基于持续集成的 Android 自动化测试①

王 焱, 张 征

(华中科技大学 自动化学院, 武汉 430074)

摘 要: Android 测试方面的研究大多集中在测试工具和框架的实现上, 有些工具和框架可以实现测试用例的自动生成和测试脚本的自动执行. 然而在项目开发过程中, 测试这个活动是需要人工启动的, 不能及时有效地保证新增或者修改代码的质量. 在 Robotium 测试框架的基础上, 通过研究持续集成方案, 包括被测代码和测试代码的托管、版本控制, 应用的自动构建, 测试的自动执行, 实现了 Android 的自动化测试平台. 使用该测试平台, 可以及时自动地对被测代码的修改进行测试, 直观可控地保证了 Android 应用的质量.

关键词: Android 测试; Robotium 框架; 自动化测试; 持续集成

Continuous Integration Based Study on Automated Testing for Android Platform

WANG Yan, ZHANG Zheng

(School of Automation, Huazhong University of Science & Technology, Wuhan 430074, China)

Abstract: Research on Android test mostly focused on the realization of the testing tools and frameworks, some tools and frameworks can realize automatic test case generation and test scripts execution. However, during software development, test requires human start, thus can't guarantee the quality of new or modified code timely and effectively. Based on Robotium framework, this paper researched continuous integration solution, including the code under test and the test code hosting, version controlling, building the application and test automatically, finally realized the Android automation test platform. With the test platform, the code updates can be test automatically in time, the quality of the Android applications can be intuitive and controllable.

Key words: Android test; Robotium framework; automated testing; continuous integration

伴随着 Android 系统的走红, 市场对 Android 应用程序(APP)的开发提出了更高的要求, 快速推出、快速迭代、快速响应用户的需求. 然后在这个 “快速”的过程中, APP 的质量难以等到保证, 在用户使用时出现应用无响应(ANR, Application Not Responding), 甚至直接崩溃(Crash), 导致的直接后果就是被用户卸载. 如何才能保证产品的质量? 这就需要对 Android 应用程序进行全面严密的测试.

Android APP 的测试在业界已经有大量的研究, 从 JUnit 单元测试, 到 Instrumentation 进行全方面的集成测试, 还有 Monkey、MonkeyRunner、Robotium、Robolectric 等自动化测试工具和框架[1]. 其中 Robotium 框架由于其功能强大、使用简单赢得了大量的用户, 它能够模拟几乎所有的用户操作, 实现对 APP 全面的功能测试. 同时, 国内外学术领域对 Android 测试也有很多研究. 张灿等[2]基于录制－回放技术提出了 Android 平台的自动化测试解决方案. 候菊敏[3]把基于关键字驱动的软件测试技术移植到 Android 测试中, 实现了从数据文件中直接导入测试用例进行测试, 使测试数据与测试代码分离, 降低测试数据与测试代码的耦合性. 郭柏廷[4]提出基于一段式状态机的 GUI 测试模型, 可以提取 Android 应用程序 GUI 控件可能的互动行为, 用来协助生成测试用例序列. Tae-San Baek 等[5]从 Android 应用程序源码中抽象出状态图, 根据该状态

① 基金项目:国家自然科学基金(61100145)

收稿时间:2014-11-21;收到修改稿时间:2014-12-29

图产生测试场景, 并用 Jython 语言实现自动化测试. Amalfitano 等[6]基于爬虫技术建立 Android 应用程序的 GUI 模型, 并生成可以自动执行的测试用例.

这些研究大多是集中在 Android 测试工具和框架的实现上. 在自动化测试方面, 关注在测试用例的自动化生成和测试脚本的自动化执行上. 然而测试作为项目的一个重要部分, 应该在项目开发过程能够随时自动化运行, 每次修改应用代码或者添加新代码的时候, 应该自动运行部分或者全部测试代码, 从而保证代码的修改和添加都能通过先前的测试[7].

本文基于软件工程的持续集成思想, 提出了一种解决 Android 自动化测试问题的方案. 持续集成是一种敏捷软件工程实践, 通过持续地、频繁地集成和测试, 来减少集成时间和提高软件质量[8], 它的一个显著特点是自动化的. 本文的研究旨在实现基于持续集成的 Android 自动化测试平台, 在该平台上使用 Robotium 框架对 Android APP 进行测试, 从测试之前的准备, 到测试的执行, 最后测试结果的输出, 这一整套自动化流程建立在许多优秀的开源软件和工具之上.

1 Android测试框架Robotium

Robotium 是一款测试 Android Application 的测试框架, 它是 Android 自带类 Instrumentation 的一个封装, 方便测试人员直接调用封装好的接口, 实际上我们直接使用 Instrumentation 也能够进行全面的测试, 但 Robotium 可以简化我们的测试步骤.

如图 1 所示, Robotium 基于 Android 自带的测试工具类 Instumentation, 应用程序和测试程序运行在同一个进程中, 测试程序通过 InstrumentationTestRunner 来控制应用程序, 实现各种测试操作.

