计算机科学
COMPUTER SCIENCE

Android GUI自动化测试综述

杨艺, 王嬉, 赵春蕾, 步志亮

引用本文

杨艺, 王嬉, 赵春蕾, 步志亮.Android GUI自动化测试综述[J]. 计算机科学, 2022, 49(11A): 210900231-10.

YANG Yi, WANG Xi, ZHAO Chun-lei, BU Zhi-liang. Overview of Android GUI Automated Testing[J]. Computer Science, 2022, 49(11A): 210900231-10.

计算机科学 COMPUTER SCIENCE
http://www.jsjkx.com
DOI:10.11896/jsjkx.210900231

Android GUI 自动化测试综述

杨 艺 王 嬉 赵春蕾 步志亮
天津理工大学教育部视觉与系统省部共建重点实验室 天津 300384
天津市智能计算与软件新技术重点实验室 天津 300384
(1059355586@qq.com)

摘 要 新类型新版本的手机应用数量与日俱增,使得传统的人工测试方法已经无法负荷,因此需要研究人员提出更加有效的自动化测试方法。在自动化测试的过程中,Android 应用程序的 GUI(Graphical User Interface),即图形用户界面,发挥着极其重要的作用,GUI 自动化测试凭借其出色的测试覆盖率和故障检测性能,成为研究人员的重点研究对象。文中对当前 GUI 自动化测试的相关研究进行梳理和总结,选取其中有代表性、普遍性的自动化测试框架进行详细剖析,从测试策略、探索策略、错误报告、是否支持重放、测试环境、支持的事件类型、是否使用 APP 源码、是否开源、系统事件识别方法几个方面来对挑选出的自动化测试工具进行相应的分类、分析与对比。同时选取部分有代表性的自动化测试框架进行对比实验,以探究测试效率以及各自的优缺点。最后提出当前研究所面临的挑战以及未来的发展前景。

关键词:Android;自动化测试;GUI 测试;测试框架;测试用例生成
中图法分类号 TP391

Overview of Android GUI Automated Testing

YANG Yi,WANG Xi,ZHAO Chun-lei and BU Zhi-liang
Key Laboratory of Computer Vision and System of Ministry of Education,Tianjin University of Technology,Tianjin 300384,China
Tianjin Key Laboratory of Intelligent Computing and Novel Software Technology,Tianjin University of Technology,Tianjin 300384,China

Abstract With the increasing number of new types and versions of mobile apps,the traditional manual testing methods can't cater for the demand. Therefore,more effective automated testing methods need to be proposed. In the process of automated testing,the GUI (Graphical User Interface) of Android apps plays an extremely important role. GUI automated testing has become the focus of researchers because of its excellent test coverage and ability of crash detection. In this paper,the current research on GUI automated testing is sorted out and summarized,and the representative automated testing framework is chosen for detailed analysis. The selected automated testing tools are classified,analyzed and compared from the aspects of testing strategy,exploration strategy,crash report,whether to support replay,testing environment,supported event type,whether to use source code,whether open source,and system event identification method. At the same time,some representative automated testing frameworks are selected for contrast experiments to explore the testing efficiency and their advantages and disadvantages. Finally,the challenges faced by the current research and the future development prospects are proposed.

Keywords Android,Automated testing,GUI testing,Testing tools,Test case generation

1 引言

据中国互联网络信息中心统计,截至 2020 年 12 月,我国国内市场上监测到的 APP(Application,移动互联网应用)数量为 345 万[1]。在短周期开发且测试不充分的情况下,许多 APP 的质量并不过关,存在着运行卡顿,响应速度慢,甚至是黑屏、闪退的问题,给用户带来不好的使用体验,这说明在投放到市场之前,对 APP 进行完整、全面的测试是不可或缺的一环。

移动应用测试可分为人工测试和自动化测试。人工测试是最原始的测试方法。由于人工测试的测试速度慢,且测试量大时容易出错,无法满足市场上的测试需求,因此自动化测试应运而生。从整体上看,自动化测试是把人为驱动的测试行为转化为机器执行的过程。而 GUI 自动化测试是自动化测试中的一类,与传统的人工测试方法相比,其降低了测试成本和时间[2]。它通过自动获取、识别、分析 AUT(Application Under Test,被测应用程序)的 GUI,并根据框架所使用策略或算法的不同,生成不同的测试输入,以达到不同的测试效果,这也是近几年研究人员的研究热点。当前极大部分主流的开源 GUI 测试工具的运行都是将测试工具安装在 PC 端,

基金项目:国家基金面上-联合基金项目(U1536122);科技部"科技助力经济 2020"重点专项(SQ2020YFF0413781);天津市自然科学青年基金(18JCQNJC69900)
This work was supported by the National Funds-joint Fund Projects(U1536122),Key Special Project of "Science and Technology Helps Economy 2020" of the Ministry of Science and Technology(SQ2020YFF0413781) and Tianjin Natural Science Youth Fund (18JCQNJC69900).
通信作者:赵春蕾(zcltjut@gmail.com)

测试人员使用命令行、代码或者脚本等方式来操作测试工具，工具产生的事件序列将被发送到 Android 设备(真实设备或者模拟器)上并运行。

Android 应用测试的目的是测试运行在 Android 设备上的应用的功能、可用性和兼容性[3]。本文重点对 Android GUI 自动化测试的已有研究工作进行综述，主要选取了近年来有代表性、普遍性的或在某些方面有着显著效果的框架进行详细的介绍及对比分析，并对部分框架提出了完善建议。文中分别从测试策略、探索策略、错误报告、是否支持重放、测试环境、支持的事件类型、是否使用 APP 源码、是否开源、系统事件识别方法这 9 个不同的角度来对这些自动化测试工具进行描述，并根据不同的角度，进行了相应的分类、分析与对比，最后总结全文并展望未来的发展前景。

本文第 2 节介绍了自动化测试的发展历程；第 3 节讨论了传统 GUI 测试与移动 GUI 测试的相同点与不同点，并且对自动化测试框架进行分类，从 9 个角度对当前市场上普遍使用的框架和在某些方面(如图片存储方面、分析速度方面、代码覆盖率方面等)效果显著的框架进行详细介绍和优缺点分析，并将部分具有代表性的自动化测试框架进行对比分析；第 4 节将第 3 节中提到的部分有代表性的自动化测试框架进行对比实验，在测试环境相同的情况下，对比使用不同测试框架在相同时间对 AUT 进行测试得到的 Activity 覆盖率，并对实验结果进行分析；最后总结了当前 GUI 自动化测试面临的挑战，阐述了可能的解决方法，以及前景预测。

2 自动化测试发展历程

自动化测试的研究从实现技术角度可以分为 4 个阶段。第一个阶段是线性测试阶段，也就是简单的录制/回放阶段，使用硬件对键盘的输入进行录制并回放。但使用这样的方式录制的测试脚本很难维护，并且缺少检查点功能。后来的工作以此为基础进行改进，通过软件来代替硬件，并增加了检查点的功能，能够对 AUT 的一些功能进行验证，扩大了测试的范围。但此时的测试脚本是一种程序设计语言，编写测试脚本成为了测试人员面临的新的挑战，且测试脚本的维护问题依旧没有得到解决，在测试过程中一旦出现故障，测试人员需要对整个测试脚本进行重新录制。

第二个阶段是模块化和库阶段，将测试过程分成不同的区域，将各个区域的测试与检查操作封装成函数，形成库文件被测试用例调用。虽然各个功能独立维护，但是将操作和数据放在一起会使得一旦需要对大量不同的数据进行测试，测试人员就需要编写大量的冗余测试用例。

