软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software, [doi: 10.13328/j.cnki.jos.000000]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel: +86-10-62562563

安卓移动应用兼容性测试综述*

陈　翔[1,2]

[1](西北工业大学 软件学院,陕西 西安　226019)

[2](计算机软件新技术国家重点实验室(南京大学),江苏 南京　210023)

通讯作者: 陈翔, E-mail: ****@ntu.edu.cn

摘　要: 随着安卓平台的应用的普及，越来越多的问题随之而来，本论文通过对国内外大量相关文献的分类研究来剖析和对比针对安卓移动应用兼容性测试的理论研究和相关发明。本论文的撰写过程中，我们收集筛选了大量国际会议的相关研究文献，并通过精读和复现研究成果进行相关文献的成果的落实，同时，对于同一类型的研究，进行了对比分析，讨论其相同和不同的原因。

关键词: 安卓移动应用，兼容性测试，故障定位。

中图法分类号: TP311

中文引用格式:

英文引用格式:

State-of-the-Art Survey of　Android Mobile App Compatibility Test

CHEN Xiang[1,2]

[1](School of Software, Northwestern Polytechnical University , Xi'an 226019, China)

[2](State Key Laboratory for Novel Software Technology (Nanjing University), Nanjing 210023, China)

Abstract:

Key words: Android mobile app, compatibility testing, fault location .

1　引言

计算机技术的高速发展，便利了人们的生活和工作，但是随着花样繁多的移动端软件，尤其是安卓台下开发的移动应用软件的普遍化，人们对软件的期望和要求越来越高，并不局限与软件的主要功能的要求。用户对软件的易用性，一致性，安全性都有很高的要求。某种意义上来说，这也是推动移动应用开发和测试的发展的要素。本论文研究的是最为普遍的安卓平台下开发和使用的移动应用的兼容性问题的检测定位以及修复。

本论文的撰写过程中，首先针对 7 个国际会议，包括三个 A 类会议和四个 B 类会议，根据关键词“安卓移动应用”“兼容性”“测试”等进行筛选和收集近三年的论文，初步获得大量论文近八十篇。随后我们那年对该八十篇论文进行粗读，进行第二次筛选，留下了近 35 篇与研究主题相关度较高的论文，第三次筛选，我们对论文进行分类分析，留下了具有代表性的 17 篇论文。至此，我们研究和分析的主要的论文集就整合完毕。通过对收集的相关论文的精读，复现和分析，我们展开了主题的研究和论文的撰写。

* 基金项目: 国家自然科学基金(00000000, 00000000); 南京大学计算机软件新技术国家重点实验室开放课题(KFKT00000000)

Foundation item: National Natural Science Foundation of China (00000000, 00000000); State Key Laboratory for Novel Software Technology (Nanjing University)开放课题 (KFKT00000000)

收稿时间: 0000-00-00; 修改时间: 0000-00-00; 采用时间: 0000-00-00; jos 在线出版时间: 0000-00-00

CNKI 在线出版时间: 0000-00-00

本论文共包括八部分，分别是引言，基本概念，检测方法，兼容性问题的定位，兼容性问题的修复，评估，总结和未来展望。这八个部分包括了对研究对象的阐述，对比和分析，以及最终的评估和结论，结构鲜明的介绍了本研究的成果和价值。

2 基本概念

3 检测方法

3.1 Cider（Understanding and detecting Callback Compatibility Issues for Android Applications）

Huang 等人[1]在 2018 年提出了一个图形化模型来捕捉由 API 进化引起的控制流不一致问题，并设计了一个静态分析技术 Cider 来检测回调兼容性问题。利用基于图的模型升成回调控制流图，捕捉 Android 应用程序中回调 API 的不一致控制流。利用 PI 图模型来检测 app CCFG 结构不一致引起的回调兼容性问题。对跨多个 API 级别的回调 API 调用协议中不一致性进行建模。PI 图中有两类节点（1）拒绝回调 API 的回调节点；（2）拒绝回调 API 的前一点和后一点作为辅助节点。基于 PI 图模型，通过带有不同 API 级别间隔的边遍历 PI 图，为 Android 应用程序生成简化的 CCFG 结构。帮助针对不同 API 级别的 Android 应用程序进行控制流分析，来检测回调兼容性问题。

