计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(10):240−247 [doi: 10.15888/j.cnki.csa.006597]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于控件路径的跨设备 UI 自动化测试方法①

侯　津[1,2], 顾乃杰[1,2], 丁世举[1,2], 杜云开[1,2]

[1](中国科学技术大学 计算机科学与技术学院, 合肥 230027)
[2](安徽省计算与通信软件重点实验室, 合肥 230027)
通讯作者: 顾乃杰, E-mail: gunj@ustc.edu.cn

摘　要: 随着移动应用的爆炸式增长, 如何高效、正确地进行 UI 自动化测试成为了一个重要问题. 传统自动化方法大多需手动编写测试脚本, 自动化程度更高的录制回放方法则普遍具有跨设备能力不足的问题, 而且现有断言机制已经不足以描述丰富的 UI 语义. 针对上述问题, 本文提出一种跨设备能力强且可以描述丰富 UI 语义的录制回放自动化测试方法. 该方法使用控件路径精确定位控件, 并结合跨设备 UI 自适应方法以提高跨设备能力; 通过提出两种新的断言机制以支持与数字排序和图片相关的 UI 语义. 在该方法基础上, 本文面向 Android 和 iOS 应用程序实现了一种自动化测试框架 RRF, 实验结果表明 RRF 的回放成功率比其他自动化测试工具更高.

关键词: UI 自动化测试; 跨设备; 控件路径; 断言

引用格式: 侯津,顾乃杰,丁世举,杜云开.基于控件路径的跨设备 UI 自动化测试方法.计算机系统应用,2018,27(10):240–247. http://www.c-s-a.org.cn/1003-3254/6597.html

UI Automating Test Method for Cross-Device Based on Widget Path

HOU Jin[1,2], GU Nai-Jie[1,2], DING Shi-Ju[1,2], DU Yun-Kai[1,2]

[1](School of Computer Science and Technology, University of Science and Technology of China, Hefei 230027, China)
[2](Anhui Provincial Key Laboratory of Computing and Communication Software, Hefei 230027, China)

Abstract: With the explosive growth of mobile applications, how to carry out UI automation testing efficiently and correctly becomes an important issue. Most of the traditional automated methods require developers to write test scripts manually, and another high-level testing method called “recording and playback” does not has capability of crossing devices. In addition, existing assertion mechanisms are generally not capable of describing the UI semantics completely. Due to those problems, this paper presents a new recording and playback method which is capable of crossing devices and describing the UI semantics. This method uses the widget path to precisely locate the widgets and employs the cross-device UI adaptive method to improve the capability of device-crossing. Furthermore, this study proposes two new assertion mechanisms to support UI semantics which check on number sorting and pictures. In addition, this study builds a prototype framework called RRF according to the proposed methods, and the experimental results show that RRF has a higher success rate of playback than other automated test tools.

Key words: UI automated testing; cross-device; widget path; assertion

1　引言

近年来, 智能手机和平板的普及程度日益提高, 其上的应用程序数量也急剧增加, 应用程序频繁的版本迭代导致大量测试资源消耗在回归测试中. UI (User Interface) 测试作为回归测试的重点, 其效率的提升将会直接影响测试成本. 因此, UI 测试的自动化已经成为

① 收稿时间: 2018-03-15; 修改时间: 2018-04-03, 2018-04-12; 采用时间: 2018-04-17; csa 在线出版时间: 2018-09-28

移动应用测试中一个重要的研究课题.

按照自动化程度的不同, 目前 UI 自动化测试方法可分为手动编写测试脚本和录制回放法两大类. 前者要求测试人员编写测试脚本, 测试成本较高. 后者录制脚本后进行回放, 又可细分为基于控件属性、基于坐标、基于图像匹配等方式, 但这些方式普遍存在跨设备能力较弱的问题, 并且只能进行简单的文本断言. 现有的 UI 自动化测试方法在面对录放设备屏幕分辨率差异较大、或控件缩放规则不同、或开发人员未赋予控件唯一属性时, 自动化回放成功率不高, 导致这种现象的主要原因在于这些自动化测试方法缺少唯一定位控件的有效方式, 从而同一控件在录放时定位至不同控件, 导致回放失败.

针对缺少唯一定位控件的有效方式而导致的跨设备能力不足及 UI 语义描述过于简单的问题, 本文提出一种基于控件路径的跨设备 UI 自动化测试方法, 在此基础上实现了针对 Android 应用和 iOS 应用的 UI 自动化测试框架 RRF (Record Replay Framework). 该方法使用了控件路径以唯一准确地定位控件, 以此实现跨设备脚本录制和回放, 并提出两种新的断言机制以支持与数字排序和图片相关的 UI 语义. 具体实现上, 该方法一方面通过用户操作坐标和 GUI (Graphical User Interface) 控件树生成控件路径并写入文件, 从而录制跨设备测试脚本; 另一方面进行文本、排序、图片断言处理或使用一般搜索算法结合跨设备 UI 自适应方法, 将控件路径信息转化成操作坐标和操作类型, 以生成手机事件进行回放.

