OPhone自动化测试技术概述

■ 文 / 金镝

本文将对OPhone平台上可采用的几种自动化测试技术进行介绍，并对每种技术的优缺点做简要的总结。

OPhone台除了为应用程序开发提供丰富的API外，也为开展自动化测试提供了多种途径。

基于JUnit+Instrumentation框架的自动化测试

OPhone平台中整合了JUnit测试框架和Instrumentation机制。图1是JUnit+Instrumentation自动化测试框架的架构图。

图1 JUnit+Instrumentation自动化测试框架架构

JUnit是广泛应用于Java程序开发的单元测试框架，它定义了特定格式的方法作为测试用例，提供TestSuite测试套件和TestRunner测试执行器，分别用于组织测试用例和运行测试用例并反馈测试结果。

Instrumentation则是一种操作系统和上层应用程序之间的监控机制。当应用程序运行时，若Instrumentation设置为开启，它将在应用程序运行前初始化，与应用程序运行在同一个进程中，监测应用程序与系统之间的交互，并可以对应用程序进行一定的控制。

为了更加便捷地进行自动化测试，OPhone平台利用Instrumentation对JUnit框架进行扩展，如图2所示。

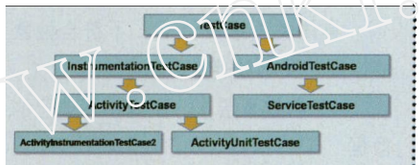

图2 OPhone平台对JUnit的扩展

经过这样的扩展，框架的易用性得到了提高。举例来说，假设AppActivity是某应用的一个Activity，我们想测试AppActivity界面上的一些功能，可以实现一个ActivitityInstruemtationTestCase2<AppActivity>的子类。当这个子类运行时，AppActivity就会自动启动，而省去了通过Intent等手动启动的步骤。

更重要的是，扩展后原有的JUnit单元测试被赋予了更多的功能。例如，用AndroidTestCase.getContext()、Instrumentation.getTargetContext()等方法就可以在测试代码中得到测试程序和被测应用程序的Context，进而访问到资源文件。

如果再利用Instrumentation模拟用户从图形界面发起的诸如点击或拖拽屏幕、输入字符、选择菜单项等操作，就可以实现自动化功能测试甚至系统测试。与数据库操作和View操作配合使用后，这种自动化测试能实现与测试人员手动测试同样的效果。

下面是一段基于JUnit+Instrumentation框架对记事本应用进行测试的自动化测试代码：

```
public void testInsertSaveNum()
throws Exception {
String TEST_TITLE = "Add-Save
Number";
String TEST_NOTE = "Note";
//启动NotesList
mNotesList = (NotesList)
mInstrumentation.
startActivitySync(mIntent);

//调用NotesList界面菜单中的"添加"
选项
mInstrumentation.invokeMenuAct
ionSync(mNotesList, NotesList.
INSERT_ID, 0);

mInstrumentation.
waitForIdleSync();

//此时转到了NoteEditor界面，利用
ActivityMonitor来得到NoteEditor对象
mNoteEditor = (NoteEditor) waitF
orActivity(mNoteEditorMonitor);
SystemClock.sleep(waitTime);

//在标题栏和内容栏输入内容
inputTitleNote(TEST_TITLE, TEST_
NOTE);

//调用NoteEditor界面菜单中的"保
存"选项
mInstrumentation.invokeMenuActi
onSync(mNoteEditor, SAVE_ID, 0);
mInstrumentation.
waitForIdleSync();
SystemClock.sleep(waitTime);

//此时应用程序转回到NotesList界面，
取ListView并校验数量
mNotesListView = (ListView)
mNotesList.getListView();
assertTrue("fail to add and
save a note",mNotesListView.
getCount() == 1);
    }
```

JUnit和Instrumentation相结合，既充分发挥了JUnit在测试用例开发、组织等方面的特点，又提供了对应用程序进行控制的有效途径。

JUnit+Instrumentation框架非常适

合对单个应用程序进行自动化测试。首先框架可以调用丰富的API，基本可实现对某个应用的绝大部分UI操作和数据库操作，并且可以调用很多系统级的API和标准Java API；其次，基于该框架的测试用例用Java代码直接实现，执行效率高；此外，每个应用的测试用例都可以编译成单独的apk安装包，既可以在模拟器上执行测试也可以在手机上执行，使用起来方便灵活。

然而，JUnit+Instrumentation框架也有其局限性。例如：一些界面元素，如Toast等无法获取和校验；跨应用测试难度大；复杂的交互性测试实现困难等。

