规则驱动的 Android 应用 DFS 测试技术

叶 佳[1] 葛红军[2] 曹 春[2] 朱 晋[1] 张 营[2]
(中兴通讯股份有限公司终端事业部 南京 210012)[1] (南京大学计算机科学与技术系 南京 210046)[2]

摘 要 GUI 自动化测试是 Android 应用研究领域的重要组成部分,针对 Android 应用的 GUI 测试技术得到了广泛的研究。其中,基于 DFS 算法的 GUI 遍历测试技术得到了广泛的应用。然而,现有的 DFS 测试技术却仍然具有效率低下、覆盖率较低的问题。文中提出了结合外部预定义规则来驱动 DFS 自动化遍历的改进方法,以提高 DFS 自动化遍历的效率和覆盖率;基于规则驱动的改进方法实现了 RDTA 测试工具,进行了与 Monkey 以及无规则驱动下的 DFS 的对比实验,验证了该方法的有效性。

关键词 Android 测试,DFS,规则驱动,测试效率,覆盖率
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.09.015

Rule-driven DFS Testing Technology for Android Application

YE Jia[1] GE Hong-jun[2] CAO Chun[2] ZHU Jin[1] ZHANG Ying[2]
(Terminal Business Division, ZTE Corporation, Nanjing 210012, China)[1]
(Department of Computer Science and Technology, Nanjing University, Nanjing 210046, China)[2]

Abstract Automated GUI testing is the important part of Android application research field. Several technologies for automated Android GUI testing have attracted wide attention. Testing technology based on DFS exploration has been extensively used among them. However, the existing DFS testing technology is still inefficient and has low testing coverage. This paper proposed an improved approach by driving the DFS automated exploration with external predefined rules to improve the efficiency and coverage. A testing tool called RDTA based on the proposed approach was implemented and the performance of RDTA was evaluated by comparing to Monkey and original DFS without rules. The result verifies the effectiveness of the approach.

Keywords Android testing, DFS, Rule-driven, Test efficiency, Coverage

1 引言

近年来,智能手机在人们的日常生活中日益普及。根据 IDC 的调查,Android 设备占据了约 85% 的市场份额,是目前主流的智能手机平台。然而,由于 Android 版本和设备型号的多样性、Android 应用快速的更新迭代周期以及手工测试的效率较低,很多应用往往在没有被充分测试的情况下被发布出来,导致其中很可能隐藏着异常错误(Bug)。

近年来,为提高 Android 应用测试的效率和应用软件的可信水平[1],自动化测试技术得到了广泛的研究。根据自动化测试技术遍历应用 GUI 的不同策略[2],可以将测试工具分为以下几类:1)随机探索策略,测试工具如 Monkey 和 Dynodroid[3],它们在测试过程选择的探索策略是随机算法,其目的往往不是系统化地对应用进行测试,而是通过产生大量的随机输入序列,对应用进行压力测试;2)基于模型的探索策略,测试工具包括 GUIRipper[4],A3E-Depth-First[5] 等,它们在自动化遍历测试过程中对应用的 GUI 状态和状态的转移进行建模[6-7],最终生成测试过程的状态转移跳转的有向图或者有限状态机,并且记录测试的路径作为脚本;3)系统的探索策略,测试工具如 Evodroid[8] 和 ACTEve,这一类测试工具采用符号执行[9]、演化算法等方式来引导探索的路径,相比于随机测试,它能探索到更深入的路径分支。

基于模型的自动化测试算法的应用较为广泛,文献[4-5,10-11]中都是使用了 DFS 算法作为测试策略,但是随着应用的复杂性越来越高,其在测试过程中会遇到状态爆炸的问题,从而导致最终的效率较低。文献[12]提出自动化的测试过程应该在用户指导下提高测试效果。

在测试实践中,测试人员往往还通过编写测试脚本的方式进行 Android 应用测试[13],测试工具包括 Android 原生的 Instrumentation 框架以及 MonkeyRunner 工具。编写脚本的方式能够很好地利用测试人员对应用的先验知识,约束测试的路径,达到定向测试的效果。但是该方式会使测试人员的工作量较大,而且随着界面的变化,脚本的复用率低,需要重新编写新的脚本。

