味；代码异味检测工具

Overview of code smells detection methods and tools of Android applications

WANG Luying, BIAN Yixin, ZHAO Song, ZHU Xiao, TU Jie
(School of Computer Science and Information Engineering, Harbin Normal University, Harbin 150025, China)

【Abstract】There are a lot of code smells in Android applications. Good code smells detection tools and methods can help program developers and maintainers quickly, efficiently and accurately find odors in a large number of programs. It is a powerful technological support way for software development and maintenance. This paper summarizes the code odor detection tools and methods in Android applications. Based on this, the corresponding analysis and comparison are given from the application environment, functions, supported languages, detected odor types and detection accuracy, and the deficiencies of the current tools and future research directions are futherly discussed and analyzed. The fruits could provide a reference for the research on code smells detection and refactoring in Android applications.

【Key words】Android applications; code smells; code smells detection tools

0 引 言

代码异味又被称为代码坏味道，最初是由 Fowler[1] 提出的。代码异味的存在增加了程序的维护成本，降低了代码质量。近年来，随着移动通信技术的迅猛发展，移动应用程序已经成为软件行业的发展主体。因此，许多学者从不同的角度对 Android 应用程序中的代码异味进行了广泛和深入的研究，开发了许多 Android 应用程序中代码异味自动检测工具。研究发现，用于检测传统桌面应用程序代码异味的检测工具也可检测 Android 应用程序中的代码异味[2]。但是，这些传统的工具只能检测 Android 应用程序中面向对象的代码异味。

不同于传统的桌面应用程序，Android 应用程序中除了存在面向对象的代码异味，还存在 Android 特有的代码异味。随着研究的不断深入，研究者不断提出针对 Android 应用程序中 Android 特有代码异味的检测方法和工具，以便于日后对 Android 应用程序中特有的代码异味展开进一步的研究。

目前，针对 Android 应用程序中的代码异味自动检测工具种类繁多，功能和适用环境各不相同。本文对比分析了目前具有代表性的 Android 应用程序中的代码异味检测工具和方法，以便 Android 应用程序的研究人员在面对实际代码异味时，能够选择合适的检测工具。

1 相关研究工作综述

本文重点关注 Android 应用程序的代码异味检

基金项目：国家自然科学基金(61902094)；黑龙江省自然科学基金(QC2018082)；黑龙江省普通本科高等学校青年创新人才培养计划(UNPYSCT-2018183)；哈尔滨师范大学博士科研启动基金项目(XKB201801)；哈尔滨师范大学计算机科学与信息工程学院科研项目(JKYKYY202004,JKYKYZ202104)。

作者简介：王露颖(1998-)，女，硕士研究生，主要研究方向：代码重构、程序分析；边奕心(1979-)，女，博士，讲师，主要研究方向：软件重构、程序分析；赵 松(1976-)，男，硕士，讲师，主要研究方向：代码重构、程序分析、计算机体系结构；朱 晓(1984-)，男，博士，讲师，主要研究方向：生物信息处理。

通讯作者：边奕心 Email：bianyu79@163.com

收稿日期：2022-02-23

测工具。为了调查检测工具的应用情况，通过以下 3 个步骤对文献进行检索：

(1)定义搜索字符串。本文定义了 3 组搜索字符串用于文献搜索，见表 1。主要在 IEEE Xplore、ACM、Springer 和 Google scholar 使用搜索字符串对文献进行了调查。

(2)选择主要研究对象。本文主要选取与 Android 应用程序中代码异味研究相关的文献。

(3)筛选目前开源可获取的自动检测工具。在这一步中，过滤掉已经不可获取、无法用于今后研究的 Android 应用程序代码异味自动检测工具。

表 1　用于文献调查的 3 组搜索字符串

Tab. 1　Three sets of search strings for literatures survey

编号	字符串
1	Android Code Smells、Android Code Bad Smells、Android Bad Smell、Android Anti-patterns、Android Android Design Flaws、Android-specific code smell、Design Smells
2	Detection、Recovery、Recognition、Identification
3	Approaches、Methods、Techniques、Tools

本文最终选取了 14 篇文献，见表 2。可以发现，针对 Android 应用程序中代码异味研究的论文，大多数都是在 2015~2020 年发表的，并且在仅研究 Android 应用程序中面向对象的代码异味时，研究人员大多仍使用传统的检测工具。

