软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2022,33(5):1612−1634 [doi: 10.13328/j.cnki.jos.006551]

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

基于 GUI 事件的安卓应用录制重放关键技术综述*

李 聪[1,2], 蒋炎岩[1,2], 许 畅[1,2]

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

[2](南京大学 计算机科学与技术系, 江苏 南京 210023)

通信作者: 蒋炎岩, E-mail: jyy@nju.edu.cn

摘 要: 基于 GUI 事件的安卓应用录制重放技术致力于以自动化的方式捕捉和回放人类和移动应用的交互轨迹, 达到降低测试成本、提高测试用例复用率的目的. 录制重放技术面临的挑战来源于应用、版本和设备 3 个维度. 试图从人类录制重放的角度, 将录制重放建模为一个搜索问题, 并提出模拟人类录制重放行为的通用框架. 框架包含 3 部分: 组件表示与录制技术、事件等价策略和局部搜索策略. 通过对已有技术进行总结和分析, 以全新的视角更好地认识了已有工作的优势和不足, 并提出未来可行的研究方向.

关键词: 安卓应用; 软件测试; 录制重放; GUI 测试

中图法分类号: TP311

中文引用格式: 李聪, 蒋炎岩, 许畅. 基于GUI事件的安卓应用录制重放关键技术综述. 软件学报, 2022, 33(5): 1612–1634. http://www.jos.org.cn/1000-9825/6551.htm

英文引用格式: Li C, Jiang YY, Xu C. GUI Event-based Record and Replay Technologies for Android Apps: A Survey. Ruan Jian Xue Bao/Journal of Software, 2022, 33(5): 1612–1634 (in Chinese). http://www.jos.org.cn/1000-9825/6551.htm

GUI Event-based Record and Replay Technologies for Android Apps: A Survey

LI Cong[1,2], JIANG Yan-Yan[1,2], XU Chang[1,2]

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

[2](Department of Computer Science and Technology, Nanjing University, Nanjing 210023, China)

Abstract: GUI event-based record and replay technologies for Android apps aim at automatically capturing and playing back the UI interactions between users and apps. Record and replay are challenging because it involves a cross-understanding of three different program semantics: application difference, version evolution, and device compatibility. This study models record and replay as a search problem, and analyzes this problem from a human perspective. Accordingly, this study proposes a general framework to demonstrate the key points in record and replay: the widget representation and recording technologies, the event semantic equivalence strategies, and the local search strategies. By summarizing and analyzing existing technologies from a new perspective that is suitable for the framework, this study has a better understanding of the advantages and disadvantages of existing technologies and proposes feasible future research directions.

Key words: Android app; software testing; record and replay; GUI testing

1 引 言

截至 2021 年 5 月, Google Play Store 已拥有超过 290 万个安卓应用程序 (Android applications)[1], 占据接近 80% 的市场份额[2]. 移动应用的缺陷 (bug) 和漏洞 (valnerability) 也在迅速增长: 2016 年的数据显示, 市场中有至

* 基金项目: 国家自然科学基金重点项目 (61932021); 江苏省自然科学基金前沿引领技术基础研究专项课题 (BK20202001); 江苏省软件新技术与产业化协同创新中心资助

本文由“领域软件工程”专题特约编辑汤恩义副教授、江贺教授、陈俊洁副教授、李必信教授以及唐滨副教授推荐.

收稿时间: 2021-08-08; 修改时间: 2021-10-09; 采用时间: 2022-01-10; jos 在线出版时间: 2022-01-28

少 24.7% 的应用包含至少一个高危漏洞 (high-risk security flaw)[3], 对用户体验和隐私安全都带来极大影响. 提升安卓应用质量是应用开发者的重要任务, 也是学术界的热门研究话题[4−17].

目前, 由开发者撰写的功能性测试用例仍是保障应用质量的最有效手段[18], 是因为开发者具有应用逻辑的领域知识 (domain knowledge), 能够编写触发应用深层状态的高质量测试用例, 进而触发隐蔽的缺陷[15]. 然而, 人工编写并维护测试用例不仅成本昂贵且具有测试用例复用率低的特点, 体现在如下 3 方面.

(1) 安卓应用更新迭代迅速, 测试用例难以“跨版本”. 为了应对新旧版本的差异 (如应用界面变化), 开发和测试人员不得不为已有功能重新编写测试用例, 进而测试旧版本功能是否可以在新版本正确运行.

(2) 安卓生态碎片化 (fragmentation) 严重, 测试用例难以“跨设备”. 截至 2021 年, 世界上已经有超过 24 000 种设备型号 (device model)[19]和超过 5 个被官方维护的安卓版本[20]. 应用通常在不同屏幕大小、方向的设备上显示不同的界面, 故开发者须为主流设备同一功能 (如登录) 重复撰写测试用例.

(3) 开发者多发布同类应用 (如淘宝[21]和天猫[22]), 测试用例难以“跨类别”. 尽管同类应用有相似的界面和交互逻辑, 应用团队的开发者仍然针对每一应用每一功能 (如登录) 分别撰写测试用例, 测试用例复用率低下且浪费人力成本[23−25].

上述 3 方面因素共同导致安卓应用开发者在测试时面临版本 (version)、设备 (device) 和应用 (application) 这 3 个维度的挑战 (图 1). 在本文中, 我们将由应用、版本和设备 3 个维度构成的空间称为配置空间 (configuration space, *CS*). 该空间中的任意点 $(a, v, d) \in CS$ 称为配置点 (configuration point).

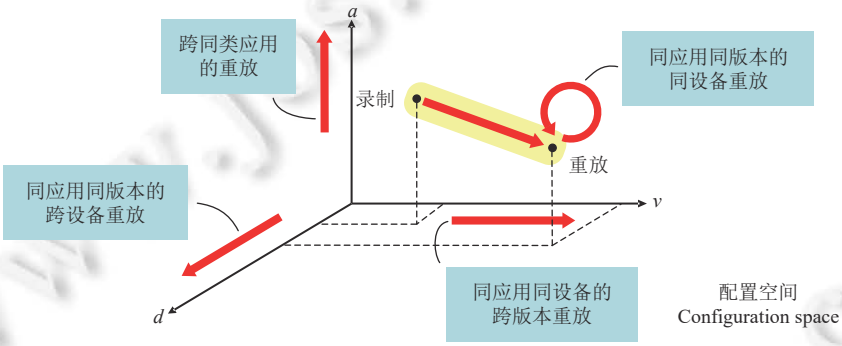

图 1 三维配置空间

基于 GUI 事件的录制重放技术 (GUI event-based record and replay technologies, 录制重放) 是降低测试成本、提高测试用例复用率的一种有效手段. 录制重放致力于自动化地捕捉和回放人类和移动应用的交互轨迹. 基于安卓系统事件驱动 (event-driven) 的特性, 理想的录制重放技术能将任意配置点上用户 (开发者) 与移动应用在一段时间内的交互序列转化为用事件描述的测试用例, 并在任意配置点重放事件序列, 实现测试用例“一次编写, 到处运行” (record once, replay anywhere), 减小人力成本、提高测试用例复用率.

然而, 实现在任意配置点上重放录制的事件序列面临极大挑战, 因其涉及 3 方面语义的交叉理解[26,27]: 应用在版本变迁时语义变化的理解、应用在跨设备实现时的语义等价理解以及同类应用的语义等价理解. 考虑到交叉语义理解所面临的挑战以及现实中录制重放任务的具体需求, 已有工作根据录制重放应用场景和任务的不同进行了一定范围的取舍 (图 2).

- 同应用同版本的同设备重放, 通常用于缺陷复现. 在该任务里, 由用户 (一般为终端用户) 录制应用的某个缺陷并连同应用版本和设备信息上报给开发者, 希望得到开发者的修复. 开发者根据该配置点 (应用、版本和设备) 的信息, 将对应版本的应用在相同的设备上进行重放, 并检视和修复缺陷.
- 同应用同版本的跨设备重放, 通常用于兼容性测试. 开发或测试人员为某一版本应用在某台设备上编写一份测试用例, 并希望通过录制重放技术在多台安装了同版本应用的其他设备上重放, 从而测试应用对设备的兼容性.

● 同应用同设备的跨版本重放, 通常用于回归测试. 应用新版本开发完成后, 开发和测试人员希望通过录制重放技术复用旧版本的测试用例来测试旧版本的功能是否可以在新版本上正确运行, 或希望通过重放技术对失效的测试脚本进行修复.

● 跨同类应用的同设备重放, 通常用于差分测试 (differential testing) 和测试迁移 (test migration). 此类任务基于同类应用有相似的界面和交互逻辑的假设, 希望复用同类型其他应用的测试用例来提前测试自有应用是否出现类似的缺陷, 并予以修复, 从而保证应用发布前期的质量.

基于上述 4 种录制重放场景, 本文从相关的国际会议和期刊中系统性地选取了 51 篇论文和 10 个工具, 方法如下: 在 Google Scholar 中使用关键词“record(-)and(-)replay”“capture(-)and(-)replay”“record(-)replay”“capture(-)replay”“GUI replay”“event-based replay”和“(non-)deterministic replay”搜索并人工筛选与“安卓应用录制重放”相关的工作, 最后在剩余工作中使用滚雪球法[28]对工作列表进行扩充. 图 3 展示了该 51 篇论文的分布. 可以看到, 基于 GUI 事件的安卓应用录制重放论文发表数量自 2013 年起呈现上升趋势, 说明该方向自彼时逐步得到关注并于 2018 年起得到广泛关注.

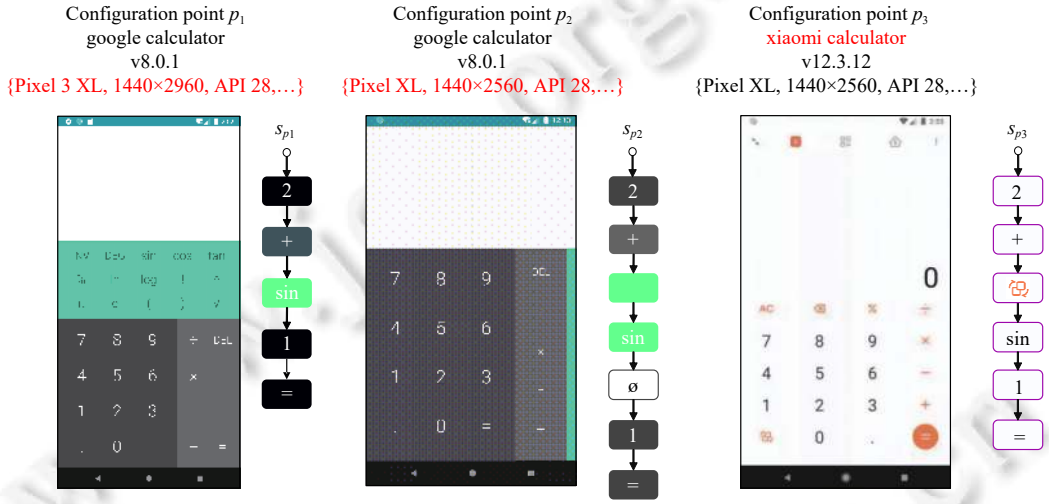

图 2 示例: 在不同的配置点录制重放“2+sin(1)=”

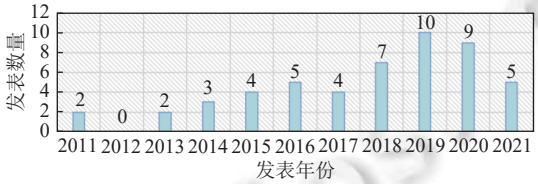

图 3 2011–2021.07 基于 GUI 事件的录制重放论文数量分布

该 51 篇论文包含 5 篇综述性质的论文: Flinn 等人[29], Lam 等人[26], Martino 等人[30], Zhao 等人[27] 和 Mariani 等人[31]. 其中, Finn 等人对确定性重放 (deterministic replay) 是否可以成为移动计算的一个有效工具进行了研究; Lam 等人对 2017 年前的录制重放工具进行了汇总, 并对它们是否可在工业环境中适用进行了分析; Martino 等人将录制重放工具用于测试输入生成 (test input generation), 并对其效果进行了对比; Zhao 等人提出了一个用于评估测试迁移 (test migration) 方法有效性的工具; Mariani 等人对测试迁移技术中使用的“事件语义等价”判定策略进行了深入探讨. 而本文基于前述 4 种场景, 提出了一个能涵盖目前所有录制重放工作的通用框架, 并力图从更高层次对录制重放技术进行系统的分析, 阐明已有工作在各方面选择的具体技术手段及原因, 分析已有工作使用于何种录制重放场景, 从而更好地认识已有工作的优势和不足、展望未来研究方向. 文末表 1 对该 51 篇论文和 10 个工

具根据本综述所提框架进行了分类展示, 这包含组件表示与录制技术、事件语义等价策略和局部搜索策略 3 部分.

本文第 2 节对录制重放进行了问题分析并提出了本篇综述的基本框架; 第 3、4 和 5 节分别基于框架中的 3 部分对已有工作进行了系统分类、分析和总结, 并给出各类工作在图 1 中的适用场景和任务; 第 6 节对已有工作进行了讨论; 第 7 节对录制重放的未来可行的研究方向进行了探讨; 第 8 节对全文进行了总结.

2 问题分析与综述框架

图 2 展示了在 3 个配置点录制“2+sin(1)=”的示例, 鉴于图中所有事件类型均为 click, 且不包含任何用户输入, 因此图中省略了事件类型和用户输入, 仅保留接收组件作为事件表示. 每个示例给出了计算器应用的版本、设备 GUI 界面和产生的 GUI 事件序列. 每个 GUI 界面上存在一个或多个蕴含应用语义的 GUI 组件, 如 2 代表语义“数字 2”. 本节使用图 2 辅助阐述本文所需基本概念、符号和问题定义.