文中提出了一种结合 DFS 自动化测试与规则的测试方式。本文将详细讨论如何不通过测试脚本,而是通过定义外

到稿日期:2017-12-15 返修日期:2018-01-15 本文受中兴通讯研究基金资助。
叶 佳 男,硕士,主要研究领域为移动终端产品的软件研发与测试技术;葛红军 男,硕士,主要研究领域为安卓 GUI 智能自动化测试技术,E-mail:ghj_nju@163.com(通信作者);曹 春 男,博士,副教授,主要研究领域为软件工程、软件测试与分析等;朱 晋 男,硕士,主要研究领域为移动终端软件自动化测试;张 营 男,博士,主要研究领域为软件工程、云计算平台等。

部规则的方式，使得测试人员的先验知识可以驱动和约束测试过程，提高测试的效率和覆盖率。

2 Android 应用的 DFS 测试

2.1 相关概念

1)活动(Activity)：在一个 Android 应用中，Activity 是最重要的组件。Activity 为应用提供了一个屏幕，其中包含的一些界面组件，可以监听、处理用户的事件并做出响应。在 Android 测试中，Activity 的覆盖率通常衡量对整个应用所有的测试的覆盖率。

2)界面组件(Widget)：每一个界面组件都是组成 Android 的一个界面元素，都是可视化且可用来与用户交互的组件，例如 ImageView，TextView，ListView，LinearLayout 等。界面组件元素之间构成了一个分层的树形结构。

3)驱动事件(Event)：在不同的界面组件上可以根据用户的操作(Action)产生不同的响应，在界面组件上可以进行的例如点击、长按、滑动、文本输入等操作都是 Android 的 GUI 测试中的驱动事件。

4)状态(State)：即 Activity 界面中 GUI 的所有组件的状态集合，用户触发的驱动事件会引起界面组件变化以及 Activity 间的跳转，即引起状态的变化。在 Android 的 GUI 测试中，状态的变化是指 GUI 界面的变化。

2.2 DFS 的测试模型

DFS 算法是一种基于模型的探索策略。DFS 的自动化测试过程的主要思想是用深度优先和回溯的策略来选择驱动事件，在测试过程中进行 Android 状态的转移，形成一个状态转移图或有限状态机。在测试流程中，算法的核心流程包括驱动事件的获取、驱动事件策略的选择、状态转移图的构造以及状态的回溯等。

首先是驱动事件的获取。在 Android 的 GUI 测试中，每个状态都有自己的界面，界面中包含了众多的界面元素，而每个界面元素都包含了一个或者若干个可以触发的事件，如在 TextView 上可以触发点击事件，在 EditText 上可以触发输入文本的事件和长点击事件，在 ListView 上可以触发滑动事件。在每个状态下获取该界面的所有 GUI 元素，并且遍历 GUI 的所有元素，获取所有的驱动事件，构成驱动事件的集合 EventSet。

其次是选择驱动事件的策略。在现有的 DFS 算法中，根据当前获取到的驱动事件的全集 EventSet，选取第一个尚未被探索过的驱动事件，直到该状态下的所有驱动事件都被探索结束。其中，有两种情况：1)在触发驱动事件之后，会进入到一个新的未被探索过的状态，根据 DFS 算法的原理会深度优先探索该新的状态；2)在该状态下的驱动都被探索结束，无需继续探索，或者触发驱动事件之后进入一个曾经探索过的状态，但需要进行回溯。在 DFS 的探索算法中，回溯是一个十分重要的过程，即将当前状态回溯到一个有效的祖先节点上。若状态 A 是状态 B 的祖先状态，则从状态 B 回溯到状态 A 需要重启应用回到应用的初始状态即根状态，然后执行初始状态到状态 A 之间的所有驱动事件的序列，到达状态 A。

