分类号_____ 编 号_____

U D C_____ 密 级_____

南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

本科生毕业设计（论文）

题　　目：跨设备的 Android 应用录制与回放工具

姓　　名：余添诚

学　　号：11712019

系　　别：计算机科学与工程系

专　　业：计算机科学与技术

指导教师：刘烨庞教授

2021 年　月　日

诚信承诺书

1. 本人郑重承诺所呈交的毕业设计（论文），是在导师的指导下，独立进行研究工作所取得的成果，所有数据、图片资料均真实可靠。
2. 除文中已经注明引用的内容外，本论文不包含任何其他人或集体已经发表或撰写过的作品或成果。对本论文的研究作出重要贡献的个人和集体，均已在文中以明确的方式标明。
3. 本人承诺在毕业论文（设计）选题和研究内容过程中没有抄袭他人研究成果和伪造相关数据等行为。
4. 在毕业论文（设计）中对侵犯任何方面知识产权的行为，由本人承担相应的法律责任。

作者签名: _____

_____ 年___ 月___ 日

目 录

摘 要

随着移动设备的日趋流行，对于移动应用的自动化测试成为降低开发者时间成本和保障产品质量的关键点。本研究设计并实现了一套能够在不同的 Android 设备上录制和回放应用输入的工具，并提升录制与回放的性能及准确性等相关性能参数。在避免使用定制化系统或修改被测应用源代码的情况下，能够准确高效地录制来自触屏及各项传感器（如陀螺仪、GPS、摄像头）的输入，并在不同型号的 Android 设备上自适应地回放。我们通过实验展示了该工具在? 种不同的实验环境下（含不同型号真实设备以及模拟器）以及超过? 款商业闭源或开源应用上测试的结果。

关键词： Google Android, 应用测试, 移动应用, 录制与回放

ABSTRACT

With the increasing popularity of mobile devices, automated testing for mobile applications has become the key point for lowering developers' time costs and assuring product quality. The main goal of this work is to implement a Record-and-Replay system for various Android devices, improving the quality and accuracy of the results, without using customized operating system or modifying application source code, while be able to accurately and efficiently record from touch screen and various sensors (e.g. gyroscope, GPS, camera), and replay on different models of Android devices in an adaptive manner. We demonstrate the tool's ability by performing experiments on ? different environments (including various models of real-world devices and emulators) with over ? commercial closed-source or open-source applications.

Keywords: Google Android, App testing, Mobile applications, Record-and-replay

第一章 导言

随着移动设备的日趋流行，自动化测试移动应用的关键性日渐显著。然而，在目前的企业开发环境中，应用测试仍然主要以高时间成本的人工方式进行，能够在不同设备上进行录制和回放将显著地减轻开发者和测试人员的负担。

与传统程序不同的是，移动应用中多样化的用户输入方式——触摸屏和不同的传感器（如陀螺仪，GPS，摄像头），以及屏幕尺寸和系统版本的差异化，给应用的自动化测试带来了挑战。触屏、按键和各种传感器等难以通过传统方法录制和重放的因素对于移动应用的用户界面交互结果至关重要。移动设备较弱的性能也使精确地录制与回放对时间敏感的操作序列存在困难。

现今学术界与工业界已有多种不同录制与回放的方案，但各自均有一定的缺陷和局限性。基于定制系统的方案 [1] 需要针对每个机型编写和安装定制化系统，开发以及维护成本高昂；基于内核事件的方案 [2] 从内核级别获取的传感器数据有限，难以满足录制较高层级数据（网络、GPS）的需求；基于静态插桩 [3] 则难以对加密的商业闭源应用使用，不便于商业公司外派测试工作；基于屏幕坐标的回放 [2] 需要在不同的设备上多次重复录制相似的输入，增加用户的时间成本；基于 root 权限的动态插桩 [2] 在部分品牌或机型上难以实现，并可能导致机器失去品牌保修。