2.1 事件与事件序列

GUI 事件 (GUI event), 本文简称事件. 遵从已有工作[9,11,15], 本文将 GUI 事件定义为事件类型 (type, 如点击 click、滑动 swipe、接收组件 (receiver widget) 和可选用户输入 (input value, 如在输入框中输入的文本) 构成的三元组 (three-value tuple). 如事件 $e=\langle t, w, z\rangle$ 表示类型为 t 的事件 e 可以将用户输入 z 发送给组件 w 从而触发应用特定行为. 本文用 $e.t$、$e.w$ 和 $e.z$ 分别表示事件 e 的类型、组件和用户输入.

GUI 事件序列 (GUI event sequence), 本文简称事件序列或序列, 是由事件构成的有序序列: $s = [e_1, e_2,..., e_{|s|}]$, 其中 $1\leqslant i<j\leqslant|s|$当且仅当事件 e_i 在事件 e_j 之前发生. 鉴于安卓应用由事件驱动, 主流的录制重放技术以事件序列表示测试用例. 本文将混用术语事件序列和测试用例, 读者应根据上下文辨别它们是否表示同一个概念. 特别地, Ø在本文中表示空事件序列.

考虑到测试用例在给定配置点进行录制或重放, 如有必要强调配置点信息, 本文将采用下标形式表示事件序列的配置点. 如 s_p 表示在配置点 $p = (a, v, d)$ 进行录制/重放得到的事件序列. 本文用 $s_p[i]$ 表示事件序列 s_p 的第 i 个事件, 用 $s_p[i:j]$ 表示 s_p 的子事件序列 $[s_p[i], s_p[i+1],..., s_p[j-1]]$, 其中 $1 \leqslant i < j \leqslant |s_p|$. 本文用函数 $pred$: $E\times S\to E$ 和 $succ$: $E\times S\to E$ 分别表示某事件的前驱和后继事件, 如 $pred(s_p[i], s_p) = s_p[i - 1]$ 和 $succ(s_p[i], s_p) = s_p[i + 1]$. 特别地, $pred(s_p[1], s_p) = succ(s_p[|s_p|], s_p) = \perp$.

2.2 问题定义与分析

移动应用设计语言各不相同, 触发同样功能在不同配置点将产生不同事件序列. 在图 2 中:

- 配置点 p_1 = (Google Calculator, v8.0.1, {Pixel 3 XL, 1440 × 2960, API 28, ... }): Google Calculator v8.0.1 在 Pixel 3 XL 上触发“2+sin(1)=”需 6 个事件;
- 配置点 p_2 = (Google Calculator, v8.0.1, {Pixel XL, 1440 × 2560, API 28, ... }): Google Calculator v8.0.1 在 Pixel XL 上触发“2+sin(1)=”需 8 个事件;
- 配置点 p_3 = (XiaoMi Calculator, v12.3.12, {Pixel XL, 1440 × 2560, API 28, ... }): Xiaomi Calculator v12.3.12 在 Pixel XL 上触发“2+sin(1)=”需 7 个事件.

基于 GUI 事件的安卓应用录制重放技术要实现在给定重放配置点搜索满足指定约束条件的事件序列. 这本质上是一个搜索问题 (search problem), 即:

给定可以判定重放是否成功的评判标准$\cong$: $S\times S\to\{\top,\perp\}$和事件序列 s_p, 其中 $p = (a, v, d) \in CS$, 录制重放技术致力于在重放配置点 $p' = (a', v', d') \in CS$ 搜索事件序列 $s_{p'}$, 使得:

$$s_p \cong s_{p'}, \text{即}, \cong (s_p,s_{p'}) = \top.$$

尽管概念上简单, 但解决该问题不可避免地面临以下两个挑战.

- 如何判断重放是否成功? 即, 评判标准$\cong(\cdot,\cdot)$是什么?
- 搜索空间 (search space) S 是什么、怎么构造, 以及如何进行搜索?

为完成录制重放并为之建立合理且通用的算法, 本文试图从人类重放事件序列的角度来理解和分析上述

问题.

2.3 人类重放: 分而治之

我们发现, 人类通常使用分而治之 (divide and conquer) 的方式重放事件序列. 人类具有应用的领域知识, 并将事件序列理解为应用的使用场景 (usage scenario). 在执行重放时, 通常会将一个使用场景分解为多个场景步骤 (scenario steps, 简称步骤). 这些步骤执行上相互独立, 语义上相互关联, 每个步骤承担了使用场景的部分语义. 图 2 例子中的 3 条事件序列均可分解为以下 5 个与触发“2+sin(1)=”语义相关的步骤.

(1) 在计算器中输入“数字 2”;

(2) 在计算器中输入“运算符 +”;

(3) 在计算器中输入“运算符 sin”;

(4) 在计算器中输入“数字 1”;

(5) 在计算器中输入“运算符 =”完成计算, 并查看运算结果.

为了完成上述 5 个步骤, 一个人类用户从 (前一步骤结束时所在的) 特定 GUI 界面出发, 不断探索应用并独立完成每个步骤. 对于步骤 (1), 人类用户从应用初始界面出发, 探索并寻找带有语义“数字 2”的组件 (如2或2), 并触发与动作“输入”语义相关的特定类型事件——点击. 完成步骤 (1) 后, 用户继续从当前界面出发, 不断为每个步骤独立地重复前述行为, 直到完成步骤 (5). 在某些步骤中, 用户为了能完成对应语义的事件如 $s_{p_2}[5]$ = 〈click, “sin”, ⊥〉, 会探索应用并触发一些与该步骤语义无关的额外事件如 $s_{p_2}[4]$ = 〈click, “”, ⊥〉.

本文发现, 尽管各步骤在不同的配置点表现为不同的事件序列 (如步骤 (3) 在 p_1 上表现为 $s_{p_1}[4{:}5]$, 在 p_2 和 p_3 上分别表现为 $s_{p_2}[4{:}6]$ 和 $s_{p_3}[4{:}6]$), 但事件序列中的事件可根据其与步骤语义的相关性分为以下两种:

● 核心事件 (core event): 与步骤语义强相关. 各步骤在各配置点仅对应一个核心事件. 尽管同一步骤在各配置点对应的核心事件不同 (如2或2), 但它们语义等价. 在本文中, 我们用函数 η: $S \times INT \to \{\top, \bot\}$来判断一个事件是否是对应序列的核心事件, 如 $\eta(s_{p_2}, 4) = \bot$和 $\eta(s_{p_2}, 5) = \top$.

● 辅助事件 (auxiliary event): 与步骤语义无关. 各步骤可对应零或多个辅助事件. 这些辅助事件用于帮助用户探索并寻找核心事件. 本文称辅助事件构成的事件序列为对应核心事件的辅助序列. 核心事件可以不存在辅助序列, 即辅助序列可以为空. 在图 2 中, s_{p_1} 的所有事件均为核心事件, 其任意事件的辅助序列为$\emptyset$; $s_{p_2}[5]$ 的辅助序列为 $s_{p_2}[4{:}5]$; $s_{p_3}[5]$ 的辅助序列为 $s_{p_3}[4{:}5]$.

在本文中, 我们称事件序列的核心事件按序构成的序列为该事件序列的核心序列, 并用函数 $H: S \to S$ 表示, 即,

$$H(s) = [s[h_1], s[h_2], ..., s[h_n]]$$

其中, $h_1, h_2, ..., h_n$ 满足如下约束:

$$\forall 1 \leqslant i < j \leqslant n. 1 \leqslant h_i < h_j \leqslant |s| \text{ and}$$
$$\forall 1 \leqslant i \leqslant n. \eta(s, h_i) = \top \text{ and}$$
$$\forall k < \{h_i | 1 \leqslant i \leqslant n\}. 1 \leqslant k \leqslant |s| \to \eta(s, k) = \bot.$$

在图 2 中,

$$H(s_{p_1}) = s_{p_1},$$
$$H(s_{p_2}) = [s_{p_2}[1], s_{p_2}[2], s_{p_2}[3], s_{p_2}[5], s_{p_2}[7], s_{p_2}[8]],$$
$$H(s_{p_3}) = [s_{p_3}[1], s_{p_3}[2], s_{p_3}[3], s_{p_3}[5], s_{p_3}[6], s_{p_3}[7]].$$

2.4 评判标准

人类分而治之的重放方法说明, 人类重放时致力于在重放配置点按序触发各步骤的核心事件, 即,

$$s_p \cong s_{p'} \text{ iff. } |H(s_p)| = |H(s_{p'})| \wedge \forall 1 \leqslant k \leqslant |H(s_p)|. \varphi(H(s_p)[k], H(s_{p'})[k]).$$

其中, 函数 φ: $E \times E \to \{\top, \bot\}$用于判断两个事件是否语义等价 (如点击2和点击2语义等价).

在具体的重放任务中, 除了使用上述评判标准, 一般还会额外添加适用于各任务的特殊约束, 如:

- 在缺陷重现中, 一般还要保证 $s_{p'}$能够触发对应的缺陷[32];
- 在测试迁移中, 一般还要保证断言 (assertion)[33]能够成功[23,27].

2.5 搜索空间

配置点 $p = (a, v, d)$ 有与之对应的搜索空间 S_p, 该搜索空间表示应用 a 的 v 版本在设备 p 上的所有事件序列集合. 该搜索空间由三元组 $S_p = \langle E, R, \varepsilon \rangle$ 表示:

- E 为该配置点的所有事件;
- $R : E \times E$ 表示事件先后关系, 如 $(e_1, e_2) \in R$ 表示事件 e_2 发生于事件 e_1 之后, 本文用 $e_1 \rightarrow^{S_p} e_2$ 表示, 我们称 e_1 为 e_2 的前驱事件, e_2 为 e_1 的后继事件;
- $\varepsilon \in E$ 表示启动事件.

搜索空间中由根节点到任意节点的路径 (path) 表示一条合法的事件序列.

在配置点 p'上进行重放的本质是在搜索空间 $S_{p'}$中搜索事件序列 $s_{p'}$使得 $s_p \cong s_{p'}$. 最朴素的搜索方式是全局暴力搜索 (global brute-force search): 尝试 $S_{p'}$中的每条事件序列, 并返回第一条使$\cong(\cdot,\cdot) = \top$的事件序列. 全局暴力搜索诚然简单, 但一个安卓应用能产生海量的事件序列, 巨大的搜索空间使得暴力搜索效率低下.

人类采取了分而治之的方式应对巨大的搜索空间: 根据语义将事件序列 s_p 分解为执行步骤, 并从应用某个状态出发的局部空间里, 为各步骤的核心事件搜索辅助序列. 该策略利用应用事件驱动的性质, 减小搜索空间, 提升搜索效率.

2.6 综述框架

算法 1. 录制重放通用算法框架.

```
1  Function Replay(s, (a, v, d), (a′, v′, d′), φ)
2    s′ ← Ø;
3    for each event e ∈ s with index i do
4      l, e′ ← LocalSearch(e, (a′, v′, d′), φ); // 为 e 搜索辅助序列和语义等价事件
5      if e′= ⊥ then
6        if is-aux(s, i) then continue; // 通过启发式规则判断: 丢弃辅助事件并继续执行
7        else abort; // 非辅助事件时通知用户重放失败
8      s′← s′ : : l : : [e′];
9      exec-seq(l: :[e′], (a′, v′, d′));
10   return s′;
```

本文提出一个解决录制重放问题的合理且通用的算法框架. 该框架的合理性来源于本文对人类录制重放行为的观察: 即, 人类通常采用分而治之、局部搜索的模式进行重放. 基于此观察, 本框架按序为目标事件序列的核心事件根据应用当前状态进行局部搜索. 考虑到辅助事件在各配置点表现形式有所差异, 且不存在语义相关事件, 本框架亦借鉴已有工作, 采用先乐观假设后启发式验证的方式进行核心/辅助事件的判断. 具体而言, 本框架乐观地假设序列中的各事件均为核心事件, 并为之搜索辅助序列和语义等价事件. 搜索失败时, 通过一些启发式规则判断其是否为辅助事件并丢弃判断为真的事件, 继续执行, 否则通知用户重放失败. 为了实现一个通用的录制重放算法, 本文仔细对比了现有的录制技术和重放算法, 提取其共性、抽象其差异 (表现在组件表示与录制技术、事件语义等价策略和局部搜索策略 3 方面), 从而总结出算法 1 所示的通用框架. 为使用该框架实现一个具体的录制重放算法, 开发者需要:

- 选择合适的组件表示方式及录制技术, 从而保证事件序列 (s') 能被正确录制、表示和执行;
- 实现合适的事件语义等价策略 (φ), 从而可以对录制时事件和重放时事件进行语义等价性判定;
- 实现合适的局部搜索策略 (LocalSearch), 从而可以为核心事件搜索辅助序列.

具体而言, 该算法以录制事件序列 s'、录制配置点 (a, v, d)、重放配置点 (a', v', d') 以及语义等价函数 $\varphi(\cdot, \cdot)$ 作为输入, 输出重放序列 s'. 算法乐观地假设 s 中的各事件 e 均为核心事件 (第 3 行), 并为之局部搜索辅助序列 l 和语义等价事件 e' (第 4 行). 搜索失败表示不存在任何事件 e' 使得 $\varphi(e, e') = \top$ (第 5 行), 此时算法通过启发式规则来判断乐观假设是否被违背 (即事件 e 是否为辅助事件, 第 6 行). 当判断 e 确为辅助事件时, 算法保守地丢弃 e 并继续重放过程 (第 6 行), 否则, 算法向用户报告重放失败 (第 7 行). 失败的原因可能是两个配置点对应的搜索空间语义差异巨大 (如应用在新版本采取了全新的设计, 即便人类用户也需重新学习方可使用), 也可能是技术局限导致搜索不够充分. 算法使用函数 *is-aux*$(\cdot, \cdot)$ 尝试通过启发式规则模拟函数 $\neg\eta(\cdot, \cdot)$, 一种常见的模拟方式是判断是否能在有限的后继事件中发现出现在当前界面的核心事件, 即 $\exists\, i < j \leqslant i + K.\ \exists e' \in \textit{curr-gui}(a', v', d').\ \varphi(s[j], e') = \top$, 其中 *curr-gui*$(a', v', d')$ 表示当前界面产生的所有事件集合. 搜索成功时 (即 $e' \neq \bot$), 算法将辅助序列 1 及语义等价事件 e' 添加到 s' (第 8 行), 并通过执行 $l::[e']$ 改变应用的状态 (第 9 行), 继续搜索过程. 算法最后输出重放序列 s' (第 10 行).