表 2　Android 应用程序中代码异味的相关文献

Tab. 2　Literatures on code smells in Android applications

时间	作者	工具	检测语言	数据集	代码异味
2015	Hecht[3]	Paprika	Java	15 个 Android 应用程序(Google Play)	4 种面向对象的代码异味 4 种特有的代码异味
2015	Hecht 等人[4]	Paprika	Java	15 个 Android 应用程序(Google Play)	4 种面向对象的代码异味 4 种特有的代码异味
2015	Hecht 等人[5]	Paprika	Java	106 个 Android 应用程序(Google Play)	3 种面向对象的代码异味 4 种特有的代码异味
2016	Hecht 等人[6]	Paprika	Java	2 个 Android 应用程序	3 种特有的代码异味
2017	Palomba 等人[7]	aDoctor	Java	18 个 Android 应用程序	15 种特有的代码异味
2017	Habchi 等人[8]	基于 Paprika	Objective-C 和 Swift	279 个 iOS 应用程序(Github)	4 种面向对象的代码异味 3 种 iOS 特有的代码异味
2017	Carette 等人[9]	Paprika+NAGA-VIPER	Java	5 个 Android 应用程序	3 种特有的代码异味
2017	Grano 等人[10]	Paprika	Java	395 个 Android 应用程序(F-Droid)	4 种面向对象的代码异味 4 种特有的代码异味
2018	Lim 等人[11]	JSpIRIT+DECOR+JDeodorant+TACO	Java	6 个 Android 应用程序(Google Play)	5 种面向对象的代码异味
2019	Mateus 等人[12]	Paprika	Java 和 Kotlin	925 个 Android 应用程序(F-Droid)	4 种面向对象的代码异味 6 种特有的代码异味
2020	Rahkema 等人[13]	Paprika 和 GraphifySwift(iOS)	Swift	273 个 iOS 应用程序 694 个 Android 应用程序	19 种面向对象的代码异味
2020	Gong 等人[14]	aDoctor	Java	3 680 个 Android 应用程序	10 种特有的代码异味
2020	Rasool 等人[15]	DAAP	Java	7 个 Android 应用程序	25 种特有的代码异味
2021	Hamdi 等人[16]	organic 和 aDoctor	Java	1 923 个 Android 应用程序 (GitHub 和 Google Play)	6 种面向对象的代码异味 15 种特有的代码异味

2　代码异味检测方法

2.1　代码异味检测流程

目前，代码异味检测方法的基本流程是相似的，如图 1 所示。

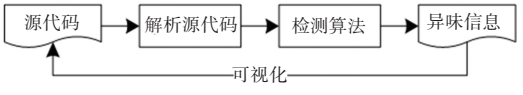


图 1　代码异味检测流程

Fig. 1　Code smells detection process

首先，对输入的项目源代码进行预处理，提取出

待测文件(如.java、.c 文件等);其次,根据不同的检测方法,将处理好的代码传换成中间形式,如抽象语法树列等结构;最后,使用不同的检测方法对其中的代码异味进行检测,输出检测结果。

2.2 代码异味检测方法分类

由于代码异味的存在增加了系统维护的难度,给软件系统带来了长期隐患。因此,对软件系统中异味进行检测是很有必要的。现有的代码异味检测工具所使用的检测方法主要可以分为 5 类。对此可给出研究分述如下。

(1)基于症状的检测方法。基于症状的异味检测方法是利用不同的症状和概念来描述代码异味,将异味的"症状"描述成中间形式后,转化为检测算法对其进行检测。

(2)基于度量的检测方法。基于度量的检测方法是最常用的代码异味探测方法。这种检测方法以逻辑表达式或条件表达式的形式,将一组度量和不同的阈值组合成一个检测规则,以检测代码中的不同异味。

(3)基于搜索的检测方法。基于搜索的检测方法源于基于搜索的软件工程(SBSE)相关研究。SBSE 使用基于搜索的方法来解决软件工程中的优化问题,其中绝大多数的技术都应用了机器学习算法。基于搜索的异味检测方法的提出,一定程度上弥补了之前的检测方法在精确度上的不足。

(4)基于概率的检测方法。基于概率的检测方法通过确定一个类中存在异味的概率,来识别代码中的异味类型。一些基于概率的检测方法,还应用了模糊逻辑规则和频繁模式树,对代码异味进行探测。