2.3 DFS 的测试中的问题

DFS 测试算法在理论上是遍历整个应用的所有状态下的所有驱动事件，但在实践过程中发现，该策略往往会使效率低下，测试的覆盖率难以达到很好的效果。其根本原因在于：在驱动事件的选择过程中，依次遍历所有的驱动事件，而在整个测试流程中，测试的时间是有限的，测试的状态又是复杂且易变的，现在市场上主流的应用的功能都很丰富，包含的界面数量以及界面的复杂性快速增加，在有限的时间内依次遍历所有的驱动事件已经不再是最有效、合理的选择。在 Android 测试过程中，由于界面元素的不断变化和驱动事件带来的状态转移，测试过程将会不断产生新的状态，构成无限多的测试状态，导致探索路径爆炸[14]。

其次，驱动事件的集合将包含很多冗余事件，如图 1 所示的 *TextView*“美食”和 *ImageView* 图片在触发点击事件之后，所引起的测试状态的变化是相同的，定义这样的驱动事件是冗余的驱动事件。其一方面导致无效的状态转移，使测试过程的重复效率变低；另一方面是测试状态进入一个曾经探索过的状态，并在该状态需要进行回溯。在 Android 测试算法中回溯需要重启应用，将应用的状态从根节点执行到需要回溯到的目标节点的路径，回溯重启应用是一个十分耗时的过程，在测试过程中应该尽量避免低效的回溯[14]。

图 1 冗余驱动事件

Fig. 1 Redundant driven events

最后，针对 DFS 遍历过程中的驱动事件，例如文本输入的驱动事件，许多安卓自动化测试研究工作都回避了触发文本输入的驱动事件，或者是选择输入随机字符串的方式，希望能触发文本输入事件中可能存在的 Bug。但是在实际测试过程中，许多文本输入事件需要特定的有效信息才能促成有效的 Activity 或者状态的跳转，如输入注册邮箱。现在 Android 市场上的很多流行软件在未登录状态下能遍历到的 Activity 只是该应用所有 Activity 的很少部分。

3 规则驱动下的 Android 测试

3.1 规则驱动的测试模型

测试 DFS 算法在测试过程中存在着效率和覆盖率较低的问题，根据 2.3 节的分析发现，依次遍历所有的驱动事件以及测试过程中的大量回溯过程是其根本原因。在测试的运行实践中，DFS 的自动化探索过程对测试的应用是无先验知识的，对应用的遍历测试是试探性的；而导致低效的探索过程和回溯过程，在有先验知识的指导下将会得到避免。测试人员在特定的界面对于该界面需要测试的界面元素、需要避免测试的界面元素以及对界面元素测试的优先级等往往有更好的理解；测试过程中 DFS 自动遍历算法时难以处理的文本输入，对于测试人员而言可能是简单的操作。如果通过将测试人员的先验知识注入到自动化测试过程中，约束和限制 DFS 依次遍历所有驱动事件的方式，将会达到对整个测试的 DFS 树剪枝的作用，避免了很多无效、冗余的驱动事件以及十分耗

时的回溯过程和重启应用。

算法 1　规则驱动的 DFS 的 GUI 测试算法

```
输入:当前状态 currentState,上一个状态 parentState
输出:深度优先搜索树 DFSTree
1. set widgetTree=GetUIElement(currentState); //获取当前 State 的界面元素集合
2. set events=GetEventsByRuleEngine(widgetTree) //规则引擎根据当前界面元素生成驱动事件集合
3. repeat
4.    set event=GetFirstUnfiredEvent(events); //获取第一个未被执行的 event
5.    Fire(event); //执行 event
6.    set newState=GetCurrentState(); //获取当前 State
7.    if IsLeaf(newState) then //判断该 State 是否需要继续探索
8.       BackTo(currentState); //回溯到上一状态
9.       continue;
10.   else
11.      DFSTree(newState,currentState); //递归探索子节点
12.   end if
13. until AllFired(events) //直到该 State 所有 Event 已经执行完毕
14. if parentState==NULL then //判断是否为根节点
15.   return DFSTree; //测试结束,返回生成的 DFSTree
16. else
17.   BackTo(parentState); //回溯到上一个状态
18.   return NULL;
19. endif
```

因此,对于 DFS 的自动化遍历过程,需要提供一种用户干预的机制,用户通过配置外部规则的方式将其先验知识注入到自动化的测试过程中,DFS 的测试遍历过程将不会是完全不受限制的,而是受规则约束的,即规则驱动遍历测试。规则是在生成和选择驱动事件时,约束所有驱动事件的集合或者对驱动事件进行定制。