本文的组织结构如下: 第 2 节简要介绍 UI 自动化测试的研究进展; 第 3 节详细叙述本文提出的基于控件路径的跨设备 UI 自动化测试方法; 第 4 节介绍该方法面向 Android 和 iOS 应用程序实现的框架 RRF; 第 5 节进行实验并分析结果; 第 6 节对本文进行总结.

2 相关工作

近年来, UI 自动化测试的自动化程度正逐渐提高, 测试方案由自动化程度较低的手动编写脚本, 发展到自动化程度较高的录制回放方式.

2.1 基于编写脚本的自动化方案

Robotium[1]是一套用于 Android 系统的自动化测试框架. 它通过重签名被测应用, 将测试脚本和被测应用运行在同一进程中, 进而抓取被测应用的 GUI 控件信息和驱动被测应用的运行. 它提供基于控件文本属性、控件唯一标识符等的控件定位方式, 并且支持原生控件、Web 控件等所有控件类型. 其缺点在于无法进行跨应用的测试. 为了支持跨应用测试, Android Software Development Kit (SDK) 提供了测试框架 UIAutomator[2], 它将测试脚本安装至设备中直接单独运行, 即可控制待测应用. 它支持跨应用测试, 但无法支持 Web 控件的测试. Appium[3]的出现克服了上述方法功能的不完整性, 它是一款集成的自动化测试框架, 提供 Android 和 iOS 客户端, 其中 iOS 客户端可以简单录制并且提供 GUI 控件树查看功能. Appium 通过在手机服务端集成 UIAutomator 和其他框架以支持跨应用和 Web 控件操作.

Robotium、UIAutomator、Appium 等自动化测试框架提供自动执行测试脚本的方法, 并且提供多种控件定位方式. 但它们都需要测试人员手动编写测试脚本, 并且测试人员需依靠其他工具查看控件属性等信息以编写脚本.

2.2 基于录制回放的自动化方案

鉴于手动编写测试脚本需耗费大量人力, 录制回放的方法被提出. 录制回放的测试方法包含基于图像匹配、基于坐标、基于控件属性等方式.

李昕宇等提出了一种基于图像匹配的移动应用自动化测试方法[4], 其工作主要针对手机软件中比较常见的文字、图片、控件、列表及网格等不同类型的区域, 建立基准图像库, 通过基于特征点匹配的图像匹配方法, 实现对手机界面显示结果的正确性评估. 这种方法虽然不需要用户编写测试脚本, 但只能进行界面测试, 无法进行功能测试.

随后出现基于坐标的录制回放工具, 例如 Jacareto[5]、MonkeyRunner[6]、Monkey[7]及 Pounder[8], 可以支持 UI 功能测试. 它们记录鼠标点击坐标和键盘事件, 在回放中, 使用捕获的信息创建新的触屏和键盘事件. 这些基于坐标的录制回放方法, 简单快速且很少需要测试人员干预. 但录放设备不同时, 同一个录制坐标在回放中可能定位到不同的控件, 从而导致回放操作和录制操作不一致. 因此这种方法跨设备能力不足, 并且不支持 UI 组件层次的断言. RERAN 是一种 Android 平台下的录制回放方法[9], 在该方法中, 用户对设备的操作直接通过底层的事件流捕获, 然后通过在设备的事件流中注入捕获的事件来实现回放过程. 这种方法类似于基于坐标的方法, 但它不仅可以捕获触

摸事件, 还可以捕获其它由传感器产生的事件. 然而, 由于它不能提供测试脚本和断言操作, 测试人员很难理解和编辑测试用例, 或检验 UI 组件的输出.

针对基于坐标的录制回放无法提供脚本修改及 UI 层次的断言等问题, Chien-Hung Liu 等提出一种 Android 平台下基于 GUI 控件树插桩的录制回放方法[10]. 该方法在视图层次结构树的根结点下面引入一个称为 interceptlayout 的模拟布局[11]. 其可以拦截从根视图分发的所有键和触摸事件, 然后生成基于 Robotium 的录制脚本, 直接使用 Robotium 即可进行回放. 但该方法需提供待测应用程序的源码, 且不能进行跨应用测试. 另外当录放设备屏幕分辨率差别较大而导致回放设备中的部分控件被屏幕遮挡的情况, 该方法也无法适用. 因此该方法仍存在跨设备能力有限的不足之处.