第三个阶段是测试框架阶段，自动化测试框架最早能够追溯到 Mercury Interactive 公司于 1996 年完成的 winrunner[4]框架，它是一款功能性测试框架，实现了移动应用的录制/回放，能够检测 AUT 能否正常运行以及能否达到预期功能。2007 年 Android 正式发布源代码时，同时发布了 Monkey[5]，MonkeyRunner 和 Instrumentation 这 3 个测试框架。其中 Monkey 向 AUT 发送伪随机的用户事件，不考虑它的 GUI 结构，比起功能测试，更适用于进行压力测试。在之后的工作中，研究人员对自动化测试中的随机策略进行改进，Machiry 等提出的 Dynodroid[6] 使用了随机策略，但是相比 Monkey 的随机输入更加智能。近年来开发人员一直对随机策略进行改进，但 AUT 的测试覆盖率并没有得到大幅度的提高。开发人员开始更换思路，提出了基于 GUI 模型的框架。这种新的策略能够截取 AUT 当前 GUI，并对所截取的图片进行分析，以此产生下一步操作的命令，向 AUT 输入相应的事件，以达到更高的测试覆盖率。基于 GUI 模型框架的典型代表有 Li 等提出的 DroidBot[7]，它基于即时生成的状态转换模型，使用不同算法生成用户界面(User Interface，UI)指导的测试输入，并且允许用户集成自己的策略或算法。随着机器学习技术的迅猛发展，研究人员在研究了基于随机的方法和基于模型的方法之后，也将机器学习融入到自动化测试框架的开发之中。Ardito 等[8]开发的的测试框架使用了机器学习算法，能够有效地为 Android 应用程序创建高级的、自适应的测试用例，并允许开发人员编写测试脚本，无需知道 AUT 具体的实现细节和用户界面。这样的处理使该测试框架具有广泛性，框架能够在大量应用程序或者同一应用程序的不同版本上执行自适应测试，不再需要手动编辑测试。

第四个阶段是混合框架阶段，以两个不同的测试框架为基础，进行二次开发构建新的自动化测试框架，以达到提升运行效率、测试覆盖率和检测效率的目的。

3 自动化测试框架分类及分析

3.1 传统 GUI 测试与移动 GUI 测试的对比

传统 GUI 测试是在 PC 端进行的应用程序测试，不论是传统 GUI 测试还是移动 GUI 测试，其使用的基础知识大都相同：同样的检测方法、同样的测试用例设计方法、均对用户界面进行检查和分析、均进行页面性能测试和应用压力测试以及稳定性测试等。

移动 GUI 测试相比传统 GUI 测试除了要考虑基本的功能测试、性能测试以外，还要考虑移动设备自身的属性特征，也存在着一些区别。

(1)页面布局不同：计算机的屏幕较大，可容纳的控件和可显示的信息更多，页面布局也更加随意灵活；而移动应用屏幕相对较小，显示的控件和信息有限，一般通过单列显示，并且使用下拉框等节省空间的控件，使页面布局更加合理。

(2)使用场景不同：传统软件的使用地点比较固定，网络信号也比较稳定；而移动端的使用场景相比 PC 端要更加复杂，且户外运动、公交地铁等新类型软件的出现，使得移动端的使用场景呈现多样化趋势。

(3)输入方式不同：传统软件一般使用鼠标和键盘进行输入操作；而移动应用除了上述两种方式之外，还添加了更加智能、多样的输入方式，如语音方式、触屏方式、电容触控笔方式等。

(4)输入事件不同：传统软件的输入事件大致分为 3 类：鼠标类、键盘类、其他类(load，focus，blur 等)，而移动应用的输入事件由于其输入方式的不同，衍生出了多种事件类型，如触摸事件、手势事件、二指缩放事件、轨迹事件、屏幕旋转事件、其他类型事件。

(5)操作系统不同：PC 端的操作系统为 Windows，Mac，Linux 等，移动端的操作系统为 Android，IOS 等。

(6)移动端的专项测试：相较于传统 GUI 测试，移动端有着因其自身属性而设置的专项测试，如旋转测试、流量测试、交叉事件测试、兼容性测试、耗电量测试、弱网络测试和用户权限测试。

3.2 Android 自动化测试框架的分类

Android 自动化测试框架是对 Android APP 进行自动化

测试时使用的工具。按框架的定义，自动化测试框架可以分为基础功能测试框架和非功能框架。基础功能测试框架主要是用于测试软件的基础功能，例如，打开一个应用软件，模拟键盘输入、鼠标点击和屏幕滑动等；而非功能测试框架则不具备测试功能，主要是用于自动化测试用例的管理和执行。按不同的测试类型，可以将其分为功能自动化测试框架、性能自动化测试框架、安全自动化测试框架、内容自动化测试框架、兼容性自动化测试框架、安装/卸载自动化测试框架。

3.3 Android 自动化测试框架的特征

本节将从测试策略、探索策略、错误报告、是否支持重放、测试环境、支持的事件类型、是否使用 APP 源码、是否开源、系统事件识别方法 9 个不同的角度来描述目前市面上较为常见的自动化测试工具。

3.3.1 测试策略

当前，Android 平台的软件自动化测试主要分为 3 类：白盒测试、黑盒测试、灰盒测试。我们调查了市面上常用的 40 个 Android 自动化测试框架，并将 3 种测试策略使用的占比情况进行统计，结果如图 1 所示。

(1)白盒测试：白盒测试工具在针对 APP 进行测试时需要获取其源码，对代码进行分析后产生控制流图，在此基础上使用某些算法分析，从而产生测试用例。这种测试策略的优点是得到的测试用例更加精确，但对 AUT 的要求较高，因此市面上使用此种策略的测试框架较少。典型的使用白盒测试策略的框架包括 Intent Fuzzer[9]，ACTEve[10]，EvoDroid[11]等。

(2)黑盒测试：黑盒测试策略无需获取 APP 源码，只需在测试过程中监听移动设备界面的 GUI 信息，通过手动或自动化测试框架来完成动作注入，即可实现持续交互。这种测试策略的优点是对 AUT 的要求较低，对市面上大部分的 APP 都能进行有效测试，但与白盒测试相比，其精确度大大降低。典型的使用黑盒测试策略的框架包括 Monkey，Dynodroid，PUMA[12]，AgileUATM[13]等。

(3)灰盒测试：灰盒测试介于白盒测试和黑盒测试之间。与黑盒测试相比更关注 APP 的内部逻辑，多用于集成测试阶段，用来判断 APP 内部的运行状态。典型的使用灰盒测试策略的框架包括 Stoat[14]，ORBIT[15]，Sapienz[16]等。

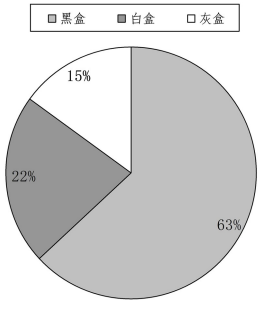

图 1　3 种测试策略占比情况

Fig. 1　Proportion of three test strategies

3.3.2 探索策略

Android 应用程序在运行时会产生大量的状态，同时，GUI 的各个状态可以由多种不同事件来触发，具有高度的互动性，因此测试用例的探索通常需要大量的时间和精力[17]。本文将现存的自动化测试框架所使用的探索策略分为 4 种：随机探索策略、基于模型的探索策略、系统探索策略、其他策略。开发人员通过这些策略生成大量事件序列，但一些以效率为导向的测试框架会生成大量低级的、对测试结果没有贡献的无效事件，导致测试过程产生大量冗余，严重影响了测试性能。因此在测试过程中，需要更优的测试策略来生成质量更高、更有效的测试输入[18]。