3.2　规则的内涵

规则的作用是将测试人员的先验知识用规则的方式来驱动测试的自动化过程,因此规则的目标是针对当前的 Activity 下的界面元素和驱动事件的集合,通过规则引擎的驱动,提高 DFS 测试的效率,避免无效的驱动事件,优先选择有效的、能引起状态转移的驱动事件,以较低优先级触发无效的驱动事件,从而提高有限时间内测试的覆盖率。

DFS 默认的遍历策略是依次遍历所有的驱动事件,规则针对"依次""所有"和驱动事件 3 个方面。通过规则从驱动事件的全集合中选择一个有效的子集合,其对驱动事件的优先级和实际操作(如点击、滑动、文本输入内容等)有一定的约束,根据测试的实践过程和测试人员的先验知识,测试工具往往会尽快触发能够引起状态转移的驱动事件,期望能在有限的测试时间内达到更高的覆盖率。规则的语义包含以下若干方面:1)界面元素的子集合,规则定义的单元,可以是某个特定的界面元素或者某一类相似的界面元素;2)子集合的遍历方式,该集合下的界面元素的个数可能很多,规则可以指定遍历的方式和限制,如指定顺序遍历、逆序遍历、随机遍历和遍历次数限制等,如果限制遍历次数则可以避免状态爆炸的问题;3)子集合上的操作事件,通过指定操作事件,可以更好地进行测试过程,如对于文本输入的界面元素,可以指定输入正确的文本,对于需要滑动的界面元素也可以指定滑动方向;4)子集合的优先级,优先级是相对于其他子集合的,优先级可以使引起状态转移和 Activity 跳转的驱动事件优先被触发,而低效冗余的驱动事件则被完全剔除或者优先级很低。

4　规则驱动的实现

4.1　规则的表示

规则的实现方式是一个外部定义规则的文件以及在选择驱动事件时的对应规则验证引擎。每条规则都是一个五元组〈activity〉〈selector〉〈modifier〉〈action〉〈priority〉,其中 activity 表示规则所约束的活动名;selector 表示界面元素子集合的选择器;modifier 表示对当前选择器的修饰器;action 表示对界面的元素的具体操作,对界面元素的操作即驱动事件;priority 表示在当前状态下该选择器的界面元素驱动事件在所有驱动事件集合中的优先级。

活动名使一个应用能够定位到特定界面的信息,如〈me.ele.Launcher〉表示"饿了么"应用的主界面的 Activity。

选择器的作用是从测试过程当前状态的界面元素的集合中选择出需要制定规则的子集合,该集合中的元素可以是某一个特定的界面元素,其在当前界面的所有元素集合中是唯一的,如在图 1 中有"美食"字样的 *TextView* 是唯一的;该元素也可以是某一类界面元素,这一类元素具有某些共性,如 *class* 都是 *ImageView*。界面元素是我们规则约束的主体,被用于定位特定的 Widget 元素在 Android 的 GUI 层次结构树上位置的信息。

修饰器的作用是进一步约束该界面元素的子集合上的遍历方式(见表 1),如图 2 所示,对于一个列表会有很多列表项,而每个列表项的遍历过程是相对等价的,如果遍历所有的列表项,则会耗费大量的时间和重复遍历旧的状态,因此可以通过选择遍历的策略和次数限制选择若干个列表项进行抽样测试。

表 1　修饰器

Table 1　Modifier

Modifier	规则语义
Random(value)	随机遍历
TopFirst(value)	顺序遍历
BottomFirst(value)	逆序遍历

注:value 表示遍历的次数限制

图 2　列表视图

Fig. 2　ListView

操作器的作用是约束选择器中界面元素集合的操作，其可选值可以是表 2 中的任意值，对界面元素上的操作即构成了特定的驱动事件，如对于 *TextInput* 操作，可以自定义输入的文本信息。

表 2 操作器

Table 2 Action

Action	操作语义
Click	点击
LongClick	长点击
Swipe Up/Down	向上/下滑动
Swipe Left/Right	向左/右滑动
TextInput(Value)	输入文本域