本文基于上述算法框架, 通过已有工作的以下 3 方面区别, 总结并分析已有工作在该框架下的具体实现.

● 组件表示与录制技术. 为了录制事件, 可能需要对系统进行一定程度的提权或修改, 或需使用特定的系统服务等. 侵入性 (intrusiveness) 反映了录制方法为进行录制需付出多大努力. 本文在第 3 节列举了流行的组件表示方法及主流录制方法, 并系统地分析了它们对系统的侵入性以及适用于何种录制重放场景和任务.

● 事件语义等价策略. 录制重放涉及对应用组件语义的理解, 我们在第 4 节分析已有工作采取了何种事件等价函数 $\varphi(\cdot, \cdot)$, 以及各等价策略对于语义理解的合理程度、适用于何种录制重放任务.

● 局部搜索策略. 探索需权衡效率和精确性: 高效的探索会牺牲部分精确性, 精确的探索亦带来高昂成本. 我们在第 5 节从精确性和搜索效率两方面对已有工作采取的局部搜索策略进行系统性分析, 并尝试解释已有工作对二者的权衡.

3 组件表示与录制技术

不同表示方法蕴含不同程度的组件和应用语义, 对录制方法提出不同的要求. 该部分的挑战在于如何定义一种合适 (且对普通用户友好) 的表示方式, 从而使得: (1) 该方式蕴含足够辅助事件语义的等价性判定和对辅助序列的局部搜索的应用语义; (2) 该方式采用的录制技术应对系统和应用产生尽可能小的依赖和侵入性修改, 从而使录制过程简单易用.

为了使表示方法蕴含更多应用语义, 已有工作根据时代特点, 针对性地提出了不同的表示方式, 并演化出基于坐标的表示、基于属性的表示、基于图像的表示和基于语言的表示这 4 种, 其采用的录制方法亦逐渐从内核 (kernel) 层向应用 (application) 层迁移. 现阶段, 已有工作以基于属性的表示为主流, 并呈现出向基于图像、语言的表示过渡的发展趋势. 本节按照该顺序对各表示方法及其主流录制技术进行介绍, 并分析该录制技术对应用和系统的侵入性, 以及表示方法适用于何种录制重放任务.

3.1 基于坐标的表示 (coordinate-based representation)

用户与应用的交互以屏幕为媒介, 组件的坐标 (coordinate) 成为表达组件语义的一种最简单直观的组件表示方法, 即, 将一个 GUI 组件 w 表示成由屏幕横纵坐标构成的二元组:

$$w = \langle x, y \rangle \text{ where } 0 \leqslant x \leqslant W \wedge 0 \leqslant y \leqslant H,$$

其中, W 和 H 分别表示设备屏幕的宽度和高度, 坐标以屏幕左上角为原点. 本文分别用 $w.x$ 和 $w.y$ 表示组件 w 的横纵坐标. 在该表示下, 一个事件被表示为:

$$e = \langle t, \langle x, y \rangle, z \rangle.$$

如图 2 中, 组件 2 位于 Pixel XL 屏幕 (350, 1644)–(658, 1983) 处, 因此点击该组件可用〈click, 〈400, 1700〉, ⊥〉表示; 〈input, 〈50, 300〉, "1"〉则表示在屏幕坐标 (50, 300) 对应的组件 (即计算器输入框) 中输入文本"1".

基于坐标的表示 (简称坐标表示) 简单直观, 清楚地表达了用户对屏幕的操作, 容易被工具理解和解析. 2017

年的研究工作显示, 微信工程师将坐标表示作为评判一个录制重放工具是否可以用于工业应用最重要的指标之一[26]. 因此坐标表示是录制重放工具的必备能力之一, 被大量已有工作[34−56]采用.

主流捕获组件坐标的方式有 4 种: 内核层捕获、框架 (framework) 层服务捕获、应用层捕获和外部捕获. 由于有些录制方法也可以用于其他组件表示, 因此本节将对其录制机制进行详细介绍, 并简略后续其他组件表示中录制方法的介绍.

(1) 内核层捕获: 第 1 种方式在硬件驱动对应用事件进行读取. 安卓沿用 Linux 内核, 继承“一切皆文件”的设计哲学. 在安卓中, 输入设备 (如屏幕、键盘、鼠标等) 被建模为驱动文件并置于输入驱动目录/dev/input/下. 对驱动文件进行读取便可获取驱动的属性和内容, 对驱动文件进行写入便可向驱动输入. 其中, 与 GUI 事件相关的是屏幕驱动文件 (如图 2 中 Pixel XL Android 8.1 设备上为/dev/input/event1). 读取屏幕文件可获知当前发生事件的屏幕坐标, 写入屏幕文件可发送点击、滑动等事件. 因驱动里获取到的是相对底层的事件 (比 AMS_MT_*等事件而非 click), 录制工具需对屏幕文件格式进行解析. 考虑到解析难度不大, 亦无需对系统进行修改, 该捕获方式成为目前最流行的坐标表示录制方法[34−35,37−43,56]. 虽该方式无需对系统和应用进行提权和修改, 但需要读取特殊的底层驱动文件, 难以被普通用户直接使用.

(2) 框架层捕获: 该方式利用安卓系统的辅助功能服务框架 (accessibility service framework). 辅助功能服务是安卓系统提供给辅助残障人士使用移动设备的服务框架[57], 该框架会将系统内的重要事件 (称为辅助事件, accessibility event, 包括 GUI 事件、GUI 界面改动等) 发送给辅助功能管理器 (accessibility manager), 由该管理器向所有已注册的辅助功能服务 (accessibility service, 如 TalkBack[58]) 进行分发. 辅助功能服务接收并处理辅助事件 (如朗读事件所对应组件的文本), 为残障人士提供帮助. 该框架允许开发者注册第三方 (third-party) 辅助功能服务[59], 录制工具[60,61]可通过注册服务捕获并处理与 GUI 事件相关的事件 (如 TYPE_VIEW_CLICKED), 从而进行坐标捕获. 录制工具可以利用该方式捕获坐标和属性信息 (见第 3.2 节). 然而, 注册第三方辅助功能服务需用户手动授权, 这将对系统产生较高安全威胁[62−64], 因此对用户和系统具有较高的要求和依赖.

(3) 应用层捕获: 该方式一般需对应用进行静态或动态插桩. 安卓事件传递从驱动开始, 在应用层结束. 在应用层, 事件从应用 GUI 界面所在的窗口 (window) 和活动 (activity) 开始, 分派 (dispatch) 给根组件 (decor view) 进行递归传递. 传递过程中, 各组件 (view group) 按序询问其子组件 (view 或 view group) 是否消费该事件, 若存在任何消费该事件的子组件, 则由该子组件进行消费并终止传递过程, 否则通知父组件事件无法消费. 传递结束仍未消费的事件会由 GUI 界面所在的活动进行消费. 该传递过程主要通过方法 dispatchTouchEvent() 进行. 该方法既存在于窗口类, 也存在于活动类和组件类. 主流的实现方案通过对该方法进行插桩, 获知 GUI 事件的坐标. 考虑到事件传递过程允许任何 view group 类型的组件对事件进行拦截, 因此也可通过 onInterceptTouchEvent() 进行插桩. 同内核层捕获类似, 因为该方法的参数 MotionEvent 是较为底层的事件, 因此需录制工具对其进行解析 (解析难度比内核层捕获小). 对应用进行插桩一般有以下两种途径.

● 静态插桩: 该插桩方式对 apk 反编译得到的中间代码 (如 smali) 进行录制代码插入[43]. 常用的反编译工具包括 apktool[65]和 androguard[66], 静态插桩工具包括 Frida[67]和 Soot[68]等. 该插桩方式仅需在应用安装前通过插桩工具进行静态代码插入, 因此运行时效率不会受插桩工具影响. 然而因为需要对应用进行预处理 (反编译、重打包和签名), 该方式难以用于工业级应用[26], 也难被普通用户接受.

● 动态插桩: 动态插桩在运行时对应用进行代码插入[54,69,70]. 因为需要动态插桩工具的运行时支持, 所以运行时效率受到插桩工具的影响. 常用的动态插桩工具包括 Frida[67], Xposed[71], VirtualXposed[72]和 JDI (Java debugging interface). 它们之间的差别在于: Frida 使用 ptrace 动态调试被插桩应用; Xposed 通过替换并修改 zygote 实现动态代码加载; VirtualXposed 在应用层创建应用虚拟执行环境; 而 JDI 是 Java 动态调试接口, 可通过调试方式运行并插桩安卓应用. 虽动态插桩无需任何预处理, 但需牺牲一定成本: Frida 和 Xposed 需要使用或修改操作系统级别的工具 (ptrace, su), 因此需对应用进行提权 (获取 root 权限); VirtualXposed 要求用户在虚拟环境中进行应用的预安装, 且难以用于工业级复杂应用 (如 Microsoft Word[73]); JDI 要求被调试应用处于调试模式 (debugging mode) 而非发布模式 (release mode).

(4) 外部捕获: 该方式一般将录制工具设计为客户/服务器 (C/S) 架构. 被录制应用在服务端环境中 (虚拟机或真机) 运行, 并通过 WebRTC、VNC 等方式将应用界面推送至客户端. 客户端记录用户的操作坐标, 并转发给服务端执行. 这种方式虽无需对应用和系统进行任何侵入性修改, 但需用户安装特殊客户端, 使用范围并不广泛[35,41,74].

除上述主流的事件录制方式, 另一种不常用的捕获方式是对安卓系统进行定制化修改, 并在系统层面对录制重放进行支持. 该方式重放成功率高, 无需对应用进行任何修改, 不仅适用于坐标表示, 亦适用于其他组件表示方式. 但需特殊的定制化系统, 对系统具有较大侵入性. 该录制方式仅有极少数工作[36]在使用.

目前, 坐标表示主要用于同应用同版本的同设备重放和极少数跨设备重放场景和任务, 如由开发者主导的缺陷重现. 区别于终端用户参与的缺陷重现, 该方式一般被应用内置的录制或追踪工具进行缺陷捕捉上报, 开发者会创建与录制配置点完全相同的配置点进行缺陷的复现、定位和修复. 基于坐标的表示尽管简单易实现, 但该表示不包含任何组件和应用程序语义, 难以用于其他复杂的录制重放场景和任务.

3.2 基于属性的表示 (property-based representation)

相比于坐标, 组件的属性 (如组件的 id、显示的文本) 蕴含更多应用语义. 该表示将组件表示为“属性-值”的集合, 即,

$$w = \{\langle k_i, v_i\rangle \mid k_i \in \Sigma\}$$

where Σ = {package,class,index,resource-id,text,content-desc,parent,activity,clickable,scrollable,...},

其中, Σ 表示常用属性集合, 这包括组件自身的属性 (如 text) 和上下文属性 (如 parent). 本文用 $w.k$ 表示组件 w 的 k 属性对应的值.

在该表示下, 图 2 中的 2 可以表示为 (限于文本长度, 我们对一些属性值进行了简化, 如 c.a.c 表示 com.android.calculator2):

{⟨package, c.a.c⟩, ⟨class, a.w.B⟩, ⟨content-desc, 2⟩, ⟨resource-id, digit_2⟩, ⟨text, 2⟩, ⟨clickable, true⟩, … }.

而组件 2 可以表示为

{⟨package, c.a.c⟩, ⟨class, a.w.B⟩, ⟨content-desc, 2⟩, ⟨resource-id, number_2⟩, ⟨text, 2⟩, ⟨clickable, true⟩, … }.

可以看到, 在该表示下, 两个组件的 package、class、text 等属性相同, 而 resource-id 不同.

基于属性的表示 (简称属性表示) 包含了组件和应用的部分语义信息, 具有良好的跨设备、版本和应用性. 属性表示是成为目前最常用的表示方法[23−25,43−52,54,60,69−92]. 微信团队的开发者亦将属性表示作为评判录制重放技术的重要指标[26].

该表示的录制方法需检视和爬取应用的 GUI 界面, 从而得到组件信息.

一种录制方式是结合坐标表示录制方法和 UI Automator[23−25,70,88−93]. UI Automator 是安卓官方提供的搭建在辅助功能服务框架上的工具, 用于爬取界面及其组件信息. UI Automator 维护了组件的树状关系和组件的属性值. 应用启动时会在内部启动名为 AccessibilityInteractionController 的 binder 服务[94]. 该服务作为服务端向辅助功能管理器提供应用当前界面的基本信息. 当 UI Automator 被开发者调用时, UI Automator 以辅助功能管理器为媒介, 从 AccessibilityInteractionController 服务处获得 GUI 界面信息. 录制工具先通过 UI Automator 获取当前界面的基本信息, 当 GUI 事件来临时, 利用坐标表示的录制方法得到坐标信息, 并查询对应坐标处的组件及其属性. 该录制方法尽管实现简单, 但无法预测 GUI 事件的来临时机, 录制工具需不断轮询 UI Automator 并时刻保持最新的 GUI 界面. 由于每次轮询都涉及至少两次进程间通信, 所以该方式的开销较大. 同时, 一旦轮询间隙选择失当, 会导致组件信息分析出错. 因此该录制方式仅极少数工作[15]使用.