近年来自动化测试的发展越来越迅速，Android 测试输入工具渐渐向支持用户交互，并允许系统交互作为测试输入的方向发展[19]。在测试过程中，为了尽可能多地探索到 APP 的状态，越来越多的开发人员开始使用更加新颖的探索策略，并尝试将两种或多种算法相结合，来对 APP 进行测试。具体策略使用情况如下。

(1)随机探索策略。随机策略是目前应用最广泛的测试策略之一，被广泛应用于嵌入式软件系统、SQL 数据库系统、Android 应用程序等测试场景。自动化测试框架通过某种算法或策略来随机指定一系列 GUI 事件并发送给移动端执行，与其他策略相比更为轻量级。但测试过程中产生的冗余输入较多，因此更适用于压力测试的场景。其中具有代表性的框架是 Monkey，Dynodroid 和 PUMA。

Monkey 是 Google 提供的一个命令行工具，可运行在模拟器或实际设备中，它向系统发送伪随机的用户事件，模拟用户的按键输入、触摸屏输入、手势输入等，对正在运行的应用程序进行测试[20]。Monkey 的优点是能够在短时间内产生数量可观的测试输入，缺点是 Monkey 测试的对象仅为应用程序包，生成的测试输入是随机的，只能由开发人员指定输入事件的比例，不能进行自定义输入。且 Monkey 测试无法进行录制回放，具有较大的局限性。针对 Monkey 随机产生的事件序列可能会重复操作相同的控件或跳转到相同的窗口的问题，Yan 等[21]通过对 Monkey 进行改进，增强 Monkey 对序列记录和重放的支持，提出了一种针对 Android 应用程序的序列减少方法，有助于理解崩溃行为和故障定位。

Dynodroid 是 Machiry 等在 2013 年提出的一个公开可用的测试工具，它同样使用了随机策略，但是相比 Monkey 的随机输入更加智能。根据应用程序中注册的事件监听器和 GUI 结构，DynoDroid 能够过滤掉大量不相关的系统事件。并且在测试的过程中加入了权重的概念，通过事件出现的频率以及上下文来调整事件权重，在决策上偏向于选择最近最少使用的事件。与此同时 Dynodroid 将自动化输入和手动输入相结合，解决了应用程序中需要输入文本信息或者勾选协议等自动化测试框架暂时无法自动完成的操作。与 Monkey 相比，Dynodroid 提供的事件更智能，产生的输入序列更加简洁。

Sasnauskas 等开发了一个意图模糊化框架 IntentFuzzer，提出了一种随机模糊和静态分析的高效运行组合。这种测试组合获得了稳健性和运行效率的良好折中，且能够有效地在众多 Android 应用程序中发现明显的崩溃错误。缺点在于其将随机模糊方法与 Android 组件间的通信工作关联起来时，忽略了 Android 处理其他攻击向量及其对策的工作，比如权限分析、恶意软件检测、内核模糊等。未来的研究方向是在可靠性和精度损失可接受的情况下，寻找更高效地发现应用程序中错误的方法。

PUMA 使用了 Monkey 来进行 AUT 状态的探索，这个动态分析框架为大量不同的动态分析任务进行实例化，支持对各种各样的应用属性进行分析，允许用户灵活指定要探索哪些应用状态以及如何探索，提供对应用运行时状态的编程

访问以进行分析，并支持动态运行时的环境修改。而 Huang 等[22]提出的自适应随机测试(ART)能够将测试用例均匀地分布输入到设备中，从而增强在 ART 的故障检测能力。

(2)基于模型的探索策略。基于模型的策略将 APP 的 GUI 和对 APP 的操作关联起来，从而生成以有向图为代表的模型。与随机策略相比，基于模型的探索技术产生的冗余输入更少，覆盖率更高。但由于 APP 复杂多变，导致生成的模型往往不够准确，并常常伴随着状态爆炸的问题。对已有工作的分析如下。

将模型与 APP 测试相结合的想法最初由 Weyuker 提出，但基于模型的探索策略近几年才流行起来[23]。Li 等提出的 DroidBot 使用基于模型的探索策略，将设备状态和触发状态转换的事件抽象成一个状态为节点，事件为边的有向图，将探索 APP 状态的问题转换成有向图的遍历问题。DroidBot 的优点是对测试设备和测试 APP 没有严格的要求，并且在现今大部分框架使用开源 APP 进行测试的情况下，DroidBot 提供了一种新方法来评估测试输入的有效性。DroidBot 能够为每个测试输入生成调用堆栈跟踪，其中包含测试输入触发的应用程序方法和系统方法。我们可以使用调用堆栈作为近似指标来量化测试输入的有效性。对于手机自带的简单 APP 上的测试，如便签、计算器、录音机、指南针、日历等，界面简单且总数量有限，操作结果单一，同一动作导向不同结果的情况很少出现。在测试上述应用时，传统的 DFS 以及 BFS 算法仍有不错的遍历效率。然而，在相对复杂的应用程序中，如淘宝、支付宝、微博等，其界面控件较多，排版复杂，同一事件可能产生不同的 UI 状态，这时使用传统的遍历方法就存在很大的局限性，可能会出现同样的下滑刷新操作得到的永远是不同的界面，简单的后退操作可能直接退出应用程序，无法保证回到上一步页面的问题，以及局部循环问题。

针对使用传统算法产生的弊端，字节跳动技术团队所开发的 Fastbot[24]提出了多种启发式遍历算法，如单步或 N 步 UCB 算法、MTree 算法以及 NStepQ 算法，来规避上述问题的产生。针对基于模型的 GUI 测试会受到手机端的内存大小和计算能力的限制这一问题，Fastbot 将消耗大量内存与计算资源的部分部署到云端，在客户端只保留 UI 信息监听和动作注入的功能。这一操作对大部分基于模型的框架都大有裨益，例如 Droidbot，分析与计算下一步的测试输入占据了很大一部分的资源，导致测试过程中被测应用的运行以及实时测试结果的产生受到了限制。

Choi 等提出的 SwiftHand[25]是一种自动化技术，主要用于为 APP 生成测试输入序列。该技术使用机器学习在测试期间学习 APP 的模型，再使用学习的模型生成访问 APP 未探索状态的用户输入，同时能够使用 APP 在生成的输入上的执行来优化模型。之前的自动探索算法偶尔需要重新启动应用程序，以便探索从初始状态可以到达的其他状态，以快速实现代码覆盖率，然而研究发现，重启应用的时间比探索任何其他状态转换所需的时间要长得多，基于此，SwiftHand 使用一种能够最小化重新启动次数的探索和学习算法，来避免重新启动应用程序，将大量的时间都用于探索新状态。

Su 等提出了 Stoat，这是一种在 APP 上执行随机模型测试的新型引导方法，用于改进 GUI 测试。Stoat 利用 APP 的行为模型迭代地优化测试生成过程，以得到高覆盖率和更加多样化的事件序列。AndroidRipper[26]、MobiGUITAR[27](前者的扩展)、ORBIT 和 AMOLA[28]使用状态机来表示 APP 模型，但它们仅使用基础图探索算法(如深度优先搜索算法、广度优先搜索算法)来实现简单的 UI 探索，因此其性能受到了一定限制。MonkeyLab[29]能够记录用户的执行跟踪，以挖掘统计语言模型，但其目的是生成可重放的事件序列。

Salihu 等提出了 AMOGA[30]策略，这是一种使用混合、动静结合方法生成 APP 的 GUI 模型以进行基于模型的测试策略。它实现了一种新颖的爬行技术，该技术能够生成与每个事件相关联的 UI 元素的事件列表，在运行时动态地执行事件序列以探索应用程序的行为。这种策略可以解决由于逆向工程的不完整性所导致的模型不能广泛捕获应用程序动态行为的问题。