优先级用于设置界面元素子集合的优先级，默认的优先级为 0，优先级大于 0 的界面元素表示被拉入白名单，其上的驱动事件将会优先被触发；而优先级的值可以设定为－1，表示该类界面元素被拉入黑名单，该类界面元素上的驱动事件是冗余的，往往会导致进入旧的状态，从而引起回溯。

4.2 选择器的实现

在 Android 的 GUI 模型中，每个 Activity 的界面结构都是一棵分层的树结构，在界面组件的外层一般都存在 LinearLayout，FrameLayout 等布局元素，如图 1 中的 *ImageView* 和 *TextView* 均被包含在 *LinearLayout* 中。

UIAutomatorViewer 是 Android SDK 包中原生的开源工具，可以获取界面的详细信息，为用户提供一种查看当前界面布局、控件属性的工具，用户可以获取当前界面层次结构树的信息，并保存在 XML 文件中。

每个界面元素包括布局元素和其他界面元素，它们都是界面树中的节点，节点信息包含了该界面元素的所有属性信息，所有的属性信息既可以用来标识一个元素的特性，也可以根据其功能性属性来生成特定的驱动事件。标识性属性包括 text，resource-id，class，package，contect-desc 等；功能性属性包括 clickable，long-clickable，checkable 等。

Selector 选择器的实现是基于控件的属性信息，以及 GUI 树的祖先后代关系、兄弟关系等。利用控件自身的属性信息可以包含如图 3 所示的 XML 中所有的详细信息，如 class 和 text 信息；利用 GUI 树的关系描述可以有效地利用节点与节点之间的关系，如图 1 中的 *TextView* 节点和 *ImageView* 节点都是 *LinearLayout* 的子节点，而且 *TextView* 和 *ImageView* 又是兄弟关系。

```
〈!－父节点－－〉
〈node index=“0” text=“父节点” resource- id=“android:id/decor_content_
 parent”
  class=“android. view. ViewGroup”
  package=“com. zte. heartyservice”
  content-desc=“”
  clickable=〈false”〉
    〈!－子节点－－〉
    〈node index="0" text=“子节点 1”
    tresource-id=“android:id/bar_con tainer” /〉
    〈node index=“1” text=“子节点 2”
    resource-id=“android:id/content”/〉
〈/node〉
```

图 3 XML GUI 树

Fig. 3 XML GUI tree

在 Selector 选择器中，需要在保存界面层次树的 XML 文件中利用元素的属性和元素之间的关系来选择元素的集合。XPath 语言是一门在 XML 文档中查找信息的语言，可以通过元素和属性对节点进行选择。对应选择器中，搜索的节点是界面的 Widget 元素，而搜索的空间是关于界面的层次结构树的 XML 文件。

XPath 的路径表达式可以支持很多谓词、通配符和运算符，因此它可以很方便地定位到需要的节点，而且 XPath 适合表示界面组件的层次树结构的关系。图 4 给出了 XPath 路径选择表达式，selector1 表示选取所有 class 属性为 *TextView* 的界面元素；selector2 表示选取 *text* 内容为美食的界面组件；selector3 表示选取父节点为列表 ListView 下的 *ImageView* 界面元素，利用了节点之间的层次关系。

```
{“selector1”:“//node[@class=‘android. widget. TextView’]”}
{“selector2”:“//node[@text=‘美食’]”}
{“selector3”:“//node[@class=‘android. widget. ListView’]//node
[@class=‘android. widget. ImageView’]”}
```

图 4 选择器示例

Fig. 4 Examples of selector

4.3 规则验证引擎的实现

规则引擎是实现外部规则驱动自动化测试过程的引擎。图 5 是 DFS 自动化测试在有规则驱动和无规则驱动下的流程。默认情况下的 DFS 会根据当前界面的 GUI 树生成全部的驱动事件集合，该集合在没有规则的约束下，会有很多冗余的驱动事件，且没有优先级关系，只能依次遍历所有的驱动事件。如算法 1 所述，规则驱动引擎根据当前的界面元素，通过外部规则的选择器选出子集合，对子集合上的元素定义了遍历的顺序、驱动事件的具体操作，以及该子集合在全集合中的优先级。通过规则驱动的测试的驱动事件，是有选择性的、有优先级的驱动事件集合。