一种改良的录制方式利用了同应用同版本的同设备重放 (称为自重放, self-replay)[43]. 该改良首先利用坐标表示录制完整事件序列, 之后通过同应用同版本的同设备重放, 在每个事件重放之前利用 UI Automator 获取界面的基本信息, 并通过坐标找到对应的组件及其属性, 从而完成坐标到属性表示的转换. 然而, 受限于同应用同版本的同设备重放能力, 该方式通常难以处理滑动事件.

上述两种方式依赖安卓提供的工具, 对系统有一定的侵入性.

第 3.1 节中提到的框架层和应用层录制方法亦可用于录制属性表示.

● 辅助功能服务提供的辅助事件如 TYPE_VIEW_CLICKED 以组件及属性方式提供 (坐标是组件的属性之一), 因此该录制方式可以直接复用[60].

● 使用 dispatchTouchEvent() 等应用程序接口进行插桩时, 开发者可以从当前活动的根组件 (activity.window.decorView) 进行树遍历, 利用坐标得到对应组件及其属性信息[54,59,70].

本节中提到的录制方式均对系统有一定程度的依赖或者侵入, 但该表示方式具有良好的跨应用、版本和设备的特性, 可以用于大多录制重放的场景和任务, 如缺陷重现、兼容性测试、回归测试和测试迁移, 是当前最流行的表示方法. 然而, 该表示依赖于开发者标注的组件属性, 因此一旦开发者漏标或者标注错误, 则容易导致重放失败. 需要说明的是, 2020 年的一份研究工作显示, 在 Google Play Store 的应用中, 有近 45% 的 GUI 界面存在至少一个 ImageButton 缺少此类标注信息[95], 导致该表示对 ImageButton 类组件的重放难以起效. 基于图像的表示是对该问题的一种有效的补救手段.

3.3 基于图像的表示 (image-based representation)

基于图像的表示 (简称图像表示) 将组件表示为由像素 (pixel) 构成的图像, 如2在这种表示下, 一个组件被表示为像素数组, 即:

$$w = [px_1, px_2, ..., px_n]$$

其中, $px_i = \langle r_i, g_i, b_i, a_i \rangle$ 表示该像素对应的红、绿、蓝和透明度通道的值. 本文用 $w[i]$ 表示组件 w 第 i 个像素的值, 并定义像素相等为各通道值均相等. 如图 2 中, 组件2可被表示为 [〈67, 67, 67, 255〉, …, 〈255, 255, 255, 255〉,…, 〈67, 67, 67, 255〉], 其背景像素为〈67, 67, 67, 255〉, 文字像素为〈255, 255, 255, 255〉.

相比于坐标和属性表示, 该表示方式容易被终端用户理解, 且具有较好的鲁棒性 (不受坐标和开发者标注影响).

主流的图像表示复用属性表示的录制方式: 在获得组件属性的时刻对 GUI 界面进行截取, 利用组件坐标和宽高属性对截图进行裁剪从而得到图像信息[52,55,87,96−98]. 因为复用属性表示的录制方式, 因此对系统有一定的依赖和侵入, 且难以被终端用户使用.

视频录制[53,99]亦常用于图像表示的录制. 该方式借用系统内部 (如内置录屏工具) 或外部 (如摄像机) 工具录制用户的操作, 通过图像处理 (image processing) 或计算机视觉 (computer vison) 的方式对视频进行分析, 获取各事件类型、组件信息及用户输入. 该方式因不借助任何安卓系统的特有能力 (由于录屏是操作系统中的一个普遍需求, 因此我们此处不将录屏当作操作系统独特的能力), 因此对系统具有零侵入, 并方便中终端用户使用. 然而, 限于目前视频分析的能力, 该类工具对用户录制视频的行为具有一定的要求. 如, V2S[99]要求用户在设置中打开“显示点按操作反馈 (show taps)”; RoScript[53]要求用户在录制时按照固定的位置放置设备和固定的步骤录制视频. 这类技术通常还需利用界面分割 (GUI segmentation) 和物体检测 (object detection)[100−102]划分界面、识别组件, 从而获取组件像素.

图像表示是特定重放任务中对终端用户最友好的录制方式之一, 如缺陷重现. 在该场景下, 终端用户仅需使用设备内置的录屏功能录制发生缺陷的步骤, 并连同应用、版本和设备信息提交给开发者, 即可完成缺陷上报. 图像表示同样适用于其他跨配置点的场景和任务. 然而, 限于目前视频分析和处理的能力, 该表示方法还未在录制重放任务中得到广泛使用.

3.4 基于语言的表示 (language-based representation)

基于语言的表示 (简称语言表示) 采用接近自然语言的中间语言或直接使用自然语言对组件进行描述, 图 2 中, 组件2可被表示为“button with text 2”(中间语言, 其中 with text 是关键词)、“the 2 button”(自然语言) 等, 即

$$w = \mathrm{l} \text{ where } \mathrm{l} \in \Gamma,$$

其中, Γ 表示该 (中间/自然) 语言所定义的句子集合.

该表示方式本质上是对属性表示的改进: 使表示趋向自然语言, 降低用户学习成本. 用户在录制时仅根据语法对事件序列进行组织, 对系统具有零侵入性. 但受限于目前自然语言进行理解 (natural language understanding) 和分析 (natural language processing) 的能力, 该表示方式难于从自然语言中提取事件序列. 目前采用自然语言表示的

工作[32,103,104,61]主要借助词性标注 (part of speech tagging) 和依存分析 (dependency parsing) 解析词语之间的依赖关系, 并提取关键的动作 (如谓语动词 “click”) 和短语 (如名词短语、宾语“2 button”) 作为事件类型和组件的部分属性 (如 text). 但此类分析难以理解存在关联关系的语句, 面临识别和提取精度的问题: 如现有工作难以将句子“click the 2 button, the + button, the 3 button, then end”中的谓语动词“end”与 click 事件类型联系起来. 因此目前仍以中间语言表示[105]为主.

语言表示具有较低的侵入性和学习成本, 易于终端用户使用, 适合缺陷重现的任务. 但难以用于终端用户介入少的录制重放场景, 如兼容性测试和测试迁移.

需要说明的一点是, 尽管语言表示将组件表示为句子的形式, 但组件信息并非以语言形式组织, 因此语言表示和属性表示通常在重放前或重放时进行相互转换[24,25,32,88].

4 事件语义等价策略

语义等价函数 $\varphi(e_1, e_2)$ 用于判断位于两个的配置空间中的事件 e_1 和 e_2 是否语义等价, 录制重放技术利用该函数在重放配置空间中为核心事件搜索辅助序列和语义等价事件. 鉴于不同配置点对于同一语义采用不同的表达形式, 如采用不同的文字、图像等表达同一概念, 因此该部分的关键技术挑战在于如何根据组件表示方式进行正确的组件语义理解. 如在图 2 中, 事件〈click,  , ⊥〉和〈click,  , ⊥〉在组件属性 (前者 resource-id 为 digit_2, 而后者为 number_2) 和图像 (前者为  , 而后者为 ) 上均不一致, 但语义等价函数 $\varphi(\cdot, \cdot)$ 需能正确判断二者表达同一语义 (即在计算器中输入“数字 2”), 这为语义等价函数的涉及带来极大挑战.

根据组件表示的不同, 语义等价包括同类组件表示的语义等价和异类组件表示的语义等价. 考虑到已有工作在录制和重放时采用同种组件表示形式, 因此本节假设 $e_1.w$ 和 $e_2.w$ 以同种组件表示形式表示. 一般情况下, 对核心事件的重放亦使用与录制时相同的事件类型, 因此本节假设 e_1 和 e_2 事件类型相同, 即 $e_1.t$=$e_2.t$. 事实上, 存在两个事件类型不同, 但仍然语义等价的情况, 比如“点击返回按钮”和“在屏幕边缘自右往左滑动”在语义上都等价于返回. 本文遵照现有工作的基本假设, 因此不考虑此种情况. 随着组件表示的发展, 已有工作采用的语义等价策略对应用语义理解的深度逐渐增加, 并逐渐演化出完全等价、基于属性的等价、基于文本的等价和基于图像的等价 4 种.

4.1 完全等价 (exact equivalence)

完全等价策略 (简称完全等价) 要求表示组件 w 的各分量相等:

- 坐标表示, $\varphi(e_1, e_2) = \top \Leftrightarrow (e_1.x = e_2.x) \wedge (e_1.y = e_2.y)$, 即要求坐标分量完全相同.
- 属性表示, $\varphi(e_1, e_2) = \top \Leftrightarrow \forall\, k \in \Sigma.(e_1.w.k = e_2.w.k)$, 即要求各属性值均相同.
- 图像表示, $\varphi(e_1, e_2) = \top \Leftrightarrow (|e_1.w| = |e_2.w|) \wedge \forall 1 \leqslant i \leqslant |e_1.w|.(e_1.w[i] = e_2.w[i])$, 即要求二者具有相同的像素数目, 且各像素均相等.i
- 语言表示, $\varphi(e_1, e_2) = \top \Leftrightarrow e_1.w = e_2.w$, 即要求 $e_1.w$ 和 $e_2.w$ 表示同一个句子.

完全等价对因配置点变化而导致的局部变化不敏感 (locality-insensitive), 任何细微改动都将导致完全等价策略失效. 图 2 中, 在完全等价策略下, 坐标表示 (位于屏幕不同坐标位置)、属性表示 (resource-id 属性不同) 和图像表示 (颜色不同) 下的组件  和  均不等价, 但若以相同的句子 (如“button with text 2”) 表示, 二者则等价. 因为录制重放技术无法假设不同配置点应用界面的一致性 (如不同应用针对同一语义采用不同的近义词表示、同一应用在跨设备时根据设备屏幕大小进行图像分辨率调整等), 因此完全等价策略被广泛应用于同配置点的录制重放[34−42,56,75,80,92,99], 却几乎无法用于跨配置点的录制重放[90,91].

跨配置点录制重放需对局部变化敏感的等价策略 (locality-sensitive semantic equivalence), 其他 3 种等价策略均属于此范畴.

4.2 基于属性的等价 (property-based semantic equivalence)

基于属性的等价 (简称属性等价) 是用于属性表示的一种等价策略. 该策略基于以下发现: 开发者在开发应用

时, 会将组件的属性分为对配置点敏感的属性 (Σ^-) 和对配置点不敏感的属性 (Σ^+), 其中前者在跨配置点时经常变化, 如 text 属性和 bound 属性, 而后者几乎不变, 如 resource-id 和 content-desc 属性. 因此该策略从对配置点不敏感的属性中选择全部或部分属性用于等价性判定, 即:

$$\varphi(e_1, e_2) = \top \Leftrightarrow \forall\, k \in \Sigma'.(e_1.w.k = e_2.w.k) \text{ where } \Sigma' \subseteq \Sigma^+.$$

在图 2 中, 一种合适的 Σ'可为 $\Sigma' = \{\text{content-desc}\}$, 在这种选择下, 组件2和2等价.

该等价策略实现简单且等价性判定高效, 被广泛用于同应用同版本的跨设备重放和同应用同设备的跨版本重放场景[43−54,60,61,69,70,74−79,81−87,104,105], 是目前使用最广泛的语义等价策略. 但该策略存在两个弊端: (1) 依赖属性表示, 继承了属性表示的缺陷, 即需要开发者正确标注属性, 而不能漏标或标错; (2) 因不同应用针对同一语义采用不同的属性标注 (如使用近义词), 因此几乎无法用于跨应用的录制重放.

完全等价和基于属性的等价策略的策略本身仅从组件表示的分量出发, 以数学方式进行等价性判定, 尽管具有合理性, 但适用范围局限, 难以用于跨任意配置点的录制重放. 跨任意配置点的录制重放倾向于以人类的角度 (human-like) 思考应用的语义和语义等价性, 这一般需要自然语言理解、自然语言处理、图像处理和计算机视觉的辅助.

4.3 基于文本的等价 (text-based semantic equivalence)

基于文本的等价 (简称文本等价) 是适用于属性表示和文本表示的一种等价策略. 该等价策略将对应表示的组件转换为易于使用自然语言理解和处理技术的表达方式, 而后用对应领域的技术来判定等价性. 这一般包含以下几个步骤.

(1) 将组件从对应表示方式转换为文档 (document): 属性表示需要这一步骤, 而语言表示一般不需要. 对于属性表示, 这一步骤通常提取组件每个字符串属性的值, 并以特殊符号 [*T*](如分号) 进行连接得到, 若以 d_w 表示组件 w 产生的文档, $\Sigma^{\text{str}} \subseteq \Sigma$ 表示所有字符串属性的集合, 那么:

$$d_w = v_1 :: [T] :: v_2 :: [T] :: \ldots :: v_n \text{ where } \forall k_i \in \Sigma^{\text{str}}.v_i = w.k_i.$$

在图 2 中, 组件2的文档表示可为“com.android.calculator; android.widget.Button;2; digit_2;2; true”.

(2) 对文档进行预处理: 这是大多数自然语言处理任务的必要步骤之一, 主要包括断句、停用词消除 (stop word elimination) 和词干提取和词形还原 (stemming and lemmatization) 等: 该步主要对分析无意的词语 (如 of) 进行消除, 并将词语转换为其词干表示. 在安卓里, 此类预处理一般还会根据安卓和应用的语义对一些特殊词语进行消除和转换 (比如带有应用独特语义的词语、名称、标语等), 这些词因具有特殊的含义, 在一个由自然语言构造的语料库中, 容易和词语本身的含义发生歧义而产生误导. 如, 2的文档表示经预处理后为“com android calculator android widget Button 2 digit 2 true”.