(5)基于可视化的检测方法。基于可视化的检测方法可以通过半自动化的过程来识别代码中的异味类型。该方法使用如面向可视化的策略和可视化设计缺陷检测策略等技术,将人类的专业知识与自动化的检测过程相结合。

现有的检测工具大多都是基于这些方法所提出的,本文对上述 5 类检测方法的优缺点做了比较,见表 3。

表 3 不同代码异味检测方法比较

Tab. 3 Comparison of different code smells detection methods

检测方法	优点	缺点
基于症状	算法实现简单,可扩展到对其他编程语言的源代码中的代码异味的检测	没有统一的概念定义代码异味的症状,人为制定检测规则工作量大,且检测精准率低
基于度量	检测准确率高,便于代码重构	局限于阈值的检测,如果阈值设置不准确,则漏检率会很高
基于搜索	弥补了之前方法在检测精度上的不足,检测准确率高	对于规模大的应用,搜索空间大,计算复杂度高
基于概率	通过模糊逻辑规则将代码异味重新排序,避免了在异味分类预测中的许多客观性问题	构建频繁模式树的代价较大,空间消耗大
基于可视化	可以降低处理大量数据的复杂性,实现半自动的代码异味检测过程	依赖于检测人员对代码异味的知识储备和检测经验,耗时且易出错

3 Android 应用程序的代码异味检测工具

从对 Android 应用程序中不同种代码异味检测的角度,可将现有 Android 应用程序中代码异味检测工具分为 3 类:一类是仅支持 Android 应用程序中面型对象的代码异味检测的工具,另一类是仅支持 Android 特有代码异味检测的工具,第三类是既支持面向对象、又支持 Android 特有的代码异味检测的工具。目前,可用于 Android 应用程序中代码异味检测的工具见表 4。

3.1 仅支持面向对象代码异味检测的工具

已有研究表明,传统的代码异味检测工具可以用来检测 Android 应用程序中面向对象的代码异味[2]。表 4 中加粗的 JSpIRIT、DECOR、TACO、JDeodorant 和 organic 表示已经用于 Android 应用程序中的代码异味检测研究。

同样用于 Android 应用程序中面向对象的代码异味检测的工具还有 inFusion。Mannan 等人[2]使用 inFusion 代码异味检测工具针对 20 种面向对象的代码异味,分析了其在 Android 应用程序和桌面应用程序中的分布差异。但 Rahkema 等人[13]研究面向对象代码异味在 Android 和 IOS 应用程序中的分布时,发现 inFusion 代码异味检测工具目前已经无法获取。经过调查,当下作为商业软件的 inFusion 确实已无法通过官方渠道免费获取。因此,后续无法使用 inFusion 对 Android 应用程序中面

向对象的代码异味进行研究，但 iPlasma 可以替代 inFusion 对部分面向对象的代码异味进行检测。iPlasma 是一个面向对象软件系统质量评估的集成环境，相当于 inFusion 的免费版。iPlasma 使用基于度量的方法对代码异味进行检测，但是 iPlasma 的功能远没有 inFusion 强大，也只能够检测到 11 种代码异味，而 inFusion 则可以检测到 22 种代码异味。

表 4　Android 应用程序中代码异味的检测工具

Tab. 4　Tools for detecting code smells in Android applications

检测工具类别	检测分类	检测工具	支持语言	可检测异味/种	能否重构
仅支持面向对象代码异味检测的工具	基于搜索	**JDeodorant**	Java	5	是
		SYMake	C\Java	4	是
	基于度量	Checkstyle	Java	4	否
		iPlasma	C++\Java	11	否
		PMD	Java 等 11 种	5	否
		TrueRefactor	Java	5	是
		organic	Java	11	否
		JSpIRIT	C++\Java	9	否
	基于症状	**DECOR**	Java	8	否
		TACO	Java	5	否
	基于可视化	Stench Blossom	Java	12	否
仅支持 Android 特有代码异味检测的工具	基于度量	aDoctor	Java	15	是
		DAAP	Java	25	否
既支持面向对象又支持 Android 特有代码异味检测的工具	基于度量	PAPRIKA	Java	17	否