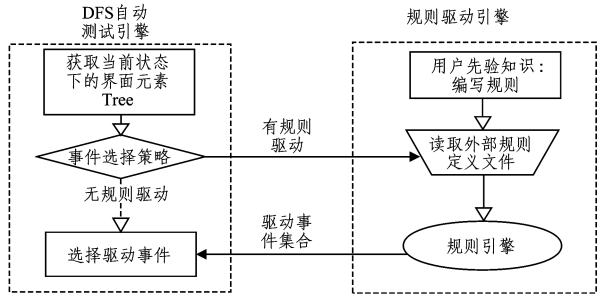

图 5 规则驱动模型

Fig. 5 Rule-driven model

5 实验结果分析

基于本文提出的算法，设计测试工具 RDTA(Rule-driven Testing Platform for Android)，并且在真实的 Android 应用上进行了实验验证。

使用的实验设备是 Nexus 5，Android 版本为 6.0.1，选取 12 个来自 Android 市场的实际应用。对每个应用进行对比实验，每组实验的时间限制为 1 h，实验中对比的测试技术分别是 Monkey 随机测试、无规则驱动的 DFS 测试技术(以下

简称 DFS)和有规则驱动的 DFS 测试(以下简称 RDTA)。根据文献[7]中对现有测试技术的实验对比,使用 Monkey 随机测试的通用性和效果较好,具有对比性。而通过对比是否有规则驱动的 DFS 遍历,希望解决以下问题:1)规则驱动的 DFS 遍历测试是否能提高测试的覆盖率和效率;2)规则驱动能够提高测试覆盖率和效率的根本原因;3)编写外部规则的易用性。

对于问题 1),分析表 3 中实验结果。相比随机 Monkey,无规则驱动的 DFS 遍历的 Activity 中有 6 个应用的效果较优,有 4 个应用的效果较差,有 2 个应用相同;而 RDTA 测试过程中的 Activity 的覆盖率普遍比随机的 Monkey 和 DFS 高,并且随着测试覆盖率的提高,RDTA 在测试过程中也发现了较多的 Bug。

表 3　不同测试方法的活动和异常个数结果

Table 3　Number results of activities and bugs of different test methods

被测应用	Activity 个数			Bug 个数		
	DFS	RDTA	Monkey	DFS	RDTA	Monkey
手机管家	8	35	10	0	0	0
日历	7	8	5	0	0	0
联系人	7	11	8	0	0	0
客服	14	43	11	0	0	0
百度外卖	4	10	7	0	0	0
糯米	5	7	5	0	0	0
饿了么	15	25	12	1	1	0
有道词典	6	15	7	0	1	0
酷我音乐	3	7	3	0	0	0
美颜相机	14	27	12	0	0	0
熊猫 TV	7	20	8	0	1	0
百度地图	5	10	2	0	0	0

对于问题 2),我们在实验过程中对比了 DFS 和 RDTA 测试过程的状态转移情况,如表 4 所列。状态类型中 NEW 表示该状态在测试过程中是在未探索的情况下被第一次探索到的,即发现了一个新的状态;而 OLD 表示该状态在之前的探索过程中已经被访问过了,在该状态下需要进行状态的回溯,从而降低测试的效率和覆盖率。从实验结果可以看出,相较于 DFS,RDTA 能够发现较多的 NEW 状态,从而避免进入 OLD 状态而引起回溯。RDTA 降低了 OLD/NEW 的比例,因此其相较于 DFS 测试效果更好。

表 4　状态对比

Table 4　Comparison of states

被测应用	DFS			RDTA		
	NEW	OLD	OLD/NEW	NEW	OLD	OLD/NEW
手机管家	26	98	3.8	57	122	2.1
日历	23	192	8.3	25	178	7.1
联系人	39	163	4.2	67	118	1.8
客服	53	192	3.6	92	102	1.1
百度外卖	59	102	1.7	81	114	1.4
糯米	40	82	2.1	51	75	1.5
饿了么	40	121	3.0	85	64	0.8
有道词典	34	161	4.7	106	316	3.0
酷我音乐	29	321	11.1	70	234	3.3
美颜相机	46	152	3.3	62	136	2.2
熊猫 TV	23	81	3.5	73	160	2.2
百度地图	32	120	3.8	50	107	2.1