(3) 将预处理后的文档向量化 (vectorization): 向量化也是自然语言处理的必要步骤之一, 用于将文档转换为方便计算的向量表示. 常用的词向量化方式包括:

a) TF-IDF[106]: TF-IDF 是一个用于表示一个词语对其所在文档重要程度的统计量. 通过对文档的重要程度, 这种向量表示可以标注出对每个组件最重要的词语作为组件的主要语义. 计算向量中的每个词语 TF-IDF 值并将之替换. 该表示虽然具有比较快的计算效率, 但难以在近义词上起效.

b) 词嵌入 (word-embedding)[107]: 词嵌入构造的向量空间中, 距离相近的单词在含义上具有相似意义. 这种表示更接近人类对程序语义的理解, 因而比较适合跨配置点的录制重放. 常用的词嵌入包括 Word2Vec[108]和 Glove[109]. 词嵌入将得到每个词对应的向量, 而后将得到的词向量进行聚合 (如连接为更高维的向量、进行求和或取平均等) 即可得到文档向量. 该向量表示的问题在于, 同一个词语在词嵌入向量空间中被表示为同一向量, 因此难以区分一词多义.

c) WordNet[110]: WordNet 是一个人工构建的语义网络. 该网络将人类认知中含义相近的词语划分为近义词组 (synsets), 并通过近义词组之间的语义关联将词组进行连接. 此种向量化方式将文档中的各个单词表示为其在 WordNet 中的词组. 因为 WordNet 中的每个单词可以存在于多个词组, 因此可以区分近义词和一词多义. 在向量化时, 为了使用更准确的词组表示, 通常需要对表示组件的某些属性 (如 text) 进行依存分析, 并提取其中关键词语

的词性.

(4) 利用向量距离进行计算, 并作为等价性结果. 由于文档向量捕获了组件的语义, 并将语义的差异表达为向量空间中的距离, 因此该步骤通过文档向量的距离来判断语义的相似度. 常用的向量距离包括 cosine 距离、WUP 距离和 WMD 距离. 其中, cosine 距离用向量的夹角描述其语义相似性; WUP 距离用于计算 WordNet 中两个词组的距离; WMD 距离通过文档中每个词在向量空间中移动到另一个文档中词语的最小距离之和来捕获语义差异, 这一般需要在第 3 步使用连接为更高维的方式保留词向量.

若以 u_w 表示组件 w 的向量表示, 那么:

$$\varphi(e_1, e_2) = \top \Leftrightarrow distance(u_{e1.w}, u_{e2.w}) < THRESHOLD,$$

其中, 函数 $distance(\cdot, \cdot)$ 用于计算两个向量的距离, 标量 $THRESHOLD$ 表示距离的阈值.

基于文本的等价[23−25,32,33,88−91,97,103]以更趋向于人类的等价方式捕获组件的语义, 更适用于跨配置点的录制重放场景和任务. 具体而言, 步骤 (1)–(4) 中描述的向量表示和距离均适用于同应用的跨版本或者跨设备重放, 但 TF-IDF 形式的向量化并不适用于跨应用的录制重放, 因为应用之间通常使用近义词来表达同一语义.

4.4 基于图像的等价 (image-based semantic equivalence)

基于图像的等价 (简称图像等价) 是适用于图像表示的一种等价方式. 该等价策略一般用于比较两张图片是否在语义上相似:

$$\varphi(e_1, e_2) = \top \Leftrightarrow semasim(e_1.w, e_2.w),$$

其中, 函数 $semasim(\cdot, \cdot)$ 用于根据两个组件图像的像素判断其是否语义等价.

第 1 种常用的图像等价函数是模板匹配 (template matching)[53,96]. 该策略使用 $e_1.w$ 作为模板, 在 (a', v', d') 的当前 GUI 界面截图中寻找与模板在像素上相似的部分, 该寻找容忍因遮挡、非刚性变换、照明、背景变化和尺寸变化等引起的像素差异, 而后判断其是否为 $e_2.w$, 如利用相似部分与 $e_2.w$ 的 IoU (intersection over union) 值. 如, RoScript[53]首先解析所录制视频中手指所在处的组件图像, 而后将该图像在当前界面进行模板搜索从而找到重放组件. 若在图 2 的 Pixel 3 XL 设备上重放在 Pixel XL 设备录制的〈click, 2, ⊥〉事件, 则需以 [〈67, 67, 67, 255〉,…, 〈255, 255, 255, 255〉,…, 〈67, 67, 67, 255〉] 为模板, 在设备 Pixel 3 XL 当前屏幕上寻找与之相似的像素片段.

第 2 种常用的图像等价函数是图像特征描述符 (image feature descriptor). 图像特征描述符通过对图像中的关键点描述图像的基本性质, 如颜色、形状、文本等. 图像特征描述符经常用于物体检测等计算机视觉任务中. 在这里, 通过对 $e_1.w$ 和 $e_2.w$ 进行特征提取, 得到图像中的关键点, 而后利用最近邻匹配等匹配算法检测两个特征是否等价. 常用的特征描述符包括 HOG[111]和 SIFT[112]. 如, LIRAT[55]和 Meter[98]对界面上所有组件计算其 SIFT 特征, 并通过 FLANN[113]等近邻匹配算法进行相似度计算. 图 4 展示了组件 2 和 2 的 SIFT 特征点及特征点间的 FLANN 匹配结果.

第 3 种常用的图像等价函数是图像哈希 (image hashing). 图像哈希是一种局部敏感哈希 (locality-sensitive hashing). 通过允许图像局部的像素变化, 图像哈希可以将相似的图像映射为相近的哈希, 而后利用汉明距离等判断是否等价. 如 2 和 2 的 pHash[114]分别为 1152996272823620864 和 12393900676948688591. 正如该例所示, 图像哈希更适用于同应用的跨版本、设备重放, 而不适用于跨同类应用的重放.

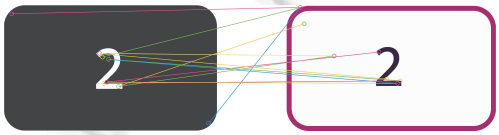

图 4 2 和 2 的 SIFT 特征点以及它们之间的 FLANN 匹配

基于图像的等价以趋近于人类的理解方式捕获组件语义, 因此亦适合于跨配置点的录制重放任务. 但受限于图像处理和计算机视觉等算法的能力, 以及移动设备中的图像与自然图像的差异[102], 目前图像等价在跨应用录制重放场景的效果还不尽如人意.

5 局部搜索策略

在算法 1 中, 局部搜索为核心事件 e 在配置点 p'的局部空间中搜索辅助序列 l 和语义等价事件 e'. 该局部空间是搜索空间 $S_{p'}$的子空间. l 和 e'需满足以下关系:

$$\varphi(e, e')\wedge((l=\emptyset)\vee((e''\rightarrow^{S_{p'}}l[1])\wedge(l[|l|]\rightarrow^{S_{p'}}e'))),$$

其中, e''表示事件 e 在 $H(s)$ 中的前序事件 $pred(e, H(s))$ 在 $S_{p'}$中的语义等价事件, 即

$$\varphi(pred(e, H(s)), e'')=\top.$$

局部搜索面临的挑战来源于巨大的搜索空间: 移动应用通常具有大量组件, 即便为单一核心事件进行辅助序列的搜索也可能产生成千上万条潜在事件序列, 这使得实现精确的局部搜索通常耗费大量时间. 为实现可用的录制重放技术, 局部搜索策略需对搜索效率和搜索精度进行合理取舍, 从而演化出零搜索、暴力搜索和基于模型的搜索 3 种搜索策略. 鉴于对效率和精度具有不同取舍, 这 3 种策略适用于不同录制重放场景和任务. 本节按照该顺序介绍每种搜索策略, 分析总结现有工作, 并给出每种搜索策略适用的录制重放场景和任务.

5.1 零搜索 (none local search)

在同应用同版本的同设备重放和部分同应用同版本的跨设备重放场景中, 移动应用的界面及界面中的组件不会增加、删除或者修改. 该情况下, 录制和重放配置点具有一致的搜索空间, 仅使用语义等价函数便可找到对应空间中满足评判标准的序列, 因而整个过程无需任何局部搜索过程.

据本文所知, RERAN[34]是该方面发表的第一份工作. RERAN 采取坐标表示的形式并在驱动层进行事件捕获和录制. RERAN 发表于 2013 年, 彼时应用简单, 因此在重放阶段, RERAN 合理地假设移动应用的界面及其组件不会发生增加、删除和修改的情况, 使用完全等价在坐标表示下的语义等价函数依次重放事件序列中的各个事件, 实现对界面 GUI 事件重放. 时至今日, RERAN 仍是同应用同版本同设备录制重放场景下的代表性工作 (state-of-the-art)[26].

但随着移动应用愈加复杂, 该假设在同应用跨设备场景下不再成立 (如图 2). 为处理界面发生的细微变化, SARA[43]和 RANDR[69,70]进行了一些特殊处理. 这两份工作采用基于属性的组件表示和基于属性的语义等价策略. 当语义等价函数对当前界面的每个组件都返回$\top$时, SARA 和 RANDR 会检查待重放事件的组件属性, 若发现其存在形如 ListView 等可滑动的父组件, 则会尝试发送滑动事件, 并时刻检查是否有语义等价的事件出现. 这两份工作虽然考虑了界面发生变化的情况, 但其仅能处理可滑动组件且没有其他任何搜索操作, 因此本文仍将其归类为零搜索.

在不违背零搜索策略假设的情况下, 零搜索容易实现、效率高且搜索精确. 零策略适用于同应用同版本的同设备重放和部分跨设备重放, 如开发者主导的缺陷重现. 在该场景和任务下, 零搜索策略[34−56,60,61,69,70,74−83,92,96,97,99,103−105,92]基于界面不变假设能够实现高效和精确的重放. 然而随着应用复杂度的上升, 该假设将难以被维系. 此外, 零搜索亦无法用于跨配置点的录制重放.

5.2 暴力搜索 (brute-force-based local search)

暴力搜索是最朴素的搜索方式: 从搜索空间的当前位置 (e'') 开始, 该策略尝试每个可以发送的事件, 并采用深度优先 (depth-first) 或广度优先 (breadth-first) 的方式进行探索; 当探索到尽头时, 便进行回溯.

若有足够时间, 暴力搜索策略能够保证找到满足评判标准的事件序列. 然而, 移动应用能够产生海量的事件序列, 即使局部空间, 也无法在有限时间内穷尽探索. 低效率导致暴力搜索难以在真实环境中应用. 因此现有使用暴力搜索的工作对暴力搜索进行了优化.

第 1 种优化方式[87,98]基于这样的假设: 移动应用为人类设计并服务, 辅助序列因为不包含任何应用语义, 所以应该足够短从而方便人类使用. 所以该优化方式通过限制探索深度来加速暴力搜索过程. 该优化在局部搜索时, 每次仅探索有限个事件, 若在有限个事件内发现语义等价事件, 则重放下一个事件; 否则停止在当前分支的探索并及时回溯. 尽管移动应用能够产生海量的事件序列, 但每个 GUI 界面仅包含为数不多的组件, 因此通过限制探索深

度能极大地减小搜索空间, 提升搜索效率.

第 2 种优化方式是并行化探索[103]. 在搜索空间的每个节点, 该优化方式通过复制 (fork) 自己的执行从而在子节点上进行并行化分路探索. 由于搜索空间巨大, 在各个节点均复制执行将产生巨大的开销, 因此该优化方式一般会定义特定的复制策略以决定并行化的数目. 在该种优化方式下, 如果任何一条分路执行发现了满足条件的辅助序列和语义等价事件, 则停止所有分路的执行并返回. 这种策略的效果一般取决于复制策略, 一个合理的策略将极大地加速搜索过程, 然而不合理的策略诚然有些许加速, 但会浪费大量 CPU 时间执行无意义的探索.

暴力搜索的一种变体是演化算法 (evolution algorithm)[89]. 移动应用搜索空间的特性——事件具有先后关系, 致使任何违背该特性的事件序列都无法被应用执行. 然而事件先后关系无法预先得知, 因此演化算法的两个重要步骤事件序列的交叉 (crossover) 和变异 (mutation) 经常产生违背事件先后关系的序列[8,10], 以致对搜索产生负影响.

给定足够的探索时间, 暴力搜索能够保证找到满足标准的序列, 因此暴力搜索理论上适合图 1 中任何一种录制重放场景和任务. 然而在真实情况下 (如敏捷开发), 足够的探索时间难以被满足, 所以鲜有暴力搜索在真实场景中得到应用.

5.3 基于模型的搜索 (model-based local search)

前述局部搜索策略基于一个基本假设: 应用的执行模型无法得到, 因此均采取重放时在线 (online) 探索的方式. 基于模型的搜索 (简称模型搜索) 在重放前或重放时通过静态或动态分析的方式获取一个粗略的应用执行模型, 并在该模型的指导下进行搜索[23−25,84−86,88,90,91,98].

应用的执行模型 (execution model) 通常被表示为一个三元组 $\mathcal{M}=\langle \mathcal{K}, E, \mathcal{T}\rangle$:

- $\mathcal{K}$ 表示应用的状态集合. 各工作对应用状态采取了不同程度的抽象[9,11,115].
- E 表示应用产生的事件集合.
- $\mathcal{T}:\mathcal{K}\times E\times\mathcal{K}$ 表示状态转移 (state transition) 集合. 各状态转移 $\langle K, e, K\rangle \in \mathcal{T}$ 包含源状态 $K\in\mathcal{K}$, 触发事件 $e\in E$ 和目标状态 $K'\in\mathcal{K}$, 表示在状态 K 发送事件 e 将使状态变化为 K'.