Kaasila 等提供了一个名为 Testdroid 的在线 Android 应用程序测试平台[12], 支持 UI 组件层次的断言且不需提供源码. 该平台通过跟踪与之交互的 UI 组件记录用户操作, 记录的操作被翻译成 Robotium 测试脚本, 可以与被测试应用程序一起上传到平台. 测试脚本被自动调度且并行地在一组可用的物理设备上执行, 然后通过平台可以访问跨多个设备的测试结果, 然而它不支持跨应用测试, 跨设备能力不足且只支持文本相关的 UI 语义.

3 基于控件路径的跨设备 GUI 测试方法

基于坐标、基于控件属性等录制回放测试方法, 由于在应对不同分辨率场景时精确定位控件的能力有限, 导致了测试脚本无法完全适用于其他设备, 因此跨设备能力差. 另外上述方法也存在断言机制简单的弊端. 在此背景下, 本节提出了基于控件路径的自动化测试方法以精确定位控件录制跨设备脚本, 以及提出了 UI 自适应方法以应对录放设备分辨率差距较大的场景; 另外还将介绍排序、图片两种断言机制.

3.1 控件路径

定义 1. 移动应用程序的 GUI 由图形用户界面对象 (控件) 组成, 一个或多个页面的控件形成的树形结构被定义为 GUI 控件树.

定义 2. 描述一个控件从 GUI 控件树根结点到该控件结点的绝对路径被定义为控件路径.

GUI 控件树包含了控件的一组固定的属性. 在 GUI 执行期间, 这些控件包含控件相应的信息, 如控件的 rect 属性, 它构成了一个矩形框用以描述控件在页面的位置和大小. 利用 GUI 控件树结合控件操作坐标信息即可生成控件路径. XPath (XML Path Language) 是 XML 路径语言, 它是一种用来表示 XML 文档中某部分位置的语言. 控件路径是 XPath 的一种表达形式用以精确描述该控件在 GUI 界面中的位置.图 1 为部分控件树属性示意图. 在此部分控件树中, 结点 d 的控件路径一种表示方式为/table[1]/cell[1]/text[1], 其中每个结点是一个控件, table、cell、text 表示控件 type 即类型, 1 表示该结点是父结点子树中第一个此类型的结点, 即该结点的 index(索引值), 也可通过 name 属性和 index 结合表示.

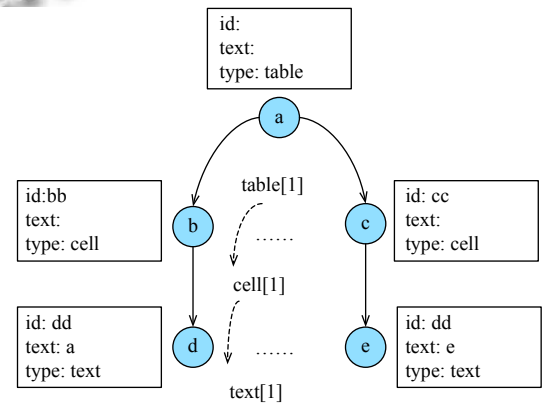

图 1 GUI 控件树及属性示意图

由图 1 可见, 控件的属性如 id 或 text 等不唯一确定. 它们可空可相同, 这些由开发人员设定. 因此采用控件属性定位控件的方法普遍存在找不到控件或定位多个控件的情况, 即使通过联合使用控件属性也仍然不能完全覆盖所有 GUI 场景. 而控件路径可以有效解决控件定位的问题以达到唯一准确定位控件的目的. 图 1 中, 每个控件都有唯一确定的控件路径与之对应. 并且当待测应用的版本稳定, 整个页面的架构和控件结构是确定的, 这是由开发代码控制的, 即同一应用的同一版本在某系统中的某状态下 GUI 控件树是确定的, 所以控件路径在搭载同一系统中的不同设备中也可唯一准确定位控件. 因此基于控件路径的方法具有良好的跨设备能力. 理论情况下移动设备上的应用只要显示为相同的页面布局且具有相同的 GUI 控件树, 此方法都适用. 如该方法适用于搭载同一版本 Android 系统的不同型号手机、平板或者搭载同一版本 iOS 系统的不同型号手机、平板.

3.2 跨设备脚本录制方法

由于基于控件路径的方法具有良好的跨设备能力, 因此跨设备脚本录制使用了控件路径作为控件定位的依据. 本节将介绍如何获取相关信息, 并将其转化成控件路径以录制成脚本.

跨设备脚本录制方法流程图见图 2. 首先通过事件处理算法处理监听信息, 以获取用户操作类型和坐标信息; 其中操作坐标信息用于生成控件路径, 操作类型用于回放的事件重构. 然后通过系统框架抓取手机当前状态的 GUI 控件树, 这里不会对应用进行任何修改; 由于不同状态下控件的属性会发生变化, GUI 控件树需要实时渲染以提供最新 GUI 控件树信息. 接着通过操坐标信息和 GUI 控件树进行控件定位并生成控件路径, 即使用坐标在 GUI 控件树中深度搜索包含该坐标的最小控件, 该控件即为用户操作控件; 记录该控件的控件路径和操作坐标在其的比例来更精确定位操作位置. 最后将所有信息写入文件, 即可生成跨设备测试脚本.