综上所述，研究者在研究桌面应用程序中面向对象的代码异味时，开发的检测工具可以用于检测 Android 应用程序中面向对象的代码异味。但是由于这些检测工具大多以插件的形式存在，所以在检测 Andriod 应用程序中面向对象的代码异味时，存在局限性。目前，Androd 应用程序大多是在 Android Studio 环境下开发的。而在此之前，Androd 应用程序都是在 Eclipse IDE 和 ADT 环境下开发的。在 2015 年 6 月 ADT 不再提供更新支持后，开发者逐渐使用 Android Studio 开发 Android 应用程序。因此，目前开源的 Androd 项目存储库中混合了有 Android Studio 和旧的 Eclipse 开发的 Android 项目。显然，在 Android Studio 中运行 JDeodorant、TACO、JSpIRIT 等来检测代码异味是不可能的，这就导致在运行以 Eclipse 插件的形式存在的工具、对 Android 应用程序中的代码异味进行检测时，会存在一些麻烦。虽然 Google 公司为 Eclipse 开发的 Android 项目提供了自动迁移工具，使其可以由 Android Studio 继续开发，但并不支持反向迁移的情况，这就意味着研究者要是想用 JDeodorant、TACO、JSpIRIT 等检测由 Andriod Studio 开发的 Andriod 项目时，需要将 Andriod Studio 项目文件夹修改为 Eclipse 的正确结构，手动完成反向迁移，不利于大规模实验研究。

3.2　仅支持 Android 特有代码异味检测的工具

目前，在对 Android 特有代码异味的研究中，使用的开源可获取的自动检测工具主要有 2 个：aDoctor 和 DAAP。2 个检测工具的对比具体如下：

(1) 2 个检测工具在设计时，参考的都是 Reimann 等人提出的异味目录。

(2)2 个检测工具都利用抽象语法树当作中间形式来解析 Java 程序。不同的是，DAAP 针对存在于 XML 程序中的代码异味，使用文档对象模型(Document Object Model)来解析 XML 程序。

(3) aDoctor 可以对其中 5 种与能耗相关的代码异味进行重构，而 DAAP 不能进行重构操作。

综合前述分析可知，目前针对 Android 特有代码异味检测的开源工具还很少，且检测方法单一。

3.3　既支持面向对象又支持 Android 特有代码异味检测的工具

Android 应用程序中不仅存在传统面向对象的代码异味，还存在 Android 特有的代码异味。目前，2 类代码异味都可以检测的开源工具只有 PAPRIKA。PAPRIKA 采用基于度量的检测方法，对 Android 应用程序中的代码异味进行探测。PAPRIKA 通过以下步骤对 Android 应用程序中的代

码异味进行探测：

(1)解析 Android 应用程序中的 APK 文件，构建 PAPRIKA 模型。首先，使用 Soot 框架及其 Dexpler 模块来分析 APK 文件，从中提取度量信息，进而构造 PAPRIKA 模型。

(2)将构建好的 PAPRIKA 模型存储到一个图形数据库中。为了提供一种可扩展的方法来分析整个 Android 应用程序，PAPRIKA 采用的是 Neo4j 图形数据库来存储和查询构建好的 PAPRIKA 模型，提取出度量信息。

(3)查询图，以检测 Android 应用程序中的代码异味。当构建好的 PAPRIKA 模型被图形数据库加载和索引时，就可以使用数据库查询语言来探测代码异味。

在此基础上分析可知，作为唯一一个既可以检测 Android 应用程序中面向对象的代码异味和特有代码异味的工具，研究人员对 PAPRIKA 检测的异味种类不断进行扩展。其中，Habchi 等人[8]在研究 Android 和 IOS 应用程序中的代码异味时，对 PAPRIKA 的功能进行了扩展，使其可以检测由 C 和 Swift 语言开发的 IOS 应用程序中的代码异味。研究结果表明，Android 应用程序比 IOS 应用程序中存在的异味数量更多。Carette 等人[9]在分析 Android 特有代码气味对应用程序性能的影响时，对 PAPRIKA 进行扩展使其可以检测和重构 3 种 Android 特有的代码异味。去除代码异味后，可以提高 Android 应用程序的性能。Mateus 等人[12]在研究由 Kotlin 语言开发 Android 应用程序是否可以提高软件质量时，改进 PAPRIKA 使其可以检测由 Kotlin 语言开发的应用程序中的代码异味。最新研究指出，在由 Kotlin 语言开发的 Android 应用程序中，面向对象的代码异味更为常见。目前，PAPRIKA 最多可以检测 4 种面向对象的代码异味、13 种 Android 特有的代码异味。