对于问题 3),对“饿了么”应用的规则进行了简单阐述。表 5 列出了该应用的 5 条简单规则,可以看出测试的效率有了明显的提高。例如,规则 1 和规则 2 是针对 *ImageView* 和 *TextView* 上的驱动事件,将 *TextView* 的优先级设置为 1,*ImageView* 的优先级设置为 −1,就可以减少触发 *ImageView* 和 *TextView* 上冗余的驱动事件;并且对于 *TextView* 的遍历方式,采取了顺序遍历 20 个 *TextView* 类型的界面元素的方法,这样的限制可以避免状态爆炸的问题;对于 ImageButton,通过实验发现它是回退的一个按钮,将其优先级设置为 −1,可以避免一些低效的回溯;对于. bwq,该 Activity 是关于用户登录名和密码的界面,该类复杂事件的自动测试难以生成,因此我们通过规则指定其输入为特定的值,如“用户名”“密码”。通过该示例可以发现,通过简单的若干条规则即可提升测试效果。

表 5　“饿了么”应用规则示例

Table 5　Example of rules defined for eleme app

Activity	Selector	Modifier	Action	Priority
. Launcher	//node[@class='android. widget. TextView']	TopFirst(20)	Click	1
. Launcher	//node[@class='android. widget. ImageView']	Random	Click	−1
. Launcher	//node[@class='android. widget. ImageButton']	*	Click	−1
. bwq	//node[@text='手机/邮箱/用户名']	*	TextInput(用户名)	2
. bwq	//node[@text='密码']	*	TextInput(密码)	2

结束语　本文首先阐述了相关工作,对现有的各种 Android 应用 GUI 测试技术进行了概述,其中基于 DFS 的测试算法在测试技术中的应用最为广泛;而脚本测试的方式最适合表达应用测试人员的先验知识和测试路径约束。其次,本文总结了基于 DFS 算法的 Android 应用的 GUI 的重要概念,并且抽象了测试模型和总体框架。然后,针对 DFS 测试过程导致效率低下和覆盖率不理想的原因从不同方面进行了总结。因此,本文着重研究了规则驱动的 Android 应用测试技术,将 DFS 的自动化测试与测试人员的外部规则结合,以达到更好的测试效果。外部规则的制定是针对 DFS 自动化测试中的问题,根据不同的问题,制定规则的语义内涵,并且外部规则的实现具有简单和通用的特点。最后,本文根据规则驱动的方案,设计了 RDTA 测试工具,验证了规则驱动的测试技术对测试的效率和覆盖率有了很大的改进,并且能够检测到可能存在的异常。

本文虽然针对 DFS 中存在的若干问题,使用规则的方式来约束完全自动化的测试过程,在测试效率上达到了一些改进,但是测试过程中仍然存在问题[15]。本文的规则具有良好的扩展性,仍然可以添加很多其他语义的规则并实现相应的规则验证的引擎;其次,本文实现的 RDTA 不支持复杂的手势事件和其他复杂事件,如录制声音等,导致不能进入某些测

(下转第 118 页)

knowledge in developer online forums via convolutional neural network[C]// Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering. ACM, 2016: 51-62.

[4] JURCZYK P, AGICHTEIN E. Discovering authorities in question answer communities by using link analysis[C]// Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management. ACM, 2007: 919-922.

[5] PENG H B, WANG J. Topology of the Knowledge Communication Network in Virtual Communities: Based on CSDN[J]. New Technology of Library and Information Service, 2009, 25(4): 44-49. (in Chinese)
彭红彬,王军. 虚拟社区中知识交流的特点分析——基于 CSDN 技术论坛的实证研究[J]. 现代图书情报技术,2009,25(4):44-49.

[6] LIN H F, WANG J, XIONG D P, et al. Category participation-based approach to find experts for community question answer services[J]. Computer Engineering & Design, 2014, 35(1): 333-338. (in Chinese)
林鸿飞,王健,熊大平,等. 基于类别参与度的社区问答专家发现方法[J]. 计算机工程与设计,2014,35(1):333-338.