文献 [4] 通过实证研究指出，现有的录制与回放工具在有效性、性能以及可靠性上仍然无法满足开发者的实际需求。在研究中，来自微信 [5] 的开发者提出其理想的录制与回放工具所应满足的功能与限制。在文中，微信开发者希望存在相关工具能够：（一）开源具体实现；（二）基于坐标录制动作；（三）基于界面组件录制操作；（四）对应用具体状态不敏感；（五）记录多个操作之间的时间间隔；以及尽量避免：（一）需要对应用插桩；（二）需要定制化系统；（三）需要 root 权限；（四）需要应用源代码。

本文提出一种跨平台、跨设备的 Android 应用录制与回放工具，通过静态修改应用文件载入插桩类库，在运行时动态插桩应用以及系统关键库调用，在最大程度地规避上述局限性的情况下高效地提供可自适应环境、状态无关、时间准确、对多种输入有效的录制与回放。通过对关键目标函数的动态插桩，该工具得以在无需用户人工介入进行应用或平台相关的配置的条件下，自动拦截并记录各类不同输入的内容和上下文数据。录制用户的 UI 操作时，该工具对 Android 系统库内处理用户操作的函数进行插桩，截取具体输入以及目标界面组件的信息，从而实现不受屏幕坐标影响、对界面组件敏感的录制与回放功能；录制移动设备传感器数据时，该工具对应用内的传感器监听器进行插桩，在应用每次读取传感器数据时录制或回放传感器数据。

第二章 研究背景

本章节主要介绍本文所涉及的 Android 系统及应用的相关背景概念，如 Android 应用所能处理的不同类型用户输入，以及实现本文所述工具原型中使用到的技术背景。

2.1 Android 应用

Android 应用主要由 Java 编写，通过 Android SDK 编译为 Dalvik 字节码后由 Android Runtime (ART) 执行，部分代码亦可使用编译型语言编译为原生代码后加载执行。

在 Android 系统中，每个应用运行在自己独立的 ART 运行时里，通过 Android Framework 的 Java 库以及其他原生库（例如加解密相关的 libcrypto.so 和 libssl.so）提供应用所需的功能。

2.2 Android 应用界面

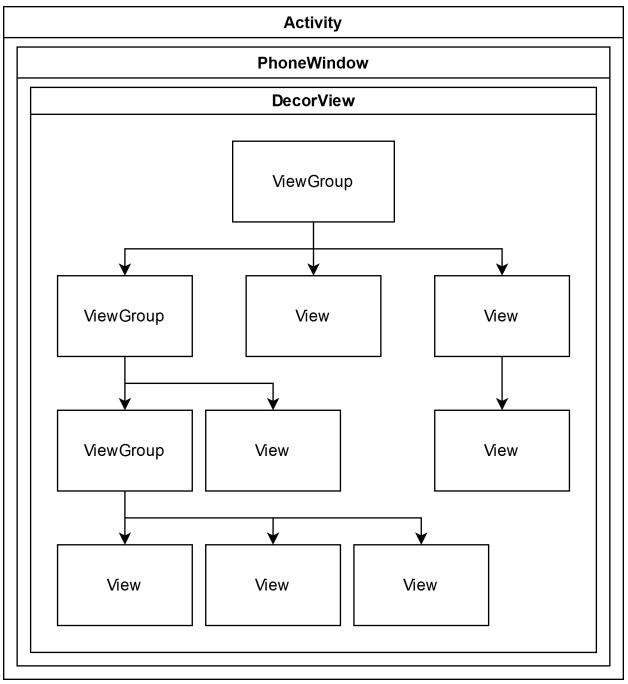

图 2.1: Android View 界面结构

Android Framework 为开发者提供了一系列预定义好的界面控件，例如文本框、选择框、按钮、图片和列表，开发者也可以继承 View 类开发自定义的控件，或继承 ViewGroup 类实现组合控件。每个 View 代表用户界面上的一个对象，应用运行时显示的界面实际为一个由 View 和 ViewGroup 组成的树状结构（见图 2.1）。应用可以在编译前通过资源文件定义界面结构，亦可在运行时动态地修改界面结构。