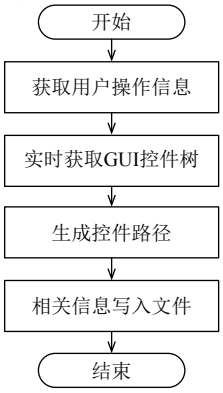

图 2 跨设备脚本录制流程图

算法 1. FindNode

Input: GUI 控件树 *nodes*, 操作坐标 *point*, 栈 *s*
Output: 控件定位结果

```
1 result←false;
2 for each child ∈ nodes.childrens do
3 if point∈child.rect then
4 result←result || FindNode(child, point, s);
5 if result || p∈nodes.rect then
6 s.push(nodes.type, nodes.index);
7 return true;
8 return false;
```

控件路径具体生成步骤见算法 1, 该算法使用操作坐标 p 在 GUI 控件树深度搜索, 查找符合条件的最小控件 (2–4 行). 待找到最小控件结点后将路径上所有结点的 type、index 入栈 (第 6 行). 深度遍历完成后, 出栈所有的结点 type 和 index 组合即可生成控件路径.

3.3 跨设备 UI 自适应

基于控件路径的测试方法支持大部分情况的跨设备回放. 但当设备间屏幕尺寸和分辨率差别较大, 待操作控件在录放设备中显示不同时, 仍可能存在回放失败的情况. 如图 3 在一个长屏手机 A 录制点击控件 b3 的脚本, 在短屏手机 B 中回放. 由于屏幕外的控件无法被操作, 而控件 b3 在短屏手机中未显示. 因此即使通过控件路径定位到该控件, 也无法回放该控件的操作. 此时跨设备 UI 自适应方法通过滑动屏幕, 重新渲染 GUI 界面来显示待操作控件至当前界面以解决这个问题.

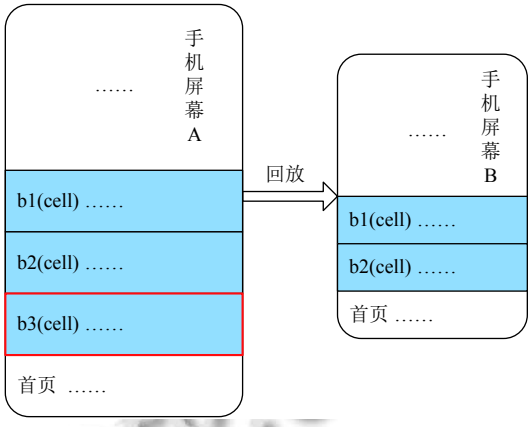

图 3 录放设备屏显示意图

跨设备 UI 自适应方法首先确定滑动的方向. 某状态 GUI 控件树见图 4, 控件 anc、a、b、c 是控件树的一部分. anc 是 a、b、c 的上层结点即父亲结点, 控件 b 显示在手机屏幕中而控件 a、c 在屏幕外. 下面结合图 4 来介绍鉴定滑动方向的方法. 假设 c 为待查找控件. 首先在屏幕内任意找到一个控件 b, 然后遍历 b 和 c 的祖先结点并找到最近公共祖先结点 anc. 由 anc 分别沿着控件 b、c 的子树向下遍历, 并比较两子树同层结点的 index, 若 c 结点的 index 较大, 则 c 在 b 的下方, 向上滑动, 否则向下滑动. 图中所示 a、b、c 在控件树的同层且控件 c 的 index 较大, 故 c 在 b 的下方, 此时通过上滑操作即可将 c 滑入屏幕内. 确定滑动方向后, 下一步需要确定滑动距离. 由于控件或自适应设备或保持不变, 单个滑动的最大距离不超过 *s*. 最大距离 *s* 根据公式 (1) 得出.

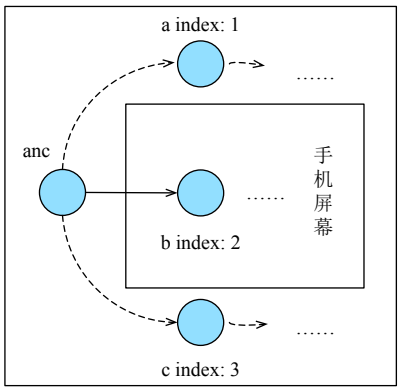

图 4 GUI 控件树简略图

$$s = \frac{h'}{h}H - H' \tag{1}$$

$$px = dp * (dpi/160) \tag{2}$$

h和h'分别是设备 A 录制和设备 B 回放中控件的高度, 当h'/h<1 时, 令h'/h=1, 此时认为控件保持不变即h'/h=1 所得的 s 值为该控件可滑动也就是缩放的最大距离. H和H'分别是录放设备屏幕的高度, 单位均为像素. 其中h'/h在 Android 平台下可由单位转换公式 (2) 转化为dpi'/dpi, 即回放与录制设备屏幕密度比. 计算出跨设备自适应的最大距离后, 每次滑动 s/4 距离, 并重新渲染 GUI 界面, 直到屏幕内显示出待查找控件.