3.2 IctApiFinder（Understanding and detecting evolution-induced compatibility issues in Android apps）

He D 等人[2]在 2018 年提出了一个名为 IctApiFinder 的新技术用来检测 Android 应用程序中不兼容的 API 用法。主要通过使用跨过程上下文敏感数据流分析框架精确计算每个 API 的可访问 Android 操作系统版本（调用 API 的操作系统版本），自动检测 Android 应用程序中不兼容的 API 使用。

定义：对于任何应用程序，如果且仅当满足以下三个条件，则使用 API 是不兼容的：

•有一个 SDK 版本的 API 级别大于或等于应用程序声明的 minSdkVersion 值。

•应用程序在该 SDK 版本上使用 API。

•该特定版本的 SDK 中不包括 API。

1，3 条件的检查较为简单。为了确定给定的 SDK 版本是否使用了 API，通过计算每个程序点的可访问 SDK 版本集，将变量 SDK_INT 直接与常量整数值进行比较，将比较结果作为检查点语句。对于 API 的每次使用，都会检查在使用点的可访问 SDK 版本中是否包含了 API，否则报告 bug。

3.3 ARPDroid（Automated Detection and repair of Incomopatible Uses of Runtime Permissions in Android Apps）

Dilhara 等人[3]在 2018 年提出了一种开源工具 ARPDroid，，通过静态控制流分析和字节码转换来检查并修复应用程序运行中出现的运行时权限导致的兼容性问题。为了检测和修复不兼容的权限使用，首先需要定位这些问题存在的代码层。在不访问或更改源代码的情况下检测和修复 Android 应用程序中不兼容的运行时权限使用。静态控制流分析确定需要检查权限使用的程序点。根据控制流分析的结果，通过专用模块确定给定的应用程序是否与运行时权限机制兼容。如果检测到应用程序不兼容（包含运行时权限模型相关的不兼容的权限使用），则将通过字节码转换修复检测模块发现的所有不兼容权限使用，然后检测转换后的应用程序是否兼容。

3.4 其他相关方法

Ki 等人[4]在 2019 年 ICSE 上提出 Mimic——一种能够正确处理和报告应用程序的 UI 兼容性问题的工具。虽然在 Mimic 提出之前，特定的对应用程序 UI 兼容性测试的工具并未出现。Ki 等人[3]提出的测试系统特定的针对应用程序 UI 兼容性问题。

Simone 等人[5]在 2019 年提出 ACRYL (Android Client-side Rule Learner)——一种机器学习的 API 兼容性问题的自动化检测方法，给方法从其他应用程序为解决 API 问题的优化中学习。这个方法它不仅允许检测兼容性问题，还可以对兼容性问题提出合理的修复建议。

Wei 等人[6]在 2016 年提出 FICFINDER——FICFINDER 根据捕获 Android API 及其相关上下文的模型执行静态代码分析，通过该模型可以触发兼容性问题。当 FICFINDER 检测到潜在问题时，它向开发人员报告相关的调试信息。

4 兼容性问题定位

4.1 静态控制流分析（ARPDroid）

在 Dilhara 等人[3]提出的开源工具 ARPDroid 中提到了不兼容权限定位方法。首先构造输入应用程序的调用图。基于调用图和应用程序每个方法的控制流图，使用输入 API 权限映射来定位不兼容的权限使用。通过调用图上的向后深度优先搜索找到 API 的权限负责调用方。

根据静态控制流的结果检测算法首先分析 app APK 中所包含的清单文件。如果应用程序的清单中声明的 a<23，则 Android 强制实施静态权限机制。否则算法进一步检查应用程序是否声明使用清单中的危险权限。如果上述步骤确定，则应用程序可能不兼容。算法继续针对不兼容的权限进行代码级检查。遍历静态控制流分析找到的所有权限负责调用方。如果每个权限负责调用方的（1）每个 API 调用站点都是权限检查的真分支执行的，（2）检查的假分支是通过 API 所需的所有权限的权限请求。则应用程序 兼容，否则不兼容。