2.3 Android 用户输入

Android 应用中直接输入来源包含触摸屏和按键。用户对触摸屏的输入会触发 MotionEvent，通过 MotionEvent 的序列可以表达所有触摸屏操作（点击、滑动、长按）；按键输入会触发 KeyEvent。在用户触发事件后，Android 会从当前 Activity 的根节点向下搜索对应的 View 传递事件，路径中的 ViewGroup 可以选择自行处理或是继续传递给 View 子节点。应用亦可通过 Android SDK 以及 Java 核心库从硬件传感器、网络或其他来源获得输入。

2.4 动态插桩 ART VM 代码

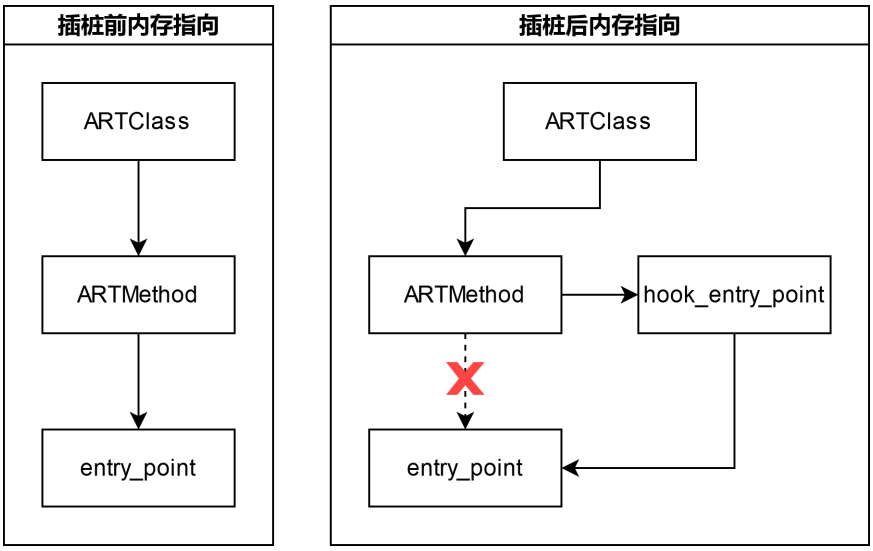

图 2.2: 动态插桩 ART 方法实现

在 ART 中，每个 Java Class 都由内存中的一个 ARTClass 对象代理内部数据结构，其中每一个函数对应着一个 ARTMethod 结构体。ARTMethod 实例中储存着具体方法实现的类型以及调用入口，因此可以通过注入 ARTMethod 类的内部变量，达到修改对应类或原生函数具体实现的目的（见图 2.2）。在本文所述工具中，我们通过 Frida[6] 注入 ART 修改目的函数的地址，以劫持或修改函数输入输出结果。

第三章　架构设计和实现

本章节主要描述本工具的设计思想和原因，以及如何在有限的条件下尽可能地实现和满足工业界的功能需求。

3.1　设计总览

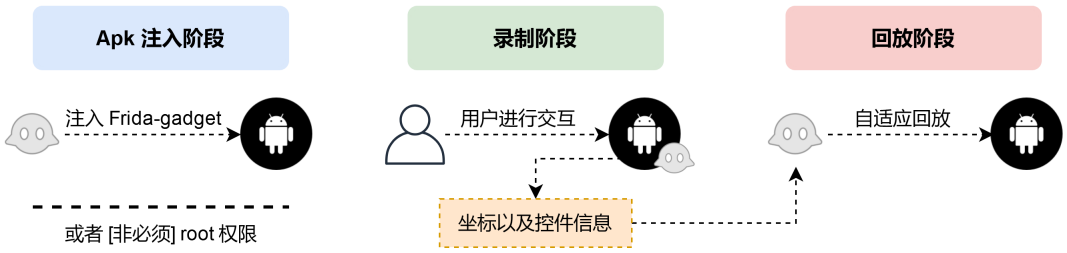

图 3.1: 设计总览

本工具主要由三个组件构成（见图 3.1）：Apk 注入器，录制代理以及回放代理。在开始录制前，先将动态插桩所需要的基础库注入到应用安装包中，如在 root 环境中可以跳过此步骤。在录制中，录制代理将会记录用户交互的坐标和控件信息，以及来自其他传感器（如 GPS、陀螺仪等）的相关数据。在回放时，回放代理将参考录制与回放所在设备的分辨率和尺寸差距，通过坐标和控件信息找到对应控件，并注入模拟的用户操作。