3.4 断言机制

断言用来验证应用程序执行的正确性, 快速定位错误.

定义 3. 在 UI 自动化测试中, 基于 UI 语义的断言算法可以表示为一个三元组 $<C,P,R>$, 其中 $C=\{c_1,c_2,\cdots,c_n\}$ 是录制时 UI 属性数据集, $c_1,c_2,\cdots,c_n$ 分别是录制时各 UI 控件的属性数据; $R=\{r_1,r_2,\cdots,r_n\}$ 是回放时 UI 属性数据集, $r_1,r_2,\cdots,r_n$ 分别是回放时各 UI 控件的属性数据. $P=\{p_1,p_2,\cdots,p_n\}$是有穷的断言规则的集合.

定义 4. 对于$\forall p_i \in P$, 由$Assert(c_i,r_i)$表示控件i是否符合断言规则p_i. 若符合规则p_i, 则$Assert(c_i,r_i)=1$, 否则$Assert(c_i,r_i)=0$.

其中 P 包括文本断言、排序断言、图片断言. 在 UI 自动化测试中,$\forall p_i \in P$, 都有$Assert(c_i,r_i)=1$, 则未检出错误 (前提是加入的断言符合程序正确运行的规律).

文本断言首先对控件进行定位, 然后将控件的属性信息和录制信息比较以验证程序执行正确性. 在许多应用中网格或者表格控件提供一列或者一行数据的排序, 如股价排序、QQ 访问量排名等. 排序断言自动判断 UI 界面某行或某列数据是否有序, 以应对该测试场景. 排序断言主要在于解决如何获取应用程序一行或一列数据的问题, 然后判断该序列有序即可. 针对这个问题, 首先根据一行或一列的两个控件路径定位具体控件, 然后查找这两个控件的最近公共祖先结点, 沿着祖先结点往下搜索本行或本列的所有控件, 加入待排序集合. 同一行的结点控件路径的最后一层 index 不同, 由公共祖先结点到倒数第二层的所有结点皆具有相同的 index, 类似于相同的行号; 相对的同一列结点则是中间某层祖先结点的 index 不同, 其后的所有层结点 index 均相同. 获取排序集合具体算法见算法 2, 首先进行初始化, 由 XPathToNode() 定位到 XPath1、XPath2 所在控件结点, 并由 getAncestor() 返回两个结点的最近公共祖先结点 ancestor(1–3 行); 然后将 ancestor 入队列 nodes, 将 XPath1、XPath2 分割成单个结点信息, 将 ancestor 结点及后代结点入队列 (4–6 行); 结点总数不同时说明选中的两个结点是非同行或同列结点, 此时直接返回 (7–8 行). 两个结点从祖先结点的下一个结点开始分别往下遍历, 当两者当前结点的 index 不同, 若已经遍历到最后一层则是这两个结点属于同行结点, 直接将这层所有结点即 curNode 的所有孩子结点入队列 nodes 并返回结果; 若非最后一层说明两个结点属于同一列, 此时将这一层 (假设第 k 层) 的所有结点即 curNode 的所有孩子结点入队列 nodes, 其后两个结点每层 (第 $k+n$ 层) 祖先结点的 index 均相同, 出队列所有结点 curNode, 并将每一个 curNode 孩子结点中 index 与 nodes1 当前结点 index 相同的结点入队列 nodes; 最后队列 nodes 中所有结点即为待排序集合 (9–18 行).

算法 2. 排序集合获取

Input: GUI 控件树 *T*, 控件路径 *XPath1*, 控件路径 *XPath2*
Output: 待校验集合

1 *node1*←XPathToNode(*XPath1*);
2 *node2*←XPathToNode(*XPath2*);
3 *ancestor*←getAncestor(*node1*,*node2*);
4 *nodes*.Enqueue(*ancestor*);
5 *nodes1*.Enqueue(*ancestor* 到 *node1* 路径上的所有结点);
6 *nodes2*.Enqueue(*ancestor* 到 *node2* 路径上的所有结点);
7 **if** *nodes1.count*≠*nodes2.count* **then**
8 **return** null;

```
9 for i=1 to nodes.count–1 do
10 queueCount←nodes.count;
11 for j=1 to queueCount do
12 curNode←nodes.Dequeue();
13 if nodes1[i].index≠nodes2[i].index then
14 nodes.Enqueue(for node∈curNode.childrens);
15 else
16 tmp←curNode.childrens[nodes1[i].index];
17 nodes.Enqueue(tmp);
18 return nodes;
```