近两年 APP 的功能趋于复杂化，需要用户同时进行交互协作的 APP 越来越多，Ravelo-Méndez 等提出了第一个公开可用的跨设备测试工具 Kraken[31]，其能够支持 APP 进行两个或多个设备的交互测试。

(3)系统探索策略。系统探索策略通常使用符号执行等更为复杂的技术进行测试，经常与基于模型的探索策略结合应用，来达到更高的测试效率。这种技术可以通过特定输入来深入探索 APP 的某些特定操作。优点在于可以利用源代码来发现前两种策略难以探索到的 APP 行为。其中具有代表性的框架包括：ACTEve，AndroidRipper，A3E Depth-First[32]等。

Anand 等在 2012 年提出了第一个使用系统探索策略的测试框架 ACTEve。它通过单独检查每个程序执行的简单数据和控制流事实来计算事件序列之间的包含，尽可能多地覆盖分支，从而缓解路径爆炸问题。2013 年，Tanzirul 等提出了 A3E Depth-First 测试方法，使用静态污点分析来对 APP 的活动进行快速有效的探索。同年，Amalfitano 等提出的 AndroidRipper 使用了基于状态的方法来动态分析 GUI 事件，它自动探索和分析被测应用程序的 GUI，目的是得到当前图形用户界面中的可触发事件，这些事件能够组成具有不同故障检测能力的测试用例，输入事件触发状态的变化。

符号执行是一种程序分析技术，使用符号值作为程序输入，程序的输出被转换为符号输入的函数。目前在软件测试领域有很好的应用前景。2013 年，Jensen 等提出了一种基于符号执行策略的目标算法，用于自动查找到达应用程序代码中给定目标行的事件序列，适用于需要长事件序列和事件参数推理的目标[33]。Mirzaei 等于 2016 年提出的 SIG-Droid[34]框架提供了支持 Android 应用程序的符号执行引擎，将基于模型的策略和符号执行结合起来，系统地生成测试输入。但其使用的深度优先搜索策略并不能保证生成所有可能的事件序列。

(4)其他策略。包括基于搜索、进化算法等策略。当前机器学习应用广泛，具有很好的发展前景。其中心思想是研究计算机如何模拟或实现人类的学习行为，以获取新的知识或技能，并将已经获得的知识作为基础知识，重新组织其结构加强自身的性能。很多机器学习方法如符号学习、进化算法、深度学习等都可以很好地应用到 Android 的自动化测试领域中。

Hu 等于 2014 年提出了第一个使用基于搜索探索策略的测试框架 AppDoctor[35]，并将搜索算法与启发式算法相结合，包括广度优先搜索、深度优先搜索和开发人员编写启发式方法，来选择要执行的测试输入。使用近似执行方法来提高

测试效率,但其重放功能具有不确定性,可能导致在测试过程中忽略了应用程序在执行时发生的错误。随后 Mahmood 等提出的 EvoDroid 测试框架,将其与进化方法相结合来进行测试,并克服了使用进化测试算法普遍存在的问题,即无法在搜索中传递优秀个体的基因组成。它通过提取接口模型和调用图模型来指导计算搜索过程。但在最坏的场景下,EvoDroid 无法系统地推理输入条件,其性能会大大降低。2016 年,Mao 等提出的基于搜索的多目标自动探索测试方法 Sapienz,是将最小化测试序列长度和最大化其他目标结合到基于帕累托最优多目标搜索的测试方法,利用基因编程来进化生成的测试输入。它使用基于多目标搜索的测试来自动探索和优化测试序列,支持多级检测,使用的进化算法可不断优化测试覆盖率、测试序列长度及探索到的崩溃次数。Li 等于 2019 年提出了一种基于深度学习的自动化 Android 应用测试方法 Humanoid[36]。它能够学习人类与 APP 交互的过程,然后让学习到的模型扮演测试人员的角色来指导测试生成。Humanoid可以在人类和模型交互的过程中判断事件的重要性,并对其进行优先级排序,从而更高效率地生成能够得到重要状态的测试输入。但该测试框架目前的局限性在于不支持系统广播、传感器等系统级事件;同时,也存在其他类似于 Humanoid的以有价值的关于应用程序行为的人类知识为指导的测试方法,如 Adamant[37],它能够将用户在使用 APP 时的思想纳入测试输入生成过程中,来生成更加有效的测试输入,但与 Humanoid 相比,Adamant 的测试过程更为复杂繁琐,需要测试人员编写相应测试脚本来完成测试。Amalfitano 等[38]通过机器学习将 Android 应用程序的自动化 GUI 探索与捕获和回放相结合,提出了 juGULAR 技术,这种混合 GUI 探索技术可以在覆盖活动、覆盖代码行和生成网络流量字节方面以合理的人工干预成本提高探测能力。但其仅对特定的情况即登录和网络设置有效果,未来可以考虑对其他非确定的情况进行处理。Packeviius 等[39]提出了一种自动化视觉测试方法,该方法基于启发式和预期状态预测算法自动搜索缺陷,用来测试 APP 在其他兼容或者类似设备上是否能够正确显示的场景。2020 年 Wang 等提出的 Combodroid[40]框架使用基于搜索的策略,提供自动或手动两种方法获取测试用例,生成长且有意义的测试输入,从而深入探索 APP 的复杂功能。

3.3.3　错误报告

当 AUT 发生异常(如闪退、黑屏、程序未响应等)时,自动化测试框架检测异常,并将发生的异常记录到错误报告中。通常包括空指针异常、进程异常、debug 异常、低内存异常、操作无响应异常和其他异常。一般地,生成的错误报告以日志、图像或文本的形式呈现,也有部分框架将错误报告信息直接显示在命令行页面中。错误报告有利于测试人员更直观地了解 AUT 存在的问题,有利于开发人员更快发现问题和解决问题,提高优化阶段的效率。对已有工作的分析如下。

使用 Monkey 进行测试,在 AUT 出现异常的情况下,错误信息会直接显示在命令行页面。或者使用 logcat 命令,将错误日志导出,可以在日志中搜索报错的关键字,加快定位错误位置的速度。DroidBot 的测试结果通过 HTML 页面呈现,页面中实时更新、显示了测试过程中的状态转换模型,其从本质上来说是一个有向图,模型中的每个节点代表当前设备运行的状态,两个节点之间的边代表触发状态转换的事件,当被测应用产生异常时,通过观察实时产生的状态转换模型中的节点就可以得到结果。Liu 提出的用于测试多媒体应用程序的 CTP[41]云测试平台也生成了 GUI 状态图,可用于分析应用程序的行为,有助于崩溃诊断和调试。其能够在测试过程中收集测试视频、屏幕截图以及每个选定设备的性能数据,并将其集成到最终的测试报告中,使用户可以通过不同的声音和视频配置获取硬件资源使用情况,如 CPU、内存、电池和网络,来评估应用程序在不同设备上的性能,确保 Android 多媒体应用程序的兼容性,同时节省测试时间和精力。也可以将自动化测试框架与报告自动生成框架结合使用,例如将 uiautomator2 与 pytest-html 相结合,通过添加--html=report.html 参数,生成测试报告。

3.3.4　是否支持重放

测试框架生成一系列可重现的测试用例,发送给移动设备进行执行。一旦在测试过程中发生异常,测试框架将重放导致异常的一系列操作,以便开发人员分析产生异常的原因。重放功能可以帮助开发人员更精准地定位到错误发生的位置,更易于分析错误发生的原因,从而使故障修复工作的效率大大提高。重放功能必须确保 APP 状态在录制和重放阶段相同,并尽可能地减轻重放延迟,即要求框架生成的事件拥有与录制事件相似的时间调度[42]。