3.2　静态注入插桩模块

为了实现动态插桩，Frida 可以由 root 权限运行，或是借由注入到应用中的 Frida-gadget 模块进行操作。在本工具中，我们提供一个 Apk 注入器，自动将所需模块注入到应用原生库依赖中，以实现在应用启动时自动加载录制或回放代理。通过无需修改应用 Delvik 字节码的注入方式，本工具可以应用在经过字节码加密程序打包过的应用上。在用户拥有设备 root 权限时，亦可使用 root 进行动态插桩。

尽管微信的开发者在其需求中提出希望工具能避免使用插桩，以保证对不同应用和系统的兼容性，但 Frida 作为业界广泛使用的跨平台插桩工具之一，可以在 Android 4.4 至 Android 11 上无需 root 运行（覆盖 99.8% 以上的 Android 用户），其对于应用的兼容性也较好。在第四章的实验验证与测试中，本工具对于微信以及其他热门的商业应用的兼容可以一定程度上的缓解开发者对于工具兼容性的忧虑。

3.3　动态插桩录制与回放

在录制与回放阶段，代理将录制或回放来自各种来源的输入数据。对于 Android 应用而言，主要的输入来源可以分为触屏/按键以及传感器。本工具通过注入一系列

的 Android Framework API 函数以在框架层截获与录制足够的上下文信息和注入与回放模拟输入事件，而不失去对各种不同应用的泛用性。在下文中，我们将详细描述如何实现对不同种类输入来源的录制与回放。

3.3.1 触屏/按键的录制与回放

表 3.1: 录制与回放触屏/按键时 Android Framework 的主要注入点

注入类	录制注入函数	回放注入函数
View	View(), setOnTouchListener(), dispatchKeyEvent()	draw(), dispatchDraw()
ViewGroup	dispatchKeyEvent()	-

表 3.1中为本工具录制与回放触屏/按键操作时的主要注入点。

本工具通过对 Android Framework 的特定 API 进行插桩，拦截并录制或修改相关上下文信息以完成对应用触屏与按键的自适应录制与回放。在框架层的捕获和修改允许本工具在最大化应用兼容性的同时能够获得录制与回放所必须的上下文信息。使用控件自适应的录制与回放，让本工具得以在不同屏幕分辨率、尺寸、比例的设备上回放所录制的操作，而无需依赖控件的绝对或相对坐标。

为准确而稳定地在不同设备和环境中区分应用的不同控件，本工具对每个控件生成一个（尽可能）独特的签名。在一个控件被构造或重新绘制时，本工具将获取其控件类型、一定层数内子控件的签名、一定层数内父控件的签名以及控件自身在界面树状结构中的相对位置。本工具亦可参考控件所显示的文字信息生成签名，由于部分控件可能显示动态文字，开发者可根据应用场景选择是否使用控件文字信息。

在录制阶段中，以往的录制与回放工具 [7] 需要读取应用中的所有的 Activity 或 View 的子类并依次进行插桩，效率较低并且容易遗漏动态加载的类；本工具通过对 View 的构造器进行插桩，可以获得所有子类构造得到的对象实例并通过 setOnTouchListener() 方法监听控件所接收到的输入事件，同时对于应用自身对于触控监听的注册进行插桩，将录制代理监听器所接收到的事件重定向至应用自身监听器中，从而实现对应用透明的事件录制。录制时将保存控件的签名信息、时间戳以及具体记录的原始事件内容。

在回放阶段中，回放代理插桩 View.draw() 和 View.dispatchDraw()，在控件绘制时更新控件签名并保留对控件的引用，允许在无需坐标，乃至于控件在屏幕上不可见的情况下，持续追踪每个被应用创建的控件。对应录制时时间戳的顺序，本工具依次生成根据设备区别动态生成的输入事件，并在对应的时间点将其注入回 View 所在的事件传递链中，让应用和系统代码与用户真实触摸一样的流程处理模拟事件。在控件回放时，本工具亦会自动调整模拟触控事件相对于控件的位置，以正确回放对点击坐标敏感的控件（如绘图控件等）。回放代理同时支持自适应的坐标点击模