由于控件的某些属性由开发人员设定, 所以文本控件属性值存在为空的情况, 或者界面某部分不是一个控件如只是一个文本控件的部分, 而文本断言或排序断言需识别出控件, 然后对控件中的文本属性值进行校验. 所以在面对上述两种断言场景时, 上述断言无法使用. 针对上述情况, 基于图像匹配的图片断言被提出, 其提供用户截取一部分区域然后与回放时页面截图进行比较以提供自动校验 UI 界面显示结果的能力, 例如校验某个搜索按钮是否显示在界面上或某部分文字排列是否和录制时相同等. 由于该方法基于图片对比所以不要求截取区域是一个控件或必须有控件属性值. 具体方法是准备录制时截取的图片和回放当前页面截图. 然后使用 OpenCV 的模板匹配算法 matchTemplate 返回图片匹配结果. 由于在不同分辨率场景录放设备相同位置截取的区域可能存在偏差, 若使用相应坐标位置在回放页面中进行截图对比, 可能匹配失败. 而本方法用录制截图在整个回放页面截图中进行查找, 只要回放页面中显示出相同的布局则会查找成功, 所以可以应对不同分辨率场景. 对于图片中有多个匹配点的情况, 匹配成功个数作为结果返回. 通过上述图片断言方法来判断回放中 UI 界面显示的正确性.

4 测试框架 RRF 实现

RRF 是基于上述方法针对 Android 应用、iOS 应用实现的自动化测试框架. 该框架的设计框架如图 5.

图 5 中实线箭头经过路径代表跨设备脚本的录制. 录制由捕获手机事件开始, 这里 RRF 采用手机端捕获和电脑端捕获两种方法. 为了支持 Android 手机端录制的事件捕获,RRF 使用了 Android SDK 提供的 getevent 工具.getevent 用来读取/dev/input/event*设备文件. 当用户与 Android 应用程序交互时, 用户事件通过 Android 设备的传感器生成/dev/input/event *设备文件并发送至内核, 事件格式为 (时间戳 设备 类型 编号 值), 触屏手势被编码为上述格式的触摸屏事件流. 例如, 1494674903/dev/input/event1 0003 0035 0000013c 表示一个点触事件. 其中前三项分别对应时间戳、设备和触摸事件类型,0035 对应于事件的 x 位置, 0000013c(十六进制) 对应于屏幕的坐标 316(十进制). 高级手势操作通常涉及多个触摸屏事件. RRF 通过触摸屏事件处理算法对这些底层数据进行处理, 抽象出用户的各种高级手势操作和坐标. 对于电脑端录制的事件获取,RRF 则通过不断截取手机屏幕图片, 映射至电脑后在电脑端操作手机屏幕; 然后监听鼠键事件, 并生成手机操作事件信息. 下一步通过不断重新渲染 GUI 界面, 由 Android 或 iOS 系统框架实时获取 GUI 控件树. 由上述步骤获取两个输入后, 利用第 3 节的控件路径生成算法进行控件定位并生成控件路径, 然后将控件路径及其他描述信息, 包括断言、录制截图等写入文件, 生成 XML 测试脚本, 到此跨设备脚本的录制过程结束. 此外 RRF 将测试用例中的数据信息和逻辑操作分存储成数据文件、逻辑文件. 当逻辑操作相同时, 只需要改动数据文件即可生成新的测试脚本, 以减少了录制的重复工作.

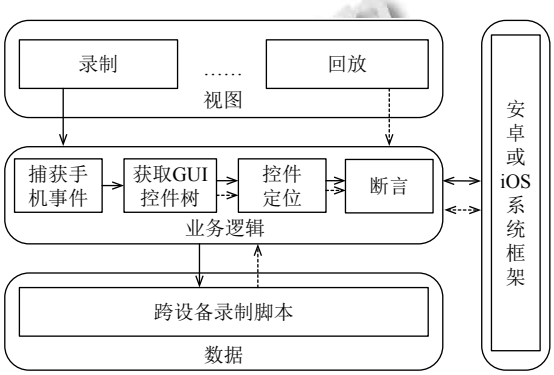

图 5 RRF 设计框架

图 5 中虚线箭头经过路径表示跨设备脚本的回放过程. 回放的核心模块是使用控件路径进行控件定位或断言. 这里的控件定位是将录制脚本翻译成待操作控件, 精确至坐标. 该模块有两个输入, 分别是当前 GUI 控件树和录制脚本信息.RRF 通过控件路径在 GUI 控件树查找控件, 面对 3.3 节所述场景通过 UI 适应方法调整 UI 显示, 然后获取控件坐标信息. 随后通过 Android 平台的 UIAutomator 框架、iOS 平台的

WebDriverAgent[13]框架等执行手机指令. 根据 3.4 节实现断言, 即控件定位后直接验证字符串、数字断言; 或利用算法 2 获取待排序集合, 实现排序断言; 或利用图片比对算法返回录制截取图片在当前页面匹配个数, 实现图片断言; 另外为了提高效率,RRF 支持同一脚本安装至多个设备同时回放, 以提高 RRF 测试效率.