3.3.5　测试环境

测试环境指在使用自动化测试框架对 AUT 进行 GUI 测试时 APP 的运行环境。主要分为两种。

(1)真实设备:指移动设备,如手机或平板电脑。在真实设备环境下进行测试,能够更真实地还原用户在使用 APP 时的操作。

(2)虚拟机:指在计算机上运行的能够模拟真实设备功能的虚拟移动设备。可以通过使用 Android 域的海量设备碎片来对 APP 进行测试。局限性在于无法模拟真机存在的电池电量不足、网络未连接等问题。

一般情况下,用户在对应用程序进行操作时都是在真实设备上运行的,因此,在测试框架对 APP 进行测试时,在真实设备环境下运行 APP 进行状态探索等至关重要。然而由于 APP 的不确定性、非标准控制流、可扩展性和开销限制等原因,在真实设备上进行测试具有一定的挑战性。不论是在真实设备还是在虚拟机上进行测试,都需要注意资源的分配,因为 CPU 竞争是影响 Android 响应性的症结所在[43],不合理的资源分配会导致 AUT 的响应慢,对测试结果产生影响。

3.3.6　支持的事件类型

在测试过程中,事件指用于执行测试用例的数据,分为 3 种。

(1)GUI 事件:指用户与 APP 的交互事件,包括点击、长按、滑动等操作。

(2)系统事件:指 APP 内部和用户的消息传递事件,包括短信通知、应用程序消息通知、电话等。

(3)文本事件:指用户在与 APP 进行交互的过程中,需要用户进行文本输入的事件,如账号登录等。

测试输入生成工具往往很难对一些需要人类知识的特殊情况做出正确的反应,如账号密码登录、首次使用指南和解锁屏幕等场景。极大部分的自动化测试框架只模拟用户的按键输入、触摸屏输入、手势输入等,并不考虑系统事件和文本事件。然而短信、音视频设备请求、设备电量不足的通知信息以及用户的登录文本信息是很常见的事件,在测试过程中考虑到这些场景能够更加充分地说明问题。

Dynodroid 对系统事件和文本事件进行了处理，Dynodroid 允许用户观察 APP 在生成输入事件时的反应，并允许用户暂停系统的事件生成，手动生成任意事件，并恢复系统的事件生成，整个系统结合了自动化和手动方法，这种结合的方式为处理文本信息提供了可能。Droidbot 处理文本事件的方式是提出了脚本机制来支持用户引导测试，基于设定的机制，用户可以自定义 Droidbot 进入特定的状态。以用户名密码登录为例，用户可以通过编写登录操作脚本来帮助框架通过 APP 的登录界面。而其他需要编写测试代码的框架，例如 uiautomator2，instrumentation 等，则需要编写具体的代码来处理文本事件。

3.3.7 是否使用 APP 源码

是否使用 APP 源码指自动化测试框架在对 AUT 进行测试时，是否使用了 APP 的源代码。在对自动化测试框架进行评估时，对开源 APP 进行测试，能使评估结果更有可信度。目前，无论使用的是白盒、黑盒还是灰盒测试策略，评估一个自动化测试框架的标准大都是其测试覆盖率（包括代码覆盖率、活动覆盖率、方法覆盖率等）。因此，在对测试框架进行测评时，首先需要获得 AUT 的源代码，从而获取 AUT 中的各种信息，进而计算框架的测试覆盖率。Pan 等开发的 Android 代码覆盖工具 ACVTool[44] 能够专门测量细粒度代码覆盖率，通过黑盒设置在类、方法和指令粒度上对测试框架的代码覆盖率进行测量。现有的测试方法主要使用 EMMA 来计算开源应用程序的测试覆盖率，但目前市场上很大部分的 APP 都对源代码进行了加壳处理。而 Droidbot 测试框架开辟了新的评估方法，其通过调用堆栈作为近似指标来量化测试输入的有效性，在测试过程中能够为每个测试输入生成包含测试输入触发的应用程序方法和系统方法的调用堆栈跟踪。

3.3.8 是否开源

是否开源指自动化测试框架的源代码是否供公众访问。测试框架的开源能够降低其维护成本，并且会有大批具有相同需求的开发人员对它进行不断改进，促进了开发人员之间的技术交流，更有利于自动化测试技术的发展。

3.3.9 系统事件识别方法

系统事件识别方法指从待测 APP 中提取系统事件使用的方法。

(1)静态分析：指根据 APP 提供的源代码来检测事件，无需执行 APP。静态分析的优点是提取系统事件的时间较短，并且提取的事件齐全。但为了安全起见，现阶段开发人员都会在 APP 发布前对其进行加固操作，导致测试人员无法获取 APP 源代码，从而无法进行静态分析，且静态分析的结果并不一定都会在真实情况下调用，可能对测试结果分析阶段产生影响。

(2)动态分析：指在 APP 运行时，动态地分析探索 GUI 界面，并提取出所有可能的事件序列的过程。动态分析的优点是能更加准确地分析软件真实调用事件的情况，缺点是消耗资源较多，耗费时间较长，并且可能存在无法触发的事件。Mu 等提出的 CoLaFUZE[45] 能够自动生成有效输入，探索驱动程序代码，利用动态分析和符号执行来为接口生成有效的输入。与目前最先进的 fuzzer 相比，该工具可以探索达到更高的代码覆盖率，并且发现更多的漏洞。Koroglu 等提出了第一个完全自动化的强化学习驱动的基于规范的测试生成器 FARLEAD-Android[46]，其通过动态执行被测应用程序，并监控 LTL 公式，来学习如何使用强化学习为 UI 测试场景生成见证。

(3)动静结合分析：将静态分析和动态分析相结合。一般先进行静态分析，再进行动态分析，动静结合分析的优点是准确率会更高，缺点是由于动态分析所需资源较多，将花费较长时间。

表 1 列出了现有的自动化测试方法对应的测试框架，共有 40 种，其排名不分先后。第 3 节所述内容是对表 1 中自动化测试框架的具体介绍、分析和对比。

表 1 自动化测试框架对比图

Table 1 Comparison diagram of automated test frameworks

Tool Name	Testing Strategy	Exploration Strategy	Crash Report	Whether playback is supported	Testing Environment	Event Type	Whether to use source code	Whether open source	System eventidentification method
Monkey	Black-box	Random	√	√	Both	GUI/System/Text	×	√	Dynamic Analysis
Dynodroid	Black-box	Random	√	×	Emulators	GUI/System/Text	√	√	Dynamic Analysis
IntentFuzzer	White-box	Random	×	×	Emulators	System	×	√	Static Analysis
VANARSena[47]	Gray-box	Random	×	√	Emulators	GUI/System/Text	√	×	Dynamic Analysis
Android Ripper	Black-box	System	×	×	Emulators	GUI/Text	√	√	Combination
ACTEve	White-box	System	×	×	Both	GUI	√	×	Dynamic Analysis
A3E Depth-First	Black-box	System	×	×	Both	GUI	×	√	Combination
A3E Targeted	Gray-box	Model	×	×	Both	GUI	×	×	Static Analysis
Crashscope[48]	Gray-box	System	√	√	Both	GUI/System/Text	√	×	Combination
GoogleRoboTest[49]	Black-box	System	√	×	Both	GUI/Text	N/A	×	N/A
MobiGUItar	Black-box	Model	×	√	N/A	GUI/Text	√	√	Dynamic Analysis
Swifthand	Black-box	Model	×	×	Both	GUI/Text	√	√	Dynamic Analysis
QUANTUM[50]	Black-box	Model	×	√	Emulators	GUI/System	N/A	×	N/A
ORBIT	Gray-box	Model	×	×	N/A	GUI	√	×	Combination
MonkeyLab	Black-box	Model	×	√	Both	GUI/Text	√	×	Combination
PUMA	Black-box	Other	×	×	Both	GUI/System/Text	×	√	Dynamic Analysis
JPF-Android[51]	White-box	Other	×	√	N/A	GUI	√	√	Dynamic Analysis
CrashDroid[52]	White-box	Other	√	√	Both	GUI/Text	√	×	Dynamic Analysis
Collider	White-box	Other	×	√	Emulators	GUI	√	×	Static Analysis
SIG-Droid	White-box	Other	×	√	Emulators	GUI/System/Text	√	×	Static Analysis
Thor[53]	Black-box	Other	×	×	Emulators	Test Case Event	√	√	N/A