拟，在控件回放不可用的情况下能够满足一定情况下的触摸回放功能。开发者也可以根据应用场景的需求使用坐标点击替代控件点击。

3.3.2 传感器的录制与回放

表 3.2: 录制与回放传感器时 Android Framework 的主要注入点

注入类	注入函数
SensorManager	registerListener()
SensorEventListener (含子类)	onSensorChanged()
LocationManager	getLastKnownLocation(), requestLocationUpdates()
LocationListener (含子类)	onLocationChanged()

表 3.2中为本工具录制与回放传感器数据时的主要注入点。Android 设备中的传感器根据系统 API 实现可以分为底层传感器（如陀螺仪、加速度传感器、重力仪等）和高级传感器（如 GPS）。对于底层传感器，应用程序统一由 SensorManager 接口注册监听器，并由监听器接收传感器更新信息；而对于高级传感器，Android 系统对于每类传感器提供特化接口，如应用可通过 LocationManager 主动获取 GPS 信息，或是注册监听器接收 GPS 更新信息。本工具对传感器的主动读取的函数入口点以及被动事件驱动的监听器和相关注册函数进行插桩，从而获得数据的接收类以及所接收的数据等回放时必须的上下文信息，并在回放时根据录制时的时间顺序将传感器数据注入对应类中。

为避免 ART VM 对代码进行内联优化导致对监听器插桩失效，本工具在应用注册监听器时同步对相同传感器注册监听器进行数据记录，而无需禁用 ART 内联优化功能，有效提高录制时的性能。

第四章　实验验证及分析

本章节中，我们通过录制与回放过程的有效性和引入的性能额外开销作为参考以评估本工作中实现的工具，并探索以下问题的答案：

RQ1. 本工具对于相同设备上的录制与重放有效性如何？

RQ2. 本工具对于不同设备上的录制与回放有效性如何？

RQ3. 本工具引入的时间与空间的额外开销有多大？

4.0.1　实验环境

我们配置实验环境的主要目的是为评估和验证本工具在不同环境下的有效性（如虚拟机、真实设备、不同 Android 版本以及屏幕分辨率、DPI 等）。我们在 Google Play（Android 官方应用商店）以及 F-Droid（开源 Android 应用库）中根据应用的下载量和应用的分类选取了? 个应用（见表 4.1），并在一款真实设备（小米 9, Android 11）和两种不同屏幕分辨率和 DPI 显著不同的 Android 虚拟机：Nexus 4（Android 9）和 Pixel 4 XL（Android 10）上进行测试。所选取的应用均与所有实验环境兼容。如表? 中所示，所有选取的应用有 100 万以上的下载量，并且分别来自? 个不同的分类中。对于每个被选中的应用，我们根据该应用的使用场景和特性介绍选择 3 种不同的常见场景进行录制与回放实验。

对于 RQ1，我们在每个设备（Nexus 4 虚拟机和 Pixel 4 XL 虚拟机以及小米 9）上分别进行录制实验，并使用本机录制结果进行回放实验。我们根据文献 [4]，选择与其中效果最优的录制与回放工具 replaykit[8] 以及 RERAN[2] 进行对比。部分应用由于虚拟机兼容原因未能在虚拟机上运行，仅在真实设备上进行测试（应用名后标记 *）。另由于 replaykit 仅支持到 Android 9，对其跳过在 Pixel 4 XL 以及小米 9 上的测试。RERAN 代码经过修改以兼容 Android 9 及更新版本的 adb shell getevent 输出格式。

对于 RQ2，我们在 Nexus 4 虚拟机上进行录制实验，并将录制结果在小米 9 与 Pixel 2 XL 虚拟机上进行回放实验。

对于 RQ3，我们测量在小米 9 上对应用进行录制和回放所带来的运行时间与存储空间的额外开销。对于运行时间，我们比较在相同场景下不开启录制、使用本工具录制以及回放所使用的时间；对于存储空间，我们统计本工具对于不同时长录制的文件大小。