在该执行模型下, 首尾相接的状态转移序列 (state transition sequence):

$$[\langle K_1,e_1,K_2\rangle,\langle K_2,e_2,K_3\rangle,\ldots,\langle K_n,e_n,K_{n+1}\rangle]$$

对应的事件将构成一条合法的事件序列:

$$[e_1, e_2,..., e_n].$$

图 5 展示了设备 Pixel XL 上 Google Calculator 的执行模型: 该模型借鉴已有工作[9,11]将状态表示为当前界面上的组件集合, 产生 K_1 和 K_2 两个状态; 点击扩展组件 (绿色条块) 将由状态 K_1 转移为 K_2, 在 K_2 上返回将转移到 K_1, 操作数字和简单操作符 (如+) 组件将保留在 K_1 状态, 操作高级操作符 (如 sin) 组件将保留在 K_2 状态. 图 5 同时展示了在该执行模型上计算“2π”的状态转移序列.

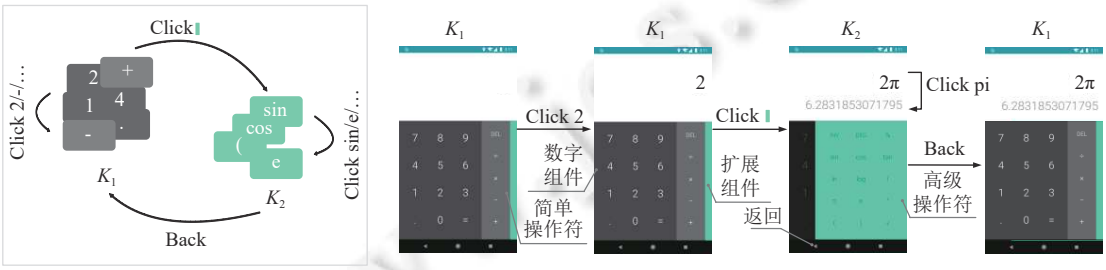

图 5 设备 Pixel XL 上 Google Calculator 的执行模型及产生“2π”的状态转移序列

CraftDroid[23]和 ATM[24,91]在执行重放前通过静态分析获取粗略的移动应用执行模型. 在为核心事件进行局部搜索时, CraftDroid 和 ATM 将询问模型包含该核心事件对应组件的状态 (可能不止一个). 二者采用基于文本的语义等价策略在模型中查找所有目的状态, 并返回所有从当前状态到目的状态的状态转移序列. 对于返回的状态序

列, CraftDroid 和 ATM 从最短序列开始尝试. 具体而言, 该模型因通过静态分析得到, 与运行时执行模型存在差异, 因此二者仅在最短序列的指导下前进一步: 二者在当前界面上利用语义等价函数寻找最短序列首事件的等价事件, 并触发该事件, 而后重复局部搜索过程. 若在最短序列对应的有限步骤内仍无法找到语义等价事件, 二者回溯并将尝试次短路径, 直到搜索到待语义等价事件.

Meter[98]在回归测试时利用录制重放技术修复一个在旧版本失效的测试用例集合. 由于静态得到的模型不够完备, Meter 采用在搜索过程中逐步建立并优化的方式得到执行模型. 具体而言, Meter 在重放初始阶段 (即前续测试用例) 采取无模型限定深度的暴力搜索方式, 并在搜索过程中根据重放的反馈不断建立和优化应用的执行模型; 在后续测试用例的重放修复中, Meter 优先采用模型搜索, 当模型搜索失败时退回暴力搜索并继续优化执行模型, 从而指导更后续测试用例的重放工作.

模型的指导大大减小了搜索空间, 并明显提高了搜索效率和准确率. 基于模型的搜索对应用进行了先验观察和建模, 因而对应用的语义有更深层的理解, 通过与属性表示和文本等价策略相结合, 模型搜索有效的指导了跨配置点的录制重放工作, 因而成为近期回归测试脚本修复、测试迁移等任务的主流实现方案. 然而, 该搜索策略强依赖于模型包含的应用先验, 因此对应用的不充分建模将影响该策略的效果. 同时, 目前主流的建模方案难以对界面复杂的应用 (Microsoft Word) 进行建模, 因此该方案还停留在简单应用类别 (计算器类、浏览器类等).

6 已有工作的讨论

表 1 对已有工作根据本文所提框架进行了分类总结. 表中部分工作采用了多种组件表示和录制技术、事件语义等价策略和局部搜索策略, 表中仅对所列工作采用的主要技术手段进行列举. 从技术组合的视角出发, 本文发现, 迄今为止零搜索、属性表示和属性等价的技术组合及零搜索、坐标表示和完全等价的技术组合占比最高, 且其中超过 70% 的工作发表于 2018 年前, 聚焦于同应用同版本的同设备重放和同应用同版本的跨设备重放; 占比第三的技术组合是模型搜索、属性表示和文本等价, 均发表于 2018 年及以后, 专注于同应用同版本的跨设备重放和跨同类应用的重放. 造成此类现象的原因是单体应用愈加复杂[43,69,70]、应用版本迭代加快[98]和同类应用的数量激增[23,24,91]. 从技术演变的视角观察, 录制重放技术经历了坐标表示到图像表示, 完全等价到图像等价, 和零搜索到模型搜索的演变. 应用复杂化和设备碎片化是造成该演变的直接原因.

安卓发展初期, 应用简单、更新迭代缓慢且设备稀少, 已有工作依据时代的特点对录制重放进行合理假设, 并实现了一些基于坐标表示、完全等价和零搜索的简单高效的工具[34−36,39,60,54]. RERAN[34]作为彼时的代表性工作最先被提出. 录制时, RERAN 通过读取屏幕设备驱动文件 (/dev/input/event*) 在内核层捕获用户输入事件的坐标信息, 并将之作为组件表示. 鉴于应用简单, RERAN 假设重放时应用与录制时完全相同, 因此将录制的坐标信息直接写入屏幕驱动文件 (完全等价) 从而完成重放. 在此过程中, RERAN 因未理解除坐标外的应用语义, 故未进行任何搜索 (零搜索).

随着安卓应用愈加复杂、设备碎片化逐渐严重, 早期假设和技术组合 (坐标表示+完全等价+零搜索) 难于应对跨设备的录制重放, 如同版本的同一应用在不同设备上尽管长相相似, 但同一组件所处坐标具有差异. 为此, 属性表示和属性等价的策略[43,69,70]出现并渐成主流. SARA[43]和 RANDR[70]是代表性工作. 在录制技术上, SARA 采用自重放进行属性表示的录制, 而 RANDR 采用了插桩的方式. 重放时, 二者亦假设应用界面的组件不会增加和删除, 因此采用属性等价在当前界面寻找等价事件 (零搜索). 这种属性表示、属性等价和零搜索的组合模式, 被众多已有工作证明了其在大多数同应用同版本的跨设备重放上的有效性[43−52,54,60,69,70,74−79,81−83]. 值得说明的是, 尽管早期工具已不再适用于跨设备的录制重放, 已有研究发现, RERAN 在同配置点的录制重放中仍难以被取代[26]. 这预示坐标表示、完全等价和零搜索的技术组合仍是同配置点录制重放的最佳组合模式. 考虑到 RERAN 对系统具有较强侵入性 (需在内核层操作屏幕驱动), V2S[99]采和 RoScript[53]采用视频和视频解析的方式对坐标进行录制, 然而二者的性能仍受限于现下机器学习和深度学习的能力.

近期的研究发现, 开发者常常会漏标或者错标组件[95], 导致前述技术组合失效. 为了应对这些问题, 文本等价、图像表示和图像等价被提出和使用[23,24,55]. LIRAT[55]是近期提出的代表性工作, 采用了图像表示、图像等价和零

搜索的技术组合. 具体而言, LIRAT 对录制的组件图像和当前界面的组件图像进行 SIFT 特征计算, 并利用 FLANN 算法[113]进行最近邻匹配从而得到等价组件. LIRAT 通过图像的方式屏蔽了因开发者失误导致的重放失败, 然而此种技术组合的性能仍然受限于图像处理的能力, 并仍需更多工作进行验证.

为充分利用屏幕空间, 开发者逐渐使用响应式模式设计和开发应用, 在这类应用里, 被重放组件将不存在于当前界面. 跨同类应用的重放亦具有相似的特点, 这使得零搜索难以应对此类场景, 亦催生了局部搜索, 尤其是模型搜索的发展. CraftDroid[23]和 ATM[24,91]采用并初步证实了属性表示、文本等价和模型搜索这一技术组合在该场景下的能力. 然而受限于构建执行模型的能力, 二者仅在交互简单的应用 (如计算器类应用、浏览器类应用等) 上进行了实验评估. 如何为工业级复杂应用 (如 Microsoft 家族应用) 快速建立准确的执行模型仍是亟待解决的问题. 此外, 此类已有工作一般使用数据驱动的方式屏蔽因文本差异带来的等价性判定问题, 因此依赖于当下自然语言处理和理解技术的能力.

表 1 基于 GUI 事件的安卓应用录制重放论文分类

搜索策略 (§5)	组件表示 (§3)	等价策略 (§4)	主要工作	代表性工具	总结
零搜索	坐标表示	完全等价	[34−42,56]	RERAN[32]	组件表示与录制技术: 它们在对系统侵入性和用户要求上有所区别. 理想的技术对系统零侵入、对用户零要求. 图像表示是最理想的表示方式. 语义等价策略: 用于判断两个事件是否语义等价. 理想的语义等价策略以人类的视角基于文本和图像进行等价性判定. 局部搜索策略: 用于为某一事件局部搜索等价的事件序列. 此类策略之间在准确性与搜索效率上取舍有异. 当前以零搜索为主流, 以模型搜索为发展趋势.
	属性表示	完全等价	[92]	SARA[43], RANDR[69,70]	
		属性等价	[43−52,54,60,69−70,74−79,81−83]		
	语言表示	属性等价	[61,104−105]	AppFlow[105]	
		文本等价	[103]		
	图像表示	完全等价	[99]	LIRAT[55]	
		属性等价	[52]		
		文本等价	[97]		
		图像等价	[53,55,96]		
暴力搜索	属性表示	属性等价	[87]	GUIDER[87]	
		文本等价	[89]		
	语言表示	文本等价	[32,103]		
	图像表示	图像等价	[98]		
模型搜索	属性表示	属性等价	[84−86]	ATM[24,91], CraftDroid[23]	
		文本等价	[23−25,88,90−91]		
	图像表示	图像等价	[98]	METER[98]	
研究趋势	已有工作倾向于以人类视角进行录制重放, 以属性表示、属性等价和零搜索为主流, 呈现出图像表示、图像等价和模型搜索的发展趋势.				

7 未来研究方向

本节基于前文对已有工作的系统性分析和讨论, 尝试对未来可能的研究方向进行探讨.

数据驱动的录制重放: 应用的复杂化和设备的碎片化预告低侵入的录制方式和人类视角的语义理解. 为兼容更多设备、方便普通用户使用, 低侵入的录制方式如录屏依然是最理想的形式. 这种录制方式要求录制工具能够进行视频分析和理解, 并从中提取事件序列和应用语义.

其中面临的第一个问题是数据集的构造: 如何利用现有的应用构建一个适用于录制重放的组件或事件序列数据集. 尽管目前存在此类数据集[116], 但该数据集中的事件序列平均长度较短, 且部分序列对应界面信息丢失, 导致该数据集难以用于长序列的录制重放.

移动应用具有同类组件差异性大 (high in-class variance), 异类组件相似性高 (high cross-class similarity) 的特点, 使得计算机视觉中的物体检测算法难以直接被直接应用[102]. 因此如何优化现有的计算机视觉算法、建立适用

于移动应用的深度学习模型, 并用于录制重放 (如界面分割和组件识别) 值得期待.

另一个值得被研究的是组件嵌入[116]. 类似于词嵌入, 组件嵌入的目标是使语义相似的组件在向量空间中具有相近的空间距离. 现有的向量化手段利用组件属性将组件文档化后向量化, 由于文档化采用了用特殊字符连接等方式, 因此该向量化易受文档构建不合理的影响. 另外, 该方案仅保留了组件的文本信息, 丢失了组件的图像信息, 因此亦造成了信息丢失. 如何结合组件的文本信息和图像信息对组件进行嵌入化仍是值得研究的问题.

针对特定场景的高效搜索: 特定场景通常拥有更多针对配置点的假设和先验, 因此可以设计出比模型搜索更高效的局部搜索策略. 如同应用同版本的跨设备重放: 该场景因仅有设备不同, 故可利用跨设备应用设计的特点进行针对性录制重放. 一种可选的方式是利用应用设计的空间局部性 (spatial locality) 和响应式模式 (responsive pattern) 特征, 将同应用同版本跨设备重放的搜索效率进行优化.

应用模型的构建: 如今基于模型的搜索仍然只被应用于简单类别, 原因是复杂应用的模型难以构建. 以 Microsoft Word 为例, 该应用的编辑界面存在大量的组件. 尽管不同组件包含不同的语义, 但在组件上发送事件却不影响模型状态的变迁. 此类应用构建的模型较为密集, 即状态少、转移多, 使得该模型难以对搜索产生有意义的指导. 因此, 如何将具有此类特征的模型进行优化 (如提出更好的状态表示方案、建立智能可演化的模型等) 亟待解决.

8 总 结

本文对现有基于 GUI 事件的安卓应用录制重放关键技术进行了问题分析, 并讨论了 3 个核心要素: 组件表示与录制技术、事件等价策略和局部搜索策略. 通过对现有工作进行总结分析, 本文展示了每个要素的主流策略, 并详细分析了每种策略适用的录制重放场景. 最后, 本文对录制重放技术的演变过程、未来可能的发展趋势和未来值得研究的问题进行了试探性探讨. 希望本文能对未来的录制重放工作产生指导, 并促进基于 GUI 事件的安卓应用录制重放技术的发展.