(续表)

Tool Name	Testing Strategy	Exploration Strategy	Crash Report	Whether playback is supported	Testing Environment	Event Type	Whether to use source code	Whether open source	System eventidentification method
AppDoctor	Black-box	Other	√	√	Both	GUI/System/Text	×	×	Dynamic Analysis
EvoDroid	White-box	Other	×	×	Emulators	GUI	√	×	Static Analysis
Sapienz	Gray-box	Other	√	√	Both	GUI/System/Text	√	√	Combination
DroidFuzzer[54]	Black-box	Random	√	×	Both	GUI	×	√	Dynamic Analysis
GUIRipper[55]	Black-box	Model	√	×	N/A	GUI	×	√	Dynamic Analysis
Humanoid	Black-box	Other	√	√	Both	GUI/System	√	√	Dynamic Analysis
AndroFrame[56]	Black-box	Other	×	√	Both	GUI/System	√	×	Dynamic Analysis
Droidbot	Black-box	Model	√	√	Both	GUI/System	×	√	Dynamic Analysis
SmartMonkey[57]	Black-box	Random	×	×	N/A	GUI/System	N/A	×	Dynamic Analysis
Stoat	Gray-box	Model	√	√	Both	GUI/System	√	√	Static Analysis
ExtendedRipper[58]	Black-box	Model	√	×	Emulators	GUI/System	√	×	Dynamic Analysis
Fastbot	Black-box	Model	√	√	Devices	GUI	×	√	Dynamic Analysis
Jabbarvand[59]	Black-box	Other	√	√	Both	GUI/System/Text	√	√	Static Analysis
TrimDroid[60]	White-box	Other	√	√	Both	GUI	√	√	Static Analysis
Combodroid	Black-box	Other	√	√	Emulators	GUI/Text	√	√	Static Analysis
Kraken	Black-box	Random	√	×	Emulators	GUI/Text	×	√	Dynamic Analysis
Adamant	White-box	Model	√	×	Emulators	GUI/System	√	√	Combination
FARLEAD-Android	Black-box	Other	N/A	√	Emulators	GUI/System	N/A	N/A	Dynamic Analysis
CHARD	Black-box	Model	√	√	Emulators	GUI/System/Text	√	×	Dynamic Analysis

4　实验结果对比

本实验使用爬虫技术在 ApkFab 网站[1)] 收集了 16 种不同类别的应用程序来进行测试,共计 167 个。并选取了较有代表性的自动化测试工具(策略)进行了框架性能的对比实验,分别为:Monkey,Droidbot(dfs_naive),Droidbot(memory_guided),Humanoid,Fastbot。测试环境为:Windows10 操作系统;CPU:Intel Core i7-6700 处理器;16 GB 内存;Python3.9 脚本语言;GPU:GTX950M;Pycharm 2020;夜神模拟器。

实验使用 Activity 覆盖率来评估框架的测试性能,Activity 覆盖率也被称为活动覆盖率,是评估自动化测试工具的一种标准,用来度量测试的全面性、完整性。为了适应大多数应用程序的运行环境,这里使用 Android 5.1 来进行测试,并在实验过程中记录了 AUT 在不同测试时间段分别达到的活动覆盖率。对比实验共重复 3 次,取其平均值作为最终的实验结果。

我们将收集到的应用程序运行在选取的测试工具(策略)上进行测试,测试时间为 2 h,其 Activity 覆盖率的总体对比情况如图 2 所示。从平均值来看,Humanoid 实现了27.52% 的 Activity 覆盖率,在所选取的测试工具中最高。

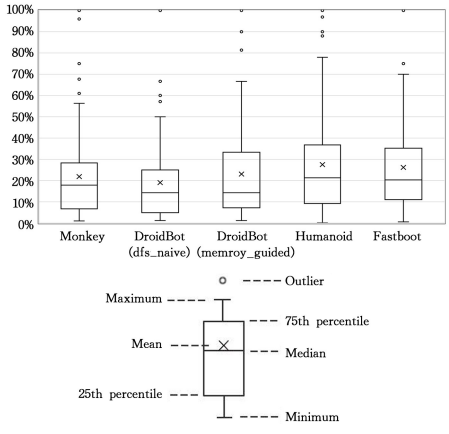

图 2　不同测试工具的 Activity 覆盖率对比

Fig. 2　Comparison of Activity coverage of different test tools

尽管 Monkey 的随机策略与其他工具的探索策略相比不够智能,但能够高速地生成测试输入事件,发送给设备执行。其产生的事件类型也更加多样,能够产生其他测试工具所不能生成的事件类型,如屏幕旋转事件、广播和意图,这使得 Monkey 在测试前期具有较好的测试覆盖率。然而 Monkey 的随机策略会产生大量重复且没有实际效果的输入事件,导致后期 Monkey 的 Activity 覆盖率不会有显著提高。Fastbot 继承自 Monkey,允许每秒高达 12 个事件的快速动作插入。其使用基于模型的探索策略,对 GUI 状态转换过程进行建模,用来发现 AUT 的稳定性问题。它将机器学习和强化学习相结合,能够以更智能的方式对用户界面进行探索,这样的设计使得其在实验中得到了更高的 Activity 覆盖率,仅次于 Humanoid。Droidbot 在输入事件之前要对最新的用户界面转换图进行分析,以相应的策略生成输入事件,并在生成事件后进行用户界面转换图的更新,这就使得其在相同的测试时间内所生成的事件数相比 Monkey 与 Fastbot 大大减少。Droidbot 提供了 5 种不同策略来生成测试输入,分别为 dfs_naive,dfs_greedy,bfs_naive,bfs_greedy,memory_guided,本次实验中选取了较有代表性的 dfs_naive 和 memory_guided 策略来进行对比实验,其中 dfs_naive 是其默认的探索策略,平均每小时约生成 504 个事件;而 memory_guided 是 Droidbot 于 2021 年最新提出的策略,使用了机器学习来自动识别相似的视图并且避免了冗余探索,平均每小时约能生成 414 个事件。相比默认的探索策略,尽管 memory_guided 策略花费了更多的时间来识别相似图片,但是其 Activity 覆盖率有所提高。Humanoid 的 Activity 覆盖率相对较高,它使用深度学习技术从人类与应用程序的交互中学习,以达到像人类一样探索应用程序的效果。Humanoid 的测试过程需要与 Droidbot 进行合作,Droidbot 根据当前的模型转换图生成几个可能的输入事件,Humanoid 对生成的事件进行优先级排序,优先选择最有可能被人类输入的事件,这样的处理使得 Humanoid 的一系列操作更加接近人类的思想,从而更有可能探索到应用程序的核心功能。