5 实验及分析

5.1 实验环境

待测设备包括一台红米 A2 搭载 Android 4.2.4 系统, 辨率为 720*1280; 一台华为荣耀 6 搭载 Android 4.4.2 系统, 分辨率为 720*1184; 一台小米 4 搭载 Android 4.0.4 系统, 分辨率为 1920*1080. 待测软件为国泰君安君宏 8.8.5, QQv7.5.5.

由于 iOS 平台下录制回放工具较少, 且 iOS 平台下的 RRF 与 Android 平台的基本一致. 本实验只针对两款 Android 平台的录制回放工具和 Android 平台下的 RRF 进行对比实验, 这两款工具分别是基于坐标的 MonkeyRunner 和基于控件属性的 iTestin[14]. 在回放成功率实验中, 回放步骤和录制完全一致则为成功, 每个测试用例各录制回放 50 次以统计回放成功率. 在断言实验中, 程序断言结果和预期结果一致则为成功, 每个测试用例同样各录制回放 50 次来记录正确率.

表 1 测试用例介绍及实验结果

测试用例	测试用例特征			录放成功率 (%)		
	跨应用	录制设备	回放设备	MonkeyRunner	iTestin	RRF
登录	否	华为荣耀 6	华为荣耀 6	100	90	98
登录	否	华为荣耀 6	小米 4	0	40	94
股票搜索	否	华为荣耀 6	红米	0	46	96
注册	是	华为荣耀 6	华为荣耀 6	100	0	100

5.2 录制回放成功率实验

本实验 3 个测试用例. 每个测试用例录制多个脚本进行回放. 登录和股票搜索测试仅对待测软件进行操作, 操作手势包括点击、滑动等, 操作控件的类型包含原生、混合、WebView 所有的控件类型; 注册测试为跨应用测试, 涉及待测软件和信息两个应用, 操作手势仅包含点击操作. 实验首先根据测试用例的操作序列, 在选定录制设备上进行脚本录制; 然后在回放设备中回放录制脚本; 最后统计录制回放成功率.

针对 MonkeyRunner、iTestin 失败的现象, 经过分析录制脚本、相关日志等记录, 基于坐标的 MonkeyRunner 在录放时相同坐标会定位到不同的控件从而回放失败率较高, 而控件路径则可以在具有相同控件树的不同设备中唯一定位控件, 因此基于控件路径的方法具有比基于坐标的测试方法具有更好的跨设备能力. 另外由于基于坐标的方式不能识别出控件, 所以也不支持上文所述的 UI 组件层次上的断言. 而 iTestin 使用基于组件属性组合的方式进行录制, 由于控件属性组合存在不能唯一定位控件的情况, 从而其在录放设备中控件定位不精确, 所以跨设备能力相对较弱, 而 RRF 使用基于控件路径的方法可以唯一精确定位控件且在回放时未显出待查找控件的情况下通过 UI 自适应方法进行控件查找以操作控件, 所以具有较好的跨设备能力. 另外由于 iTestin 基于 Robotium 框架, 所以它不具有跨应用测试的能力. 对于 RRF 失败的测试结果, 分析其原因是由于测试用例受网络的影响数据加载时长不定, 影响滑动的执行结果的准确性, 进而影响测试脚本的后续执行, 最终导致回放失败. 总体而言, RRF 提供了一种可以跨设备的黑盒自动化测试方法以进行 UI 功能测试或回归测试, 其相较于基于图片的方式具有较广的应用场景, 较于基于坐标或组件属性的方式具有较好的跨设备能力, 其在跨设备支持上达到了很好的效果, 录制回放成功率也高于传统的框架.

实验结果见表 1. 从中可以看出在实验的测试用例中 RRF 完全支持跨应用测试, 且跨设备测试成功率达 90% 以上. 作为对比, MonkeyRunner 不具有跨设备能力, iTestin 跨设备能力较弱且不能跨应用.

5.3 断言实验

由于现有的自动化测试框架不具有自动排序和图片断言能力, 第二节所述基于组件的方式只具有文本断言能力. 本实验仅对 RRF 的断言成功率进行测试.