References:

[1] Number of android apps on google play. 2021. https://www.appbrain.com/stats/number-of-android-apps

[2] Mobile operating system market share worldwide. 2021. https://gs.statcounter.com/os-market-share/mobile/worldwide

[3] 2016 NowSecure mobile security report. 2021. https://info.nowsecure.com/rs/201-XEW-873/images/2016-NowSecure-mobile-security-report.pdf

[4] UI/Application exerciser monkey. 2021. https://developer.android.com/studio/test/monkey

[5] Machiry A, Tahiliani R, Naik M. Dynodroid: An input generation system for android apps. In: Proc. of the 9th Joint Meeting on Foundations of Software Engineering. Saint Petersburg: ACM, 2013. 224–234. [doi: 10.1145/2491411.2491450]

[6] Azim T, Neamtiu I. Targeted and depth-first exploration for systematic testing of android apps. In: Proc. of the 2013 ACM SIGPLAN Int'l Conf. on Object Oriented Programming Systems Languages & Applications. Indianapolis: ACM, 2013. 641–660. [doi: 10.1145/2509136.2509549]

[7] Choi W, Necula G, Sen K. Guided GUI testing of android apps with minimal restart and approximate learning. In: Proc. of the 2013 ACM SIGPLAN Int'l Conf. on Object Oriented Programming Systems Languages & Applications. Indianapolis: ACM, 2013. 623–640. [doi: 10.1145/2509136.2509552]

[8] Mahmood R, Mirzaei N, Malek S. EvoDroid: Segmented evolutionary testing of android apps. In: Proc. of the 22nd ACM SIGSOFT Int'l Symp. on Foundations of Software Engineering. Hong Kong: ACM, 2014. 599–609. [doi: 10.1145/2635868.2635896]

[9] Su T, Meng GZ, Chen YT, Wu K, Yang WM, Yao Y, Pu GG, Liu Y, Su ZD. Guided, stochastic model-based GUI testing of Android apps. In: Proc. of the 11th Joint Meeting on Foundations of Software Engineering. Paderborn: ACM, 2017. 245–256. [doi: 10.1145/3106237.3106298]

[10] Mao K, Harman M, Jia Y. Sapienz: Multi-objective automated testing for android applications. In: Proc. of the 25th Int'l Symp. on Software Testing and Analysis. Saarbrücken: ACM, 2016. 94–105. [doi: 10.1145/2931037.2931054]

[11] Gu TX, Sun CN, Ma XX, Cao C, Xu C, Yao Y, Zhang QR, Lu J, Su ZD. Practical GUI testing of Android applications via model abstraction and refinement. In: Proc. of the 41st Int'l Conf. on Software Engineering. Montreal: IEEE, 2019. 269–280. [doi: 10.1109/

ICSE.2019.00042]

[12] Li YC, Yang ZY, Guo Y, Chen XQ. DroidBot: A lightweight UI-guided test input generator for Android. In: Proc. of the 39th Int'l Conf. on Software Engineering Companion. Buenos: IEEE, 2017. 23–26. [doi: 10.1109/ICSE-C.2017.8]

[13] Li YC, Yang ZY, Guo Y, Chen XQ. Humanoid: A deep learning-based approach to automated black-box android app testing. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering. San Diego: IEEE, 2019. 1070–1073. [doi: 10.1109/ASE.2019.00104]

[14] Dong Z, Böhme M, Cojocaru L, Roychoudhury A. Time-travel testing of Android apps. In: Proc. of the 42nd Int'l Conf. on Software Engineering. Seoul: IEEE, 2020. 481–492.

[15] Wang J, Jiang YY, Xu C, Cao C, Ma XX, Lu J. ComboDroid: Generating high-quality test inputs for Android apps via use case combinations. In: Proc. of the 42nd Int'l Conf. on Software Engineering. Seoul: IEEE, 2020. 469–480.

[16] Pan MX, Huang A, Wang GX, Zhang T, Li XD. Reinforcement learning based curiosity-driven testing of Android applications. In: Proc. of the 29th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. ACM, 2020. 153–164. [doi: 10.1145/3395363.3397354]

[17] Adamsen CQ, Mezzetti G, Møller A. Systematic execution of android test suites in adverse conditions. In: Proc. of the 2015 Int'l Symp. on Software Testing and Analysis. Baltimore: ACM, 2015. 83–93. [doi: 10.1145/2771783.2771786]

[18] Wang J, Jiang YY, Xu C, Ma XX, Lu J. Automatic test-input generation for Android applications. SCIENTIA SINICA Informationis, 2019, 49(10): 1234–1266 (in Chinese with English abstract). [doi: 10.1360/N112019-00003]

[19] Android fragmentation visualized (August 2015). 2021. https://www.opensignal.com/sites/opensignal-com/files/data/reports/global/data-2015–08/2015_08_fragmentation_report.pdf

[20] Android version history. 2021. https://en.wikipedia.org/wiki/Android_version_history

[21] Taobao. 2021. https://play.google.com/store/apps/details?id=com.taobao.taobao

[22] Tianmao. 2021. https://play.google.com/store/apps/details?id=com.tmall.wireless

[23] Lin JW, Jabbarvand R, Malek S. Test transfer across mobile apps through semantic mapping. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering. San Diego: IEEE, 2019. 42–53. [doi: 10.1109/ASE.2019.00015]

[24] Behrang F, Orso A. AppTestMigrator: A tool for automated test migration for Android apps. In: Proc. of the 42nd Int'l Conf. on Software Engineering: Companion Proc.. Seoul: IEEE, 2020. 17–20.

[25] Qin X, Zhong H, Wang XY. TestMig: Migrating GUI test cases from iOS to Android. In: Proc. of the 28th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Beijing: ACM, 2019. 284–295. [doi: 10.1145/3293882.3330575]

[26] Lam W, Wu ZK, Li DF, Wang WY, Zheng HB, Luo H, Yan P, Deng YT, Xie T. Record and replay for Android: Are we there yet in industrial cases? In: Proc. of the 11th Joint Meeting on Foundations of Software Engineering. Paderborn: ACM, 2017. 854–859. [doi: 10.1145/3106237.3117769]

[27] Zhao YX, Chen J, Sejfia A, Laser MS, Zhang J, Sarro F, Harman M, Medvidovic N. FrUITeR: A framework for evaluating UI test reuse. In: Proc. of the 28th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. ACM, 2020. 1190–1201. [doi: 10.1145/3368089.3409708]

[28] Snowball sampling. https://en.wikipedia.org/wiki/Snowball_sampling

[29] Flinn J, Mao ZM. Can deterministic replay be an enabling tool for mobile computing? In: Proc. of the 12th Workshop on Mobile Computing Systems and Applications. Phoenix: ACM, 2011. 84–89. [doi: 10.1145/2184489.2184507]

[30] Di Martino S, Fasolino AR, Starace LLL, Tramontana P. Comparing the effectiveness of capture and replay against automatic input generation for Android graphical user interface testing. Software: Testing, Verification and Reliability, 2021, 31(3): e1754. [doi: 10.1002/stvr.1754]

[31] Mariani L, Mohebbi A, Pezzè M, Terragni V. Semantic matching of GUI events for test reuse: Are we there yet? In: Proc. of the 30th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. ACM, 2021. 177–190. [doi: 10.1145/3460319.3464827]

[32] Zhao Y, Yu TT, Su T, Liu Y, Zheng W, Zhang JZ, Halfond WGJ. ReCDroid: Automatically reproducing android application crashes from bug reports. In: Proc. of the 41st Int'l Conf. on Software Engineering. Montreal: IEEE, 2019. 128–139. [doi: 10.1109/ICSE.2019.00030]

[33] Hoare CAR. An axiomatic basis for computer programming. Communications of the ACM, 1969, 12(10): 576–583. [doi: 10.1145/363235.363259]

[34] Gomez L, Neamtiu I, Azim T, Millstein T. RERAN: Timing- and touch-sensitive record and replay for Android. In: Proc. of the 35th Int'l Conf. on Software Engineering. San Francisco: IEEE, 2013. 72–81. [doi: 10.1109/ICSE.2013.6606553]

[35] Qin ZR, Tang YT, Novak E, Li Q. MobiPlay: A remote execution based record-and-replay tool for mobile applications. In: Proc. of the 38th Int'l Conf. on Software Engineering. Austin: ACM, 2016. 571–582. [doi: 10.1145/2884781.2884854]

[36] Hu YJ, Azim T, Neamtiu I. Versatile yet lightweight record-and-replay for Android. In: Proc. of the 2015 ACM SIGPLAN Int'l Conf. on Object-oriented Programming, Systems, Languages, and Applications. Pittsburgh: ACM, 2015. 349–366. [doi: 10.1145/2814270.2814320]

[37] White M, Linares-Vásquez M, Johnson P, Bernal-Cárdenas C, Poshyvanyk D. Generating reproducible and replayable bug reports from android application crashes. In: Proc. of the 23rd Int'l Conf. on Program Comprehension. Florence: IEEE, 2015. 48–59. [doi: 10.1109/ICPC.2015.14]

[38] Amalfitano D, Riccio V, Amatucci N, De Simone V, Fasolino AR. Combining automated GUI exploration of android apps with capture and replay through machine learning. Information and Software Technology, 2019, 105: 95–116. [doi: 10.1016/j.infsof.2018.08.007]

[39] Halpern M, Zhu YH, Peri R, Reddi VJ. Mosaic: Cross-platform user-interaction record and replay for the fragmented android ecosystem. In: Proc. of the 2015 Int'l Symp. on Performance Analysis of Systems and Software. Philadelphia: IEEE, 2015. 215–224. [doi: 10.1109/ISPASS.2015.7095807]

[40] Fang KM, Yan GH. IoTReplay: Troubleshooting COTS IoT devices with record and replay. In: Proc. of 2020 IEEE/ACM Symp. on Edge Computing. San Jose: IEEE, 2020. 193–205. [doi: 10.1109/SEC50012.2020.00033]

[41] Nurmuradov D, Bryce R. Caret-HM: Recording and replaying Android user sessions with heat map generation using UI state clustering. In: Proc. of the 26th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Santa Barbara: ACM, 2017. 400–403. [doi: 10.1145/3092703.3098231]

[42] Appetizerio/Replaykit. 2021. https://github.com/appetizerio/replaykit

[43] Guo JQ, Li SY, Lou JG, Yang ZJ, Liu T. Sara: Self-replay augmented record and replay for Android in industrial cases. In: Proc. of the 28th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Beijing: ACM, 2019. 90–100. [doi: 10.1145/3293882.3330557]

[44] Botbot. 2021. https://github.com/menonvarun/andro_auto_framework

[45] CulebraTester. 2021. http://culebra.dtmilano.com

[46] Robotium recorder. 2021. https://github.com/RobotiumTech/robotium

[47] Monkeyrunner. 2021. https://developer.android.com/studio/test/monkeyrunner

[48] Ranorex. 2021. https://www.ranorex.com/mobile-automation-testing/android-test-automation

[49] Appium. 2021. https://appium.io

[50] EyeStudio. 2021. https://eyeautomate.com/eyestudio

[51] Robo. 2021. https://firebase.google.com/docs/test-lab/android/robo-ux-test

[52] Airtest. 2021. https://github.com/AirtestProject/Airtest

[53] Qian J, Shang ZY, Yan SY, Wang Y, Chen L. RoScript: A visual script driven truly non-intrusive robotic testing system for touch screen applications. In: Proc. of the 42nd ACM/IEEE Int'l Conf. on Software Engineering. Seoul: ACM, 2020. 297–308. [doi: 10.1145/3377811.3380431]

[54] Negara S, Esfahani N, Buse R. Practical Android test recording with espresso test recorder. In: Proc. of the 41st Int'l Conf. on Software Engineering: Software Engineering in Practice. Montreal: IEEE, 2019. 193–202. [doi: 10.1109/ICSE-SEIP.2019.00029]

[55] Yu SC, Fang CR, Feng Y, Zhao WY, Chen ZY. LIRAT: Layout and image recognition driving automated mobile testing of cross-platform. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering. San Diego: IEEE, 2019. 1066–1069. [doi: 10.1109/ASE.2019.00103]

[56] Bell S, McDiarmid A, Irvine J. Nodobo: Mobile phone as a software sensor for social network research. In: Proc. of the 73rd IEEE Vehicular Technology Conf. (VTC Spring). 2011. 1–5. [doi: 10.1109/VETECS.2011.5956319]

[57] Build more accessible apps. 2021. https://developer.android.com/guide/topics/ui/accessibility

[58] Get started on Android with TalkBack. 2021. https://support.google.com/accessibility/android/answer/6283677?hl=en

[59] Create your own accessibility service. 2021. https://developer.android.com/guide/topics/ui/accessibility/service

[60] Fazzini M, De A. Freitas EN, Choudhary SR, Orso A. Barista: A technique for recording, encoding, and running platform independent android tests. In: Proc. of the 2017 IEEE Int'l Conf. on Software Testing, Verification and Validation. Tokyo: IEEE, 2017. 149–160. [doi: 10.1109/ICST.2017.21]

[61] Arruda F, Sampaio A, Barros FA. Capture & replay with text-based reuse and framework agnosticism. In: Proc. of the 28th Int'l Conf. on Software Engineering and Knowledge Engineering. San Francisco: KSI Research Inc., Knowledge Systems Institute Graduate School, 2016. 420–425.

[62] Huang J, Backes M, Bugiel S. A11y and privacy don't have to be mutually exclusive: Constraining accessibility service misuse on Android. In: Proc. of the 30th USENIX Security Symp. USENIX Association, 2021. 3631–3648.