[7] WATTS D J, STROGATZ S H. Collective dynamics of 'small-world' networks[J]. Nature, 1998, 393(6684): 440.

[8] ALBERT R, BARABÁSI A L. Statistical Mechanics of Complex Networks[J/OL]. Reviews of Modern Physics, 2002, 74: 47-97.

[9] CHEN L, NAYAK R. Expertise analysis in a question answer portal for author ranking[C]// Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01. IEEE Computer Society, 2008: 134-140.

[10] KAO W C, LIU D R, WANG S W. Expert finding in question-answering websites: a novel hybrid approach[C]// Proceedings of the 2010 ACM Symposium on Applied Computing. ACM, 2010: 867-871.

[11] BARABÁSI A L, ALBERT R. Emergence of scaling in random networks[J]. Science, 1999, 286(5439): 509-512.

[12] 何大韧,刘宗华,汪秉宏. 复杂系统与复杂网络[M]. 北京:高等教育出版社,2009:111.

(上接第 103 页)

试状态,从而影响测试覆盖率;在很多状态下,测试依赖其他状态,如查看"饿了么"历史订单需要订餐记录,否则测试难以覆盖。本文的未来工作将会扩展规则的语言,形成功能通用的 DSL(Domain Specific Language)语言,以低耦合的方式驱动测试过程,达到更好的测试覆盖率。

参 考 文 献

[1] DU R Y, WANG C H, HE K. Location Privacy Protection Technology on Smart Mobile Devices[J]. ZTE Communications, 2015, 21(3): 23-29. (in Chinese)
杜瑞颖,王持恒,何琨. 智能移动终端的位置隐私保护技术[J]. 中兴通讯技术,2015,21(3):23-29.

[2] CHOUDHARY S R, GORLA A, ORSO A. Automated test input generation for android: Are we there yet?[C]// 2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE). IEEE, 2015: 429-440.

[3] 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. ACM, 2013: 224-234.

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

[5] AZIM T, NEAMTIU I. Targeted and depth-first exploration for systematic testing of android apps[C]// ACM SIGPLAN Notices. ACM, 2013: 641-660.

[6] MEMON A M, BANERJEE I, NAGARAJAN A. GUI Ripping: Reverse Engineering of Graphical User Interfaces for Testing[C]// 2003 10th Working Conference on Reverse Engineering. 2003: 260.

[7] ANBUNATHAN R, BASU A. A recursive crawler algorithm to detect crash in Android application[C]// 2014 IEEE International Conference on Computational Intelligence and Computing Research (ICCIC). IEEE, 2014: 1-4.

[8] 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. ACM, 2014: 599-609.

[9] MIRZAEI N, MALEK S, PŠSŠREANU C S, et al. Testing android apps through symbolic execution[J]. ACM SIGSOFT Software Engineering Notes, 2012, 37(6): 1-5.

[10] 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.

[11] AMALFITANO D, FASOLINO A R, TRAMONTANA P. A gui crawling-based technique for android mobile application testing[C]// 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW). IEEE, 2011: 252-261.

[12] LI X, JIANG Y, LIU Y, et al. User guided automation for testing mobile apps[C]// 2014 21st Asia-Pacific Software Engineering Conference (APSEC). IEEE, 2014: 27-34.

[13] ZHANG C, XUE Y Z, CHEN J C. Design and application of Android platform-based GUI capture-replay testing tool[J]. Computer Application and Software, 2012, 29(12): 6-9. (in Chinese)
张灿,薛云志,陈军成. 一种基于 Android 平台 GUI 录制回放工具的设计与实现[J]. 计算机应用与软件,2012,29(12):6-9.

[14] CHOI W, NECULA G, SEN K. Guided gui testing of android apps with minimal restart and approximate learning[C]// ACM SIGPLAN Notices. ACM, 2013: 623-640.

[15] ZHANG D W, GUO X, HAN Z. Security and Trusted Intelligent Mobile Terminal[J]. ZTE Communications, 2015, 21(5): 39-44. (in Chinese)
张大伟,郭姮,韩臻. 安全可信智能移动终端研究[J]. 中兴通讯技术,2015,21(5):39-44.