1) APKFab. APKFab[EB/OL]. [2021-12-10]. https://apkfab.com/

结束语 由于测试人员的测试时间紧迫，对测试的重视程度不够或者自身编程能力不足等原因，如何保证 APP 的质量成为一项挑战。在 APP 数量急剧增加的背景之下，自动化测试的研究就变得极有价值，APP 的发展势必将不断推动自动化测试前进。本文主要介绍了当前 Android 自动化测试的发展背景，分别从录制回放阶段、模块化和库阶段、测试框架阶段、混合框架阶段叙述了自动化测试的发展历程，并从测试策略、探索策略、错误报告、是否支持重放、测试环境、支持的事件类型、是否使用 APP 源码、是否开源、系统事件识别方法这 9 个方面来对自动化测试框架进行分类。文中对每个分类进行了概念阐述，对部分类别进行向下的再次分类，对每一类都进行了相应自动化测试框架的介绍和优缺点分析，并通过图表形式对 40 个自动化测试框架进行了对比。通过对现今 GUI 测试的相关研究，我们分析出目前学术界和市场上的自动化测试框架仍然存在以下几个问题值得深入研究和改进。

(1)UI 文本信息的表达与解读问题。文本信息在 APP 中是非常重要的一部分，几乎每一个 UI 界面都会存在文本内容来引导和提示人们使用特定功能。然而当前的大部分自动化测试框架，并没有考虑到 UI 界面的文本信息，而仅仅使用 UI 框架来表示模型中的每个 UI 状态，如果文本信息能够得到正确的表达和解读，那么一些框架的性能将会得到进一步的提高。

(2)截图存储问题。GUI 测试过程中，测试框架将会对 AUT 不同状态的 UI 界面进行截图、保存、分析、建模以预测下一步的测试输入。然而 AUT 运行的时间越长，存储的截图数量也随之增多。以基于模型的 GUI 测试框架为例，内存大小和计算能力会限制测试框架的测试性能，如果将消耗大量内存与计算资源的部分(如 AUT 不同状态的 UI 截图)部署到云端，那么 AUT 运行的时间将可以大幅度延长且测试速度不会受到太大影响。

(3)特定信息的输入问题。虽然现在开发的很多测试框架都有不错的测试覆盖率，但是相对于完美覆盖率仍然有一定差距，特别地，我们在实验过程中发现在对一些 APP 进行测试时，其覆盖率不超过 10%。这是因为许多 APP 在使用时需要特定的输入，比如电子邮件和密码，手机号和验证码等等，这些输入很难甚至不可能自动生成。当前存在的测试框架通过提前编写测试脚本的方式来解决这个问题，但这样的解决方式对测试人员的编码能力有较高要求。一个可能的解决办法是设计出一个半自动测试方法，使测试人员可以用最少的操作为自动测试框架提供必要的指导。

(4)GUI 测试的脆弱性。与基于 Web 的 APP 一样，移动端的 APP 也存在测试脆弱性问题。脆弱性问题指在测试过程中，由于 GUI 的微小变化或在后续版本进行微小改动而导致的需要对整个测试套件进行修改的问题。未来工作的目标是执行已修改和未修改的测试用例，来判别出其是否因脆弱性而无法使用。

(5)并行测试。目前学术界和市场上提出的测试框架大都在同一时间仅支持一个 APP 的测试，大大影响了测试效率。应用市场上 APP 数量越来越多，增长速度也越来越快，对 APP 的单个测试已不能迎合市场上的需求。因此，在未来工作中，需要提出更加有效的能够针对可扩展数量的物理设备并行自动测试 Android 多媒体应用程序的测试框架，以提高测试效率。

参 考 文 献

[1] China Internet Network Information Center. The 47th China Statistical Report on Internet Development[R]. Beijing: China Internet Network Information Center, 2021.

[2] NASS M, ALÉGROTH E, FELDT R. Why many challenges with GUI test automation (will) remain[J]. Information and Software Technology. 2021, 138: 106625.

[3] KONG P, LI L, GAO J, et al. Automated Testing of Android Apps: A Systematic Literature Review[J]. IEEE Transactions on Reliability. 2019, 68(1): 45-66.

[4] PRASAD K. Software Testing Tools: Covering WinRunner, Silk Test, LoadRunner, JMeter and TestDirector with case studies w/CD[M]. New Delhi: Dreamtech Press, 2004.

[5] Google. Ui/application exerciser monkey[EB/OL]. [2021-09-26]. https://developer.android.com/studio/test/monkey.html.

[6] MACHIRY A, TAHILIANI R, NAIK M. Dynodroid: an input generation system for Android apps[C]// Proceedings of the 2013 9th Joint Meeting on Foundations of Software Engineering. 2013: 224-234.

[7] LI Y, YANG Z, GUO Y, et al. Droidbot: a lightweight ui-guided test input generator for android[C]// 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). IEEE, 2017: 23-26.

[8] ARDITO L, COPPOLA R, LEONARDI S, et al. Automated Test Selection for Android Apps Based on APK and Activity Classification[J]. IEEE Access, 2020, 8: 187648-187670.

[9] SASNAUSKAS R, REGEHR J. Intent fuzzer: crafting intents of death[C]// Proceedings of the 2014 Joint International Workshop on Dynamic Analysis (WODA) and Software and System Performance Testing, Debugging, and Analytics (PERTEA). 2014: 1-5.

[10] ANAND S, NAIK M, HARROLD M J, et al. Automated concolic testing of smartphone apps[C]// Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. 2012: 1-11.

[11] MAHMOOD R, MIRZAEI N, MALEK S. Evodroid: Segmented evolutionary testing of android apps[C]// Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. 2014: 599-609.

[12] HAO S, LIU B, NATH S, et al. Puma: Programmable ui-automation for large-scale dynamic analysis of mobile apps[C]// Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services. 2014: 204-217.

[13] NGUYEN D M, HUYNH Q T, HA N H, et al. Automated Test Input Generation via Model Inference Based on User Story and Acceptance Criteria for Mobile Application Development[J]. International Journal of Software Engineering and Knowledge Engineering, 2020, 30(3): 399-425.

[14] SU T, MENG G, CHEN Y, et al. Guided, stochastic model-based GUI testing of Android apps[C]// Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. 2017: 245-256.

[15] YANG W, PRASAD M R, XIE T. A grey-box approach for au-

tomated GUI-model generation of mobile applications[M]. Berlin:Springer,2013:250-265.

[16] MAO K,HARMAN M,JIA Y. Sapienz:Multi-objective automated testing for android applications[C]// Proceedings of the 25th International Symposium on Software Testing and Analysis. 2016:94-105.

[17] YASIN H N,HAMID S H A,RAJA Y R J. DroidbotX: Test Case Generation Tool for Android Applications Using Q-Learning[J]. Symmetry,2021,13(2):310.

[18] YAN J,ZHOU H,DENG X,et al. Efficient testing of GUI applications by event sequence reduction[J]. Science of Computer Programming,2021,201:102522.

[19] YASIN H N,HAMID S H A,YUSOF R J R,et al. An empirical analysis of test input generation tools for android apps through a sequence of events[J]. Symmetry,2020,12(11):1894.

[20] DING R M,SHENG J,CHEN H T,et al. Tencent Android automated testing practice [M]. Beijing: Machinery Industry Press,2016.

[21] YAN J,ZHOU H,DENG X,et al. Efficient testing of GUI applications by event sequence reduction[J]. Science of Computer Programming. 2021,201:102522.

[22] HUANG R,SUN W,XU Y,et al. A Survey on Adaptive Random Testing[J]. IEEE Transactions on Software Engineering. 2021,47(10):2052-2083.

[23] NOVELLA L,TUFO M,FIENGO G. Automatic Test Set Generation for Event-Driven Systems in the Absence of Specifications Combining Testing with Model Inference[J]. Information Technology and Control,2019,48(2):316-334.

[24] CAI T Q. Fastbot:moving smart monkey[EB/OL]. (2020-09-28) [2021-9-22]. https://mp. weixin. qq. com/s/3t4H2bfDjei4vXkj_Cz2pg.

[25] CHOI W,NECULA G,SEN K. Guided gui testing of android apps with minimal restart and approximate learning[J]. ACM Sigplan Notices,2013,48(10):623-640.