[63] Jang Y, Song CY, Chung SP, Wang TL, Lee W. A11y attacks: Exploiting accessibility in operating systems. In: Proc. of the 2014 ACM

SIGSAC Conf. on Computer and Communications Security. Scottsdale: ACM, 2014. 103–115. [doi: 10.1145/2660267.2660295]

[64] Kalysch A, Bove D, Müller T. How Android's UI security is undermined by accessibility. In: Proc. of the 2nd Reversing and Offensive-oriented Trends Symp. Vienna: ACM, 2018. 2. [doi: 10.1145/3289595.3289597]

[65] Apktool. 2021. https://ibotpeaches.github.io/Apktool

[66] Androguard. 2021. https://github.com/androguard/androguard

[67] Soot. 2021. https://github.com/soot-oss/soot

[68] Frida. 2021. https://frida.re/docs/examples/android

[69] Sahin O, Aliyeva A, Mathavan H, Coskun A, Egele M. RANDR: Record and replay for android applications via targeted runtime instrumentation. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering. San Diego: IEEE, 2019. 128–138. [doi: 10.1109/ASE.2019.00022]

[70] Sahin O, Aliyeva A, Mathavan H, Coskun A, Egele M. Towards practical record and replay for mobile applications. In: Proc. of the 56th Annual Design Automation Conf. Las Vegas: ACM, 2019. 230. [doi: 10.1145/3316781.3322476]

[71] Xposed. 2021. https://repo.xposed.info/module/de.robv.android.xposed.installer

[72] VirtualXposed. 2021. https://github.com/android-hacker/VirtualXposed

[73] Microsoft word: Write, edit & share docs on the go. 2021. https://play.google.com/store/apps/details?id=com.microsoft.office.word

[74] Gianazza A, Maggi F, Fattori A, Cavallaro L, Zanero S. PuppetDroid: A user-centric UI exerciser for automatic dynamic analysis of similar android applications. arXiv: 1402.4826, 2014.

[75] Linares-Vásquez M, White M, Bernal-Cárdenas C, Moran K, Poshyvanyk D. Mining android app usages for generating actionable GUI-based execution scenarios. In: Proc. of the 12th Working Conf. on Mining Software Repositories. Florence: IEEE, 2015. 111–122. [doi: 10.1109/MSR.2015.18]

[76] Park CM, Ki T, Ben Ali AJ, Pawar NS, Dantu K, Ko SY, Ziarek L. Gesto: Mapping UI events to gestures and voice commands. Proc. of the ACM on Human-computer Interaction, 2019, 3(EICS): 5. [doi: 10.1145/3300964]

[77] Liu CH, Lu CY, Cheng SJ, Chang KY, Hsiao YC, Chu WM. Capture-replay testing for android applications. In: Proc. of the 2014 Int'l Symp. on Computer, Consumer and Control. Taichung: IEEE, 2014. 1129–1132. [doi: 10.1109/IS3C.2014.293]

[78] Zhao LX, An HY, Wang XY, Wang XH. A record-replay based snapshot system for android application. In: Proc. of the 2016 Int'l Conf. on Progress in Informatics and Computing. Shanghai: IEEE, 2016. 679–683. [doi: 10.1109/PIC.2016.7949585]

[79] Zheng JH, Shen LW, Peng X, Zeng HC, Zhao WY. MashReDroid: Enabling end-user creation of Android mashups based on record and replay. Science China Information Sciences, 2020, 63(10): 202101. [doi: 10.1007/s11432-019-2646-2]

[80] Moran K, Linares-Vásquez M, Bernal-Cárdenas C, Vendome C, Poshyvanyk D. CrashScope: A practical tool for automated testing of android applications. In: Proc. of the 39th Int'l Conf. on Software Engineering Companion. Buenos Aires: IEEE, 2017. 15–18. [doi: 10.1109/ICSE-C.2017.16]

[81] Nandakumar V, Ekambaram V, Sharma V. Appstrument—A unified app instrumentation and automated playback framework for testing mobile applications. In: Proc. of the 10th Int'l Conf. on Mobile and Ubiquitous Systems: Computing, Networking, and Services. Tokyo: Springer, 2013. 474–486. [doi: 10.1007/978-3-319-11569-6_37]

[82] Ardito L, Coppola R, Torchiano M, Alégroth E. Towards automated translation between generations of GUI-based tests for mobile devices. In: Companion Proc. for the ISSTA/ECOOP 2018 Workshops. Amsterdam: ACM, 2018. 46–53. [doi: 10.1145/3236454.3236488]

[83] Coppola R, Ardito L, Torchiano M, Alégroth E. Translation from visual to layout-based Android test cases: A proof of concept. In: Proc. of the 2020 IEEE Int'l Conf. on Software Testing, Verification and Validation Workshops. Porto: IEEE, 2020. 74–83. [doi: 10.1109/ICSTW50294.2020.00027]

[84] Li X, Chang NN, Wang Y, Huang HH, Pei Y, Wang LZ, Li XD. ATOM: Automatic maintenance of GUI test scripts for evolving mobile applications. In: Proc. of the 2017 IEEE Int'l Conf. on Software Testing, Verification and Validation. Tokyo: IEEE, 2017. 161–171. [doi: 10.1109/ICST.2017.22]

[85] Chang NN, Wang LZ, Pei Y, Mondal SK, Li XD. Change-based test script maintenance for Android apps. In: Proc. of the 2018 IEEE Int'l Conf. on Software Quality, Reliability and Security. Lisbon: IEEE, 2018. 215–225. [doi: 10.1109/QRS.2018.00035]

[86] Gómez M, Rouvoy R, Adams B, Seinturier L. Reproducing context-sensitive crashes of mobile apps using crowdsourced monitoring. In: Proc. of the 2016 IEEE/ACM Int'l Conf. on Mobile Software Engineering and Systems. Austin: IEEE, 2016. 88–99. [doi: 10.1109/MobileSoft.2016.033]

[87] Xu TT, Pan MX, Pei Y, Li GY, Zeng X, Zhang T, Deng YT, Li XD. GUIDER: GUI structure and vision co-guided test script repair for Android apps. In: Proc. of the 30th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. ACM, 2021. 191–203. [doi: 10.1145/

3460319.3464830]
[88] Behrang F, Orso A. Test migration for efficient large-scale assessment of mobile app coding assignments. In: Proc. of the 27th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Amsterdam: ACM, 2018. 164–175. [doi: 10.1145/3213846.3213854]
[89] Mariani L, Pezzè M, Terragni V, Zuddas D. An evolutionary approach to adapt tests across mobile apps. In: Proc. of the 2021 ACM/IEEE Int'l Conf. on Automation of Software Test. Madrid: IEEE, 2021. 70–79. [doi: 10.1109/AST52587.2021.00016]
[90] Behrang F, Orso A. Automated test migration for mobile apps. In: Proc. of the 40th Int'l Conf. on Software Engineering: Companion Proceedings. Gothenburg: ACM, 2018. 384–385. [doi: 10.1145/3183440.3195019]
[91] Behrang F, Orso A. Test migration between mobile apps with similar functionality. In: Proc. of the 34th IEEE/ACM Int'l Conf. on Automated Software Engineering. San Diego: IEEE, 2019. 54–65. [doi: 10.1109/ASE.2019.00016]
[92] Moran K, Linares-Vásquez M, Bernal-Cárdenas C, Vendome C, Poshyvanyk D. Automatically discovering, reporting and reproducing android application crashes. In: Proc. of the 2016 IEEE Int'l Conf. on Software Testing, Verification and Validation. Chicago: IEEE, 2016. 33–44. [doi: 10.1109/ICST.2016.34]
[93] UI automator. 2021. https://developer.android.com/training/testing/ui-automator
[94] Binder. 2021. https://developer.android.com/reference/android/os/Binder
[95] Chen JS, Chen CY, Xing ZC, Xu XW, Zhut L, Li GQ, Wang JS. Unblind your apps: Predicting natural-language labels for mobile GUI components by deep learning. In: Proc. of the 42nd Int'l Conf. on Software Engineering. Seoul: IEEE, 2020. 322–334.
[96] Wang CY, Chu WC, Chen HR, Hsu CY, Chen MY. EverTutor: Automatically creating interactive guided tutorials on smartphones by user demonstration. In: Proc. of the SIGCHI Conf. on Human Factors in Computing Systems. Toronto: ACM, 2014. 4027–4036. [doi: 10.1145/2556288.2557407]
[97] Krieter P, Breiter A. Analyzing mobile application usage: Generating log files from mobile screen recordings. In: Proc. of the 20th Int'l Conf. on Human-computer Interaction with Mobile Devices and Services. Barcelona: ACM, 2018. 9. [doi: 10.1145/3229434.3229450]
[98] Pan MX, Xu TT, Pei Y, Li Z, Zhang T, Li XD. GUI-guided test script repair for mobile apps. IEEE Trans. on Software Engineering, 2020. [doi: 10.1109/TSE.2020.3007664]
[99] Bernal-Cárdenas C, Cooper N, Moran K, Chaparro O, Marcus A, Poshyvanyk D. Translating video recordings of mobile app usages into replayable scenarios. In: Proc. of the 42nd ACM/IEEE Int'l Conf. on Software Engineering. Seoul: ACM, 2020. 309–321. [doi: 10.1145/3377811.3380328]
[100] Yeh T, Chang TH, Miller RC. Sikuli: Using GUI screenshots for search and automation. In: Proc. of the 22nd Annual ACM Symp. on User Interface Software and Technology. Victoria: ACM, 2009. 183–192. [doi: 10.1145/1622176.1622213]
[101] Nguyen TA, Csallner C. Reverse engineering mobile application user interfaces with REMAUI (T). In: Proc. of the 30th IEEE/ACM Int'l Conf. on Automated Software Engineering. Lincoln: IEEE, 2015. 248–259. [doi: 10.1109/ASE.2015.32]
[102] Chen JS, Xie ML, Xing ZC, Chen CY, Xu XW, Zhu LM, Li GQ. Object detection for graphical user interface: Old fashioned or deep learning or a combination? In: Proc. of the 28th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Virtual: ACM, 2020. 1202–1214. [doi: 10.1145/3368089.3409691]
[103] Fazzini M, Prammer M, d'Amorim M, Orso A. Automatically translating bug reports into test cases for mobile apps. In: Proc. of the 27th ACM SIGSOFT Int'l Symp. on Software Testing and Analysis. Amsterdam: ACM, 2018. 141–152. [doi: 10.1145/3213846.3213869]
[104] Li Y, He JC, Zhou X, Zhang Y, Baldridge J. Mapping natural language instructions to mobile UI action sequences. In: Proc. of the 58th Annual Meeting of the Association for Computational Linguistics. ACL, 2020. 8198–8210.
[105] Hu G, Zhu LJ, Yang JF. AppFlow: Using machine learning to synthesize robust, reusable UI tests. In: Proc. of the 26th ACM Joint Meeting on European Software Engineering Conf. and Symp. on the Foundations of Software Engineering. Lake Buena Vista: ACM, 2018. 269–282. [doi: 10.1145/3236024.3236055]
[106] TF-IDF. 2021. https://en.wikipedia.org/wiki/Tf-idf
[107] Word Embedding. 2021. https://en.wikipedia.org/wiki/Word_embedding
[108] Mikolov T, Sutskever I, Chen K, Corrado G, Dean J. Distributed representations of words and phrases and their compositionality. In: Proc. of the 26th Int'l Conf. on Neural Information Processing Systems. Lake Tahoe: Curran Associates Inc., 2013. 3111–3119.
[109] Pennington J, Socher R, Manning C. GloVe: Global vectors for word representation. In: Proc. of the 2014 Conf. on Empirical Methods in Natural Language Processing. Doha: ACL, 2014. 1532–1543. [doi: 10.3115/v1/D14-1162]
[110] WordNet. 2021. https://wordnet.princeton.edu
[111] Dalal N, Triggs B. Histograms of oriented gradients for human detection. In: Proc. of the 2005 IEEE Computer Society Conf. on Computer Vision and Pattern Recognition. San Diego: IEEE, 2005. 886–893. [doi: 10.1109/CVPR.2005.177]

[112] Lowe DG. Distinctive image features from scale-invariant keypoints. Int'l Journal of Computer Vision, 2004, 60(2): 91–110. [doi: 10.1023/B:VISI.0000029664.99615.94]

[113] Muja M, Lowe DG. Fast approximate nearest neighbors with automatic algorithm configuration. In: Proc. of the 4th Int'l Conf. on Computer Vision Theory and Applications. Lisboa: INSTICC Press, 2009. 331–340.

[114] Monga V, Evans BL. Perceptual image hashing via feature points: Performance evaluation and tradeoffs. IEEE Trans. on Image Processing, 2006, 15(11): 3452–3465. [doi: 10.1109/TIP.2006.881948]

[115] Baek YM, Bae DH. Automated model-based Android GUI testing using multi-level GUI comparison criteria. In: Proc. of the 31st IEEE/ACM Int'l Conf. on Automated Software Engineering. Singapore: IEEE, 2016. 238–249.

[116] Deka B, Huang ZF, Franzen C, Hibschman J, Afergan D, Li Y, Nichols J, Kumar R. Rico: A mobile app dataset for building data-driven design applications. In: Proc. of the 30th Annual ACM Symp. on User Interface Software and Technology. 2017. 845–854. [doi: 10.1145/3126594.3126651]

附中文参考文献:

[18] 王珏, 蒋炎岩, 许畅, 马晓星, 吕建. Android 应用测试输入自动生成技术. 中国科学: 信息科学, 2019, 49(10): 1234–1266. [doi: 10.1360/N112019-00003]

李聪(1996－), 男, 博士生, CCF 学生会员, 主要研究领域为安卓应用分析与测试.

许畅(1977－), 男, 博士, 教授, 博士生导师, CCF 高级会员, 主要研究领域为软件测试与分析, 自适应软件系统.

蒋炎岩(1988－), 男, 博士, CCF 专业会员, 主要研究领域为软件分析与测试.