4.2 静态识别旧 API（APPEvolve）

Fazzini 等人[7]在 2019 年提出了一种能基于更新示例自动执行 API 用法更新的技术——AppEvolve。通过检测旧的 API 用法，当在新版 API 的上运行时，检查他们是否可以执行来来更新这些用法。 通过静态方法来识别旧的 API。首先通过相关工作中定义的过程间数据流分析，计算可以执行的 API 版本集。然后执行过程内分析来识别旧的 API 用法，检查其中的方法调用是否可以在新版本的 API 上执行。将需要更新的 API 与 API 调用的位置存储在 API 使用报告中。

5 兼容性问题修复

5.1 静态控制流分析（ARPDroid）

开源工具 ARPDroid[3]中的修复算法针对不兼容权限负责调用方的列表和每个不兼容权限负责调用方中不兼容的 API 调用站点，插入对调用站点上调用的 API 所依赖的所有权限的检查。代码转换确保原始的 callsite 落在检查的真正分支中。如果检查失败，则插入请求所有必须权限的调用。修复完所有不兼容 API 之后，算法保证在包含权限负责调用方的类中包含权限请求响应处理程序。通过对转换后的应用程序运行检测算法，对其进行验证。

5.2 更新 API(APPEvolve）

APPEvolve 工具[7]搜索现有代码库获取新版本的 API 更新示例。然后分析、排序标识的更新示例，转换成可应用于目标应用程序的通用更新修补程序（更新补丁）。最后将通用更新补丁应用于目标应用程 序中的旧 API 用法，并铜鼓哦查一测试进行验证。

5.3 FILO(Fix-Locus)

Mobilio 等人[8]在 2019 年设计和提出了一种可以定位和修复因为底层 API 的更新而引起的兼容性问题的框架，提供可选择的易用有效的更改代码段——FILO。FILO 分三个主要阶段工作：（1）测试执行阶段运行复制失败的测试用例，并从两个可用环境中收集应用程序和框架之间的交互；（2）异常检测阶段通过比较收集到的痕迹来识别具有可疑交互作用的块；（3）Fix Locus Candidates 识别阶段确定并排序可能实施修复的地方，并将相应的证据关联起来。

6 评估

6.1 数据集评估分析

6.2 结果分析

7 主要研究小组和工作总结【可选】

8 未来展望

References:

[1] Huang, Huaxun & Wei, Lili & Liu, Yepang & Cheung, Shing-Chi. (2018). Understanding and detecting callback compatibility issues for Android applications. 532-542. 10.1145/3238147.3238181.

[2] He D , Li L , Wang L , et al. [ACM Press the 33rd ACM/IEEE International Conference - Montpellier, France (2018.09.03-2018.09.07)] Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering, - ASE 2018 - Understanding and detecting evolution-induced compatibility issues in Android apps[C]. 2018:167-177.

[3] Dilhara, Malinda & Cai, Haipeng & Jenkins, John. (2018). Automated detection and repair of incompatible uses of runtime permissions in Android apps. 67-71. 10.1145/3197231.3197255.

[4] Ki, Taeyeon & Park, Chang Min & Dantu, Karthik & Ko, Steven & Ziarek, Lukasz. (2019). Mimic: UI Compatibility Testing System for Android Apps. 246-256. 10.1109/ICSE.2019.00040.

[5] Scalabrino, Simone & Bavota, Gabriele & Linares-Vásquez, Mario & Lanza, Michele & Oliveto, Rocco. (2019). Data-Driven Solutions to Detect API Compatibility Issues in Android: An Empirical Study. 288-298. 10.1109/MSR.2019.00055.

[6] Wei, L. , Liu, Y. , & Cheung, S. C. . (2016). Taming Android fragmentation: characterizing and detecting compatibility issues for Android apps. IEEE/ACM International Conference. ACM.

[7] Fazzini, Mattia & Xin, Qi & Orso, Alessandro. (2019). Automated API-usage update for Android apps. 204-215. 10.1145/3293882.3330571.

[8]