4 异味检测工具的评价

不同的检测工具对于不同规模、开发语言的项目具有不同的检测效果。目前，常用于评估检测工具性能的指标主要有 3 个，分别是查准率(*Precision*)、查全率(*Recall*) 和 F_1 值(F_1 - *score*)，各指标数学定义分别如下：

$$Precision = \frac{TP}{TP + FP} \tag{1}$$

$$Recall = \frac{TP}{TP + FN} \tag{2}$$

$$F_1 - score = \frac{2 * Precision * Recall}{Precision + Recall} \tag{3}$$

其中，*TP*(True Positive)为被检测为正的正样本数；*FP*(False Positive)为被检测为正的负样本数；*FN*(False Negative)为被检测为负的正样本数。

在检测 Android 应用程序中面向对象的代码异味时，研究人员常使用传统检测桌面应用程序中代码异味的工具。对于此类检测工具的检测精度评价已有大量研究进行了总结。其中，Lim[11]重点评价了 JSpIRIT、JDeodorant、DECOR 和 TACO 四个传统的检测工具对 Android 应用程序中代码异味的检测效果。结果表明，在对 Andriod 应用程序中的代码异味进行检测时，4 个检测工具的漏检率极高。针对 Andriod 应用程序中的异味 Large Class，DECOR 作为其中检测效果最好的工具，结果中的 F_1 值仅为 36%，而 JDeodoran 则无法检测出该异味。

目前，缺少对 Android 特有代码异味检测工具的系统评价。因此，本文首先使用 DAAP、aDoctor 和 PAPRIKA 分别对异味忽略成员的方法(Member Ignoring Method，MIM)进行检测，然后利用查准率、查全率和 F_1 值对 3 个检测工具的检测效果进行评价，从而探讨 3 个工具的检测精确度。MIM 是指某个类中的方法，该方法既是非静态方法，也是非空方法，但该方法没有访问所在类的任何属性。本文选取了 6 个开源 Android 应用程序作为待测试程序，详细描述见表 5。

表 5 6 个 Android 应用程序及相关信息

Tab. 5 Six Android applications and related information

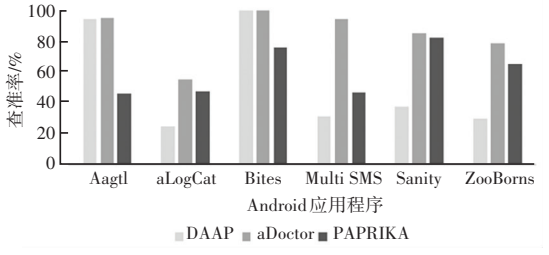
应用程序	描述	版本	类数	方法数	代码行数
Aagtl	寻宝助手	V1.0.31	133	1 649	19 710
aLogCat	终端应用	V2.6.1	16	97	1 041
Bites	食谱	V1.3	27	170	4 008
Multi SMS	短信管理	v2.3	9	88	1 263
Sanity	电话助理	V2.11	71	801	6 143
ZooBorns	图片查看器	v1.4.4	13	99	1 253

3 个检测工具对 6 个 Android 应用程序中异味 MIM 的平均检测结果见表 6，具体检测结果如图 2 所示。从表 6 和图 2 中可以看出，3 个工具对 MIM 的检测性能综合排名为：aDoctor>PAPRIKA>DAAP。由此可知，aDoctor 的检测效果最好，其 F_1 值较 DAAP 平均提高 16.18%、较 PAPRIKA 平均提高 5.11%。