表 4.1: RQ1 中选取的测试应用以及其分别在相同设备上录制与回放的实验结果

应用名称 (版本)	分类	下载量	本工具	replaykit	RERAN
Accuweather(7.8.1-5-google)	天气	1 亿 +	9/9	1/3	3/9
Alipay*(10.2.20.7000)	理财	500 万 +	2/3	2/3	2/3
BBC News(5.18.0)	新闻杂志	1000 万 +	9/9	1/3	3/9
CNN(6.16)	新闻杂志	1000 万 +	6/9	1/3	3/9
Daylio(1.38.2)	生活时尚	1000 万 +	9/9	2/3	3/9
Dcoder, Compiler IDE(3.2.9)	教育	100 万 +	9/9	3/3	6/9
Drugs.com(2.11.1)	医疗	100 万 +	3/9	2/3	3/9
ESPN(6.43.0)	体育	5000 万 +	9/9	1/3	3/9
GoodRx(6.0.26)	医疗	1000 万 +	9/9	1/3	0/9
Huobi Global(6.2.7)	理财	100 万 +	9/9	1/3	0/9
Sketch(8.6.A.0.10)	艺术和设计	5000 万 +	6/9	1/3	0/9
Steam(2.3.12)	娱乐	5000 万 +	9/9	3/3	3/9
WeChat(7.0.21)	通讯	1 亿 +	9/9	2/3	3/9
ZhiHu*(7.11.0)	社交	100 万 +	0/3	0/3	0/3
总计			?	?	?

表 4.2: RQ2 中选取的测试应用、测试场景以及其分别在不同设备上录制与回放的实验结果

应用名称 (版本)	场景	下载量	Pixel 4 XL		小米 9	
			本工具	replaykit	本工具	replaykit
总计			?	?	?	

表 4.3: RQ3 中选取的测试应用、测试场景以及其不开启录制对比开启录制与回放的实验结果：运行时间

应用名称 (版本)	场景	未录制耗时 (sec.)	录制耗时 (sec.)	回放耗时 (sec.)

表 4.4: RQ3 中选取的测试应用、测试场景以及其不开启录制对比开启录制与回放的实验结果：文件大小

应用名称 (版本)	场景	录制文件大小 (KB)	录制时长 (sec.)	每秒平均大小 (KB/s)

第五章　局限性与未来展望

参考文献

[1] HU Y, AZIM T, NEAMTIU I. Versatile yet Lightweight Record-and-Replay for Android[J/OL]. SIGPLAN Not., 2015, 50(10): 349–366. https://doi.org/10.1145/2858965.2814320.

[2] GOMEZ L, NEAMTIU I, AZIM T, et al. RERAN: Timing- and Touch-Sensitive Record and Replay for Android[C] // ICSE '13: Proceedings of the 2013 International Conference on Software Engineering. [S.l.]: IEEE Press, 2013: 72–81.

[3] Sahin O, Aliyeva A, Mathavan H, et al. RANDR: Record and Replay for Android Applications via Targeted Runtime Instrumentation[C/OL] // 2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE). 2019: 128 – 138. http://dx.doi.org/10.1109/ASE.2019.00022.

[4] LAM W, WU Z, LI D, et al. Record and Replay for Android: Are We There yet in Industrial Cases?[C/OL] // ESEC/FSE 2017: Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering. New York, NY, USA: Association for Computing Machinery, 2017: 854–859. https://doi.org/10.1145/3106237.3117769.

[5] TENCENT. WeChat[EB/OL]. 2021. https://www.wechat.com.

[6] Frida[EB/OL]. 2021. https://frida.re.

[7] GUO J, LI S, LOU J-G, et al. Sara: Self-Replay Augmented Record and Replay for Android in Industrial Cases[C/OL] // ISSTA 2019: Proceedings of the 28th ACM SIGSOFT International Symposium on Software Testing and Analysis. New York, NY, USA: Association for Computing Machinery, 2019: 90–100. https://doi.org/10.1145/3293882.3330557.

[8] APPETIZERIO. replaykit[EB/OL]. 2018. https://github.com/appetizerio/replaykit.

致　谢

致谢

余添诚

2021 年 5 月