基于键盘事件和图像识别的自动化测试

OPhone平台在操作系统层面提供将键盘事件转化为应用程序动作的能力：

步骤	动作
1	Window Manager 收到来自于Linux键盘驱动的按键事件
2	Window Manager 把scancode映射成keycode
3	Window Manager 把 scancode和keycode都发给应用程序
4	应用程序对按键事件进行响应

OPhone平台还提供了功能丰富的开发调试工具和工具库，如adb、sqlite3、viewserver和ddmlib等。

像上述的键盘事件，我们就可以通过adb shell input命令发送给设备，通过adb shell sendevent命令实现对触摸屏的操作。例如，想实现这样的过程：

1. 进入DCD应用的主界面；

2. 用向左滑动的触摸屏手势做翻页操作；

3. 调起该界面的菜单；

4. 退出DCD主界面。

就可以用如下命令实现：

```
//启动DCD主界面
adb shell am start -n oms.dcd/
oms.dcd.DcdClient

//在x向坐标00000095、y向坐标00000007
的位置点击屏幕，并保持按下状态
adb shell sendevent /dev/input/
event0: 0003  0000  00000095
adb shell sendevent /dev/input/
event0: 0003  0001  00000007
adb shell sendevent /dev/input/
event0: 0001  014a  00000001

//保持y向坐标不变，沿00000055到
00000010的x向路径横向滑动
adb shell sendevent /dev/input/
event0: 0003  0000  00000055
adb shell sendevent /dev/input/
event0: 0000  0000  00000000
……
adb shell sendevent /dev/input/
event0: 0003  0000  00000010
adb shell sendevent /dev/input/
event0: 0000  0000  00000000

//结束按下状态
adb shell sendevent /dev/input/
event0: 0001  014a  00000000

//发送一个菜单键值，调起菜单
adb shell input keyevent 1

//发送一个Back键值，收回菜单
adb shell input keyevent 4

//再发送一个Back键值，退出DCD界面
adb shell input keyevent 4
```

如果把这样的方法应用到具体测试用例上，那就是一个完整的自动化测试过程。对于需要操作数据库的情况，可通过adb shell sqlite3命令实现。不过测试必须有检验结果的校验点。可以使用图像识别的方式进行结果的校验，识别可以采用两种方法：

1. 利用系统viewserver中提供的接口获得当前界面上控件的属性信息，通过对比测试前后属性信息的变化进行结果校验；

2. 利用ddmlib包中提供的接口自动截取测试前后的屏幕截图，通过对比两幅截图相同位置的图像信息是否存在差异来进行结果校验。

以上是基于键盘事件和图像识别的自动化测试方法的基本原理。实际中，还需要做大量的工作，如支持循环或分支等复杂的测试逻辑、支持更复杂的触摸屏操作、实现多设备互通及交互操作、支持测试结果的自动生成等。

这种自动化测试的实现方式不受系统应用程序进程间安全机制的限制，突破了JUnit+Instrumentation框架下自动化测试不易跨应用测试的限制，而且也能实现多部终端的交互操作，为实现交互性系统测试的自动化提供了解决方案，同时也非常适合于替代重复性的压力测试类型的手动测试。

该方式的缺点是：（1）获取屏幕信息时存在时延，测试执行起来的效率会降低；（2）如果大量采用基于屏幕坐标上的触摸屏点击操作，需要维护多套测试脚本以适应不同屏幕分辨率的产品；（3）与UI实现紧耦合，如果应用的UI设计变化，需同步修改甚至重新设计测试用例。

基于OPhone API的自动化测试

OPhone平台开放的API中，很多都适合于开发测试程序。如利用ContentResolver可以操作应用的数据库，利用Graphic、Media、OpenGL等包中提供的接口可以实现对图片编解码、音视频媒体编解码、图形绘制能力等方面的测试。

也可根据需要，同时在测试程序中加入Instrumentation以扩展功能。

利用API的这种方式，适合于开发小巧的自动化测试工具，例如生成联系人记录的数据生成工具、可以连续拨打电话的压力测试工具。然而，由于它没有基于特定的测试框架，很难满足测试任务调度和组织的需要。

总结

以上介绍的三种自动化测试技术各有所长，如能在实际的工作中根据不同的测试类型和测试需求灵活使用，一定能提高OPhone平台及产品的测试效率和测试质量。

作者简介

金镝，中国移动通信研究院终端技术研究所项目经理。OPhone测试与质量团队成员，目前主要负责OPhone平台自动化测试方面的工作。

责任编辑：董世晓（dongsx@csdn.net）