[26] AMALFITANO D,FASOLINO A R,TRAMONTANA P,et al. Using GUI ripping for automated testing of Android applications[C]// 2012 Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering. IEEE,2012:258-261.

[27] AMALFITANO D,FASOLINO A R,TRAMONTANA P,et al. MobiGUITAR:Automated model-based testing of mobile apps[J]. IEEE Software,2014,32(5):53-59.

[28] BAEK Y M,BAE D H. Automated model-based android gui testing using multi-level gui comparison criteria[C]// Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. 2016:238-249.

[29] LINARES-VÁSQUEZ M,WHITE M,BERNAL-CÁRDENAS C,et al. Mining android app usages for generating actionable gui-based execution scenarios[C]// 2015 IEEE/ACM 12th Working Conference on Mining Software Repositories. IEEE,2015:111-122.

[30] AMALFITANO D,RICCIO V,AMATUCCI N,et al. Combining automated gui exploration of android apps with capture and replay through machine learning[J]. Information and Software Technology,2019,105:95-116.

[31] RAVELO-MÉNDEZ W,ESCOBAR-VELÁSQUEZ C,LINARES-VÁSQUEZ M. Kraken:A framework for enabling multi-device interaction-based testing of Android apps[J]. Science of Computer Programming,2021,206:102627.

[32] AZIM T,NEAMTIU I. Targeted and depth-first exploration for systematic testing of android apps[C]// Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications. 2013:641-660.

[33] JENSEN C S,PRASAD M R,MØLLER A. Automated testing with targeted event sequence generation[C]// Proceedings of the 2013 International Symposium on Software Testing and Analysis. 2013:67-77.

[34] MIRZAEI N,BAGHERI H,MAHMOOD R,et al. Sig-droid:Automated system input generation for android applications [C]// 2015 IEEE 26th International Symposium on Software Reliability Engineering(ISSRE). IEEE,2015:461-471.

[35] HU G,YUAN X,TANG Y,et al. Efficiently,effectively detecting mobile app bugs with appdoctor[C]// Proceedings of the Ninth European Conference on Computer Systems. 2014:1-15.

[36] LI Y,YANG Z,GUO Y,et al. A deep learning based approach to automated android app testing[J]. arXiv:1901. 02633,2019.

[37] PAN M,LU Y,PEI Y,et al. Effective testing of Android apps using extended IFML models[J]. Journal of Systems and Software,2020,159:110433.

[38] AMALFITANO D,RICCIO V,AMATUCCI N,et al. Combining automated gui exploration of android apps with capture and replay through machine learning[J]. Information and Software Technology,2019,105:95-116.

[39] PACKEVIČIUS Š,RUDIONIEN G,BAREIA E. Automated Visual Testing of Application User Interfaces Using Static Analysis of Screenshots[J]. International Journal of Software Engineering and Knowledge Engineering,2021,31(2):167-191.

[40] WANG J,JIANG Y,XU C,et al. ComboDroid:generating high-quality test inputs for Android apps via use case combinations [C]// Proceedings of the ACM/IEEE 42nd International Conference on Software Engineering. 2020:469-480.

[41] LIU C. A compatibility testing platform for android multimedia applications [J]. Multimedia Tools and Applications, 2019, 78(4):4885-4904.

[42] LIU Y,YAN F,XIA M,et al. TimelyRep:Timing deterministic replay for Android web applications[J]. Software Testing,Verification and Reliability,2020,30(4/5):e1745.

[43] FU J,WANG Y,ZHOU Y,et al. How resource utilization influences UI responsiveness of Android software[J]. Information and Software Technology,2022,141:106728.

[44] PILGUN A,GADYATSKAYA O,ZHAUNIAROVICH Y,et al. Fine-grained Code Coverage Measurement in Automated Black-box Android Testing[J]. ACM Transactions on Software Engineering and Methodology,2020,29(4):1-35.

[45] MU T,ZHANG H,WANG J,et al. CoLaFUZE:Coverage-Guided and Layout-Aware Fuzzing for Android Drivers[J]. IEICE Transactions on Information and Systems, 2021, 104(11):1902-1912.

[46] KOROGLU Y,SEN A. Functional test generation from UI test

scenarios using reinforcement learning for android applications [J]. Software Testing, Verification and Reliability, 2020, 31(3): e1752.

[47] RAVINDRANATH L, NATH S, PADHYE J, et al. Automatic and scalable fault detection for mobile applications[C]// Proceedings of the 12th Annual International Conference on Mobile Systems, Applications, and Services. 2014: 190-203.

[48] MORAN K, LINARES-VÁSQUEZ M, BERNAL-CÁRDENAS C, et al. Crashscope: A practical tool for automated testing of android applications[C]// 2017 IEEE/ACM 39th International Conference on Software Engineering Companion (ICSE-C). IEEE, 2017: 15-18.

[49] Google. Google firebase test lab robo test. (2021-4-28)[2021-9-26]. https://firebase.google.com/docs/test-lab/robo-ux-test.

[50] ZAEEM R N, PRASAD M R, KHURSHID S. Automated generation of oracles for testing user-interaction features of mobile apps[C]// 2014 IEEE Seventh International Conference on Software Testing, Verification and Validation. IEEE, 2014: 183-192.

[51] VAN DER MERWE H, VAN DER MERWE B, VISSER W. Execution and property specifications for jpf-android[J]. ACM SIGSOFT Software Engineering Notes, 2014, 39(1): 1-5.

[52] WHITE M, LINARES-VÁSQUEZ M, JOHNSON P, et al. Generating reproducible and replayable bug reports from android application crashes[C]// 2015 IEEE 23rd International Conference on Program Comprehension. IEEE, 2015: 48-59.

[53] ADAMSEN C Q, MEZZETTI G, MØLLER A. Systematic execution of android test suites in adverse conditions[C]// Proceedings of the 2015 International Symposium on Software Testing and Analysis. 2015: 83-93.

[54] YE H, CHENG S, ZHANG L, et al. Droidfuzzer: Fuzzing the android apps with intent-filter tag[C]// Proceedings of International Conference on Advances in Mobile Computing & Multimedia. 2013: 68-74.

[55] AMALFITANO D, FASOLINO A R, TRAMONTANA P, et al. A toolset for GUI testing of Android applications[C]// 2012 28th IEEE International Conference on Software Maintenance (ICSM). IEEE, 2012: 650-653.

[56] KOROGLU Y, SEN A, MUSLU O, et al. QBE: Q Learning-based exploration of android applications[C]// 2018 IEEE 11th International Conference on Software Testing, Verification and Validation (ICST). IEEE, 2018: 105-115.

[57] RUIZ X, CALVET L, FERRARONS J, et al. SmartMonkey: a web browser tool for solving combinatorial optimization problems in real time[C]// International Forum for Interdisciplinary Mathematics. Cham: Springer, 2015: 74-86.

[58] AMALFITANO D, FASOLINO A R, TRAMONTANA P, et al. Considering context events in event-based testing of mobile applications[C]// 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops. IEEE, 2013: 126-133.

[59] JABBARVAND R, SADEGHI A, BAGHERI H, et al. Energy-aware test-suite minimization for android apps[C]// Proceedings of the 25th International Symposium on Software Testing and Analysis. 2016: 425-436.

[60] MIRZAEI N, GARCIA J, BAGHERI H, et al. Reducing combinatorics in GUI testing of android applications[C]// 2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE). IEEE, 2016: 559-570.

YANG Yi, born in 1998, master. Her main research interests include Android automated testing and so on.

ZHAO Chun-lei, born in 1979, Ph.D, associate professor, is a member of China Computer Federation. Her main research interests include cybersecurity and so on.