本实验首先在录制过程中对涨幅和最新价加入排序断言或截取图片片段加入图片断言, 然后回放并统计回放结果, 最后根据待测软件内容人工设置断言的预期结果, 并计算断言成功率. 其中图片断言预期结果为成功匹配图片个数.

由表 2 可见, 本框架支持排序、图片断言, 且正确率高于 90%. 对于图片断言失败的测试结果, 分析原因有以下两点. 其一是因为截取图片为动态变化图片, 导致回放时录制截图恰好不在当前页面内而无法成功匹配录制截图. 其二是网络加载延时而导致当前界面未完全加载就执行图片匹配算法, 进而导致匹配失败. 针对第一点由于 App 情况不可控导致的断言失败情况, 可通过测试人员人工排查. 针对第二点失败情况,RRF 提供用户添加延时以等待图片加载完成.

表 2 断言测试结果

测试用例	录制设备	回放设备	预期结果	断言成功率 (%)
涨幅	红米	红米	有序	100
涨幅	红米	米 4	无序	100
最新价	红米	红米	有序	100
最新价	红米	米 4	无序	100
图片 1	红米	红米	2	100
图片 1	红米	米 4	2	90
图片 2	红米	米 4	1	94

6 结论

本文提出了一种基于控件路径的跨设备 UI 自动化测试方法, 并实现了针对 Android 和 iOS 应用程序的框架 RRF. 这种测试方法解决了编写和管理测试脚本困难的问题, 同时也解决了现有工具普遍跨设备能力差和断言简单的问题.RRF 实现跨设备、跨应用的录制回放, 支持多种断言场景, 并且不需对被测应用程序有任何修改, 另外还支持多设备同时回放, 可以很大程度上提供测试人员的工作效率. 在未来的工作中, 将添加测试工具对手机上各种传感器 (例如加速度计、指南针、GPS 等) 的支持, 以达到支持更多 App 自动化测试的目的.

参考文献

1 Amalfitano D, Fasolino AR, Tramontana P, *et al*. A toolset for GUI testing of android applications. Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM). Trento, Italy. 2012. 650–653. [doi: 10.1109/ICSM.2012.6405345]

2 Anbunathan R, Basu A. An event based test automation framework for Android mobiles. Proceedings of 2014 International Conference on Contemporary Computing and Informatics (IC3I). Mysore, India. 2014. 76–79. [doi: 10.1109/IC3I.2014.7019585]

3 Tao CQ, Gao J. Building a model-based GUI test automation system for mobile applications. International Journal of Software Engineering and Knowledge Engineering, 2016, (9-10): 1605–1615. [doi: 10.1142/S0218194016710042]

4 李昕宇, 侯春萍, 王宝亮, 等. 基于图像匹配的移动应用自动化测试方法研究. 计算机工程与应用, 2016, 52(13): 43–47.

5 Herding D, Spannagel C. Jacareto. https://sourceforge.net/projects/jacareto/. [2014-06-03].

6 Android Developers. MonkeyRunner. https://developer.android.com/studio/test/monkeyrunner/MonkeyRunner.html. [2017-05-15].

7 Hu CX, Neamtiu N. Automating GUI testing for Android applications. Proceedings of the 6th International Workshop on Automation of Software Test. Waikiki, HI, USA, 2011: 77–83. [doi: 10.1145/1982595.1982612]

8 Matthew P. Pounder Java GUI testing utility. https://sourceforge.net/projects/pounder/. [2013-04-09].

9 Gomez L, Neamtiu I, Azim T, *et al*. RERAN: Timing- and touch-sensitive record and replay for android. Proceedings of the 35th International Conference on Software Engineering (ICSE). San Francisco, CA, USA. 2013. 72–81. [doi: 10.1109/ICSE.2013.6606553]

10 Liu CC, Lu CY, Cheng SJ, *et al*. Capture-replay testing for android applications. Proceedings of 2014 International Symposium on Computer, Consumer and Control. Taichung, China. 2014. 1129–1132. [doi: 10.1109/IS3C.2014.293]

11 Android Developers. Layouts. https://developer.android.google.cn/guide/topics/ui/declaring-layout.html?hl=zh-cn.

12 Kaasila J, Ferreira D, Kostakos V, *et al*. Testdroid: Automated remote UI testing on android. Proceedings of the 11th International Conference on Mobile and Ubiquitous Multimedia. Ulm, Germany. 2012. 1–4. [doi: 10.1145/2406367.2406402]

13 Mercieca M. WebDriverAgent: Getting started with automated iOS testing. https://www.mutuallyhuman.com/blog/2017/04/20/webdriveragent-getting-started-with-automated-ios-testing. [2017-08-20].

14 Android 免费自动测试工具 iTestin. https://www.oschina.net/p/itestin. [2012-08-10].