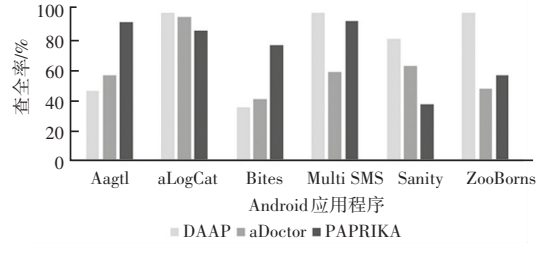
表 6　3 个检测工具对 MIM 的平均检测结果

Tab. 6　Average detection results for MIM by 3 detection tools　%

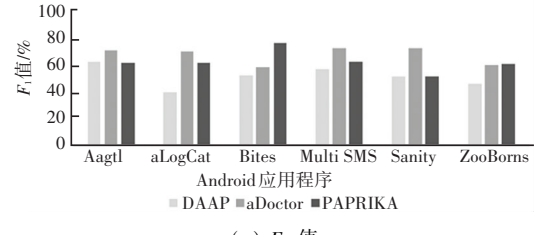
检测工具	查准率	查全率	F_1 值
DAAP	53.05	77.42	51.75
aDoctor	84.91	61.20	67.75
PAPRIKA	60.61	74.78	62.64



（a）查准率



（b）查全率



（c）F_1 值

图 2　不同工具在 6 个 Android 应用程序上的检测结果

Fig. 2　Detection results of different tools on 6 Android applications

综上所述，3 个工具对 MIM 的检测性能综合排名为：aDoctor>PAPRIKA>DAAP。aDoctor 作为检测效果最好的工具，F_1 值较 DAAP 平均提高16.18%、较 PAPRIKA 平均提高 5.11%。

5　结束语

本文对近几年用于 Android 应用程序中代码异味检测的工具和方法做了全面归纳与总结，并对 Android 特有代码异味检测工具的性能进行了简单的评估。综合目前的实际需求，Android 应用程序中代码异味的检测工具和方法的未来研究方向包括：

（1）由于 Android 应用程序与传统桌面应用程序在程序结构、API 调用、内存、CPU、网络、电池等方面的诸多差异，Android 应用程序中代码异味的种类及分布比传统桌面应用程序中的更复杂。因此，将传统的检测工具直接应用在 Android 应用程序中的代码异味检测上，其检测效果差且存在局限性。目前，针对 Android 应用程序中面向对象代码异味检测的工具仅有 PAPRIKA 一个，且最多只能检测其中的 4 种代码异味。为深入研究 Android 应用程序中的代码异味，未来可以针对 Android 应用程序的架构，提出适合且检测效果较好的代码异味检测工具。

（2）目前，针对 Android 特有代码异味检测的具有代表性的工具只有 DAAP、aDoctor 和 PAPRIKA，且这 3 个检测工具都是基于度量的检测方法所设计的。因此，在检测的过程中会受到阈值局限，出现漏检的情况，其检测结果也并不理想。因此，未来可以参考面向对象代码异味检测方法，基于其他检测方法研发出检测效果更好的检测工具。

（3）许多研究者在研究传统面向对象的代码异味时，针对多种异味之间的联系、如共存进行深入研究。而 Android 应用程序中不仅存在面向对象的代码异味，还存在其特有的代码异味。因此，针对 2 类异味之间存在的联系同样也亟待继续加大研发力度。目前，可同时检测 2 类代码异味的工具有 PAPRIKA，且其可检测的异味种类有限。因此，为了方便探索 Android 应用程序中 2 类代码异味之间的联系、进行大规模实验，未来可以继续对 PAPRIKA 进行扩展，使其可以检测更多种类的代码异味，还可以继续探索其他检测方法来检测 2 类代码异味，提出更高效、便捷的检测工具。

最后，本文研究可以给 Android 应用程序中的代码异味的研究者提供一些参考，选择合适的检测和重构工具，有助于后续对 Android 应用程序中的代码异味进行深入研究。

参考文献

[1] FOWLER M. Refactoring: improving the design of existing code [M]. USA: Addison-Wesley Professional, 2018.

[2] MANNAN U A, AHMED I, ALMURSHED R A M, et al. Understanding code smells in android applications [C]//IEEE/ACM International Conference on Mobile Software Engineering and Systems (MOBILESoft). Austin, TX, USA: IEEE, 2016: 225-236.

[3] HECHT G. An approach to detect and roid antipatterns [C]// IEEE/ACM International Conference on Software Engineering. Florence, Italy: IEEE, 2015: 766-768.

[4] HECHT G, ROUVOY R, MOHA N, et al. Detecting antipatterns in android apps [C]//2nd ACM International Conference on Mobile Software Engineering and Systems. Florence, Italy: IEEE, 2015: 148-149.

（下转第 84 页）