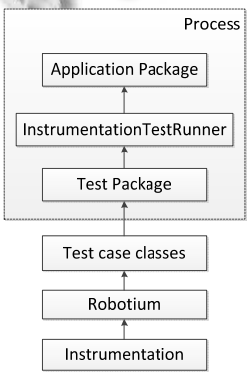

图 1 Robotium 测试架构

Robotium 框架中最重要的一个类是 Solo, 它实现了对 Android 控件操作的封装, 下面给出部分示例:

```
solo.clickOnButton("ok") 点击 ok 按钮;
solo.sendKey(Solo.MENU) 模拟键盘 menu 点击事件;
solo.clickOnText("More") 点击屏幕上文字为 More 的控件;
solo.enterText(2, "abc") 在屏幕上第三个 textView 中输入 abc;
solo.takeScreenshot("001") 截屏, 保存文件命名为 001;
……
```

基于 Robotium 框架的测试用例编写与 JUnit 类似, 所有测试方法命名以 test 开头, 另外有两个特殊的方法:setUp()和 tearDown(), 在测试开始前和结束后做一些初始化和垃圾清理的事情. 对 Android 应用程序的测试, 在测试操作之前需要先加载一个启动 Activity 类.

Robotium 框架使用系统提供的 InstrumentationTestRunner, 在测试 AndroidManifest.xml 中指定 instumentation 的 name 和 targetPackage, 其中 name 指定所使用的测试工具, targetPackage 指定被测应用的包名. 另外需要使用 android.test.runner 这个系统类库, 如图 2 所示. 在 Eclipse 中, 测试以 Android JUnit Test 运行.

```
<instrumentation
    android:name="android.test.InstrumentationTestRunner"
    android:targetPackage="dev.tonywang.sharetokk" />

<application
    android:icon="@drawable/ic_launcher"
    android:label="@string/app_name" >
    <uses-library android:name="android.test.runner" />
</application>
```

图 2 Android 测试项目 AndroidManifest.xml 文件

2 Android持续集成系统设计

2.1 系统组成

搭建持续集成环境, 首先需要一个持续集成的平台和一个统一的代码库[9], 通过持续集成平台监听代码库的变化, 有变化则拉取新的代码, 采用某种方法进行新的集成构建, 并运行自动化测试, 测试之前需要部署好相应的测试环境, 在本文研究的 Android 测试中, 需要启动 Android 模拟器(AVD). Android 测试持续集成系统组成如图 3 所示.

(1)持续集成平台

当前最流行的开源持续集成平台是 Jenkins, 它由以前的 Hudson 更名而来. Jenkins 主要由两部分组成, 第一部分是一个可以独立运行的 Java 程序, 它主要负

责运行构建. 另一部分则主要是 Web 程序, 它主要负责报告构建的状态与最终的生成结果[10].

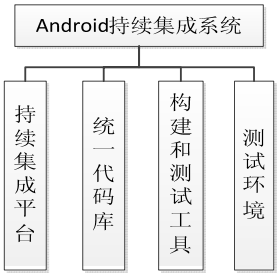

图 3 Android 持续集成系统组成

本文所研究的 Android 自动化测试, 所有配置、自动化操作和测试结果展示等事务都是在这个平台上进行, 包括源代码的拉取、Android 应用的构建、测试环境的搭建、测试的运行. 而实现这些功能大部分都是通过插件的形式调用服务器上的工具.

(2)统一代码库

持续集成有个重要的要素是, 具有一个统一的代码库. 而对于同一个代码库, 多人在协作开发项目时, 需要使用代码版本控制 VCS(Version Control System)来管理. 主流的 VCS 软件有 CVS、Subversion 和 Git[11], 本文选择使用 Git, 一是因为它是分布式的管理, 本地可进行提交;二是因为它具有更快的速度和更高的效率[12]. 另外选择 GitHub 作为代码托管库, 这样不用自己搭建一个代码库服务器, 而 GitHub 和 Git 的配合也是大家常用的.

(3)Android 构建工具

基于 Robotium 的 Android 测试需要先构建出 APK(Android Package), 运行在 Android 系统上, 然后再运行测试程序. 大多数时候, Android 开发使用 Eclipse 等集成开发环境(IDE)进行 APK 的构建, 然而在持续集成平台上一切都是通过命令操作, 项目构建应该通过运行一个简单的命令来自动完成, 构建工具就是为了实现这个作用. Java 项目最常见的两个构建工具是 Ant 和 Maven, 并且它们都很好地支持 Android 项目. 而 Maven 具有依赖管理, 打包构建, 测试, 代码分析等非常强大的功能[13], 开源项目 Android Maven Plugin 允许利用 Maven 的这些功能来开发、构建、部署、发布 Android 应用.

(4)测试环境——Android 模拟器

被测 APK 和测试 APK 需要安装到 Android 系统中来进行测试, 所以需要 Android 模拟器或者真机用来测试, 本文研究选择模拟器, 可以在测试开始的时候启动, 测试结束时关闭.

2.2 系统架构

本文采用 Jenkins 持续集成平台, 使用 GitHub 托管源代码, Git 来做源代码的版本控制, 并使用 Maven 来构建 Android 应用和运行基于 Robotium 的测试应用. 整个集成环境架构图如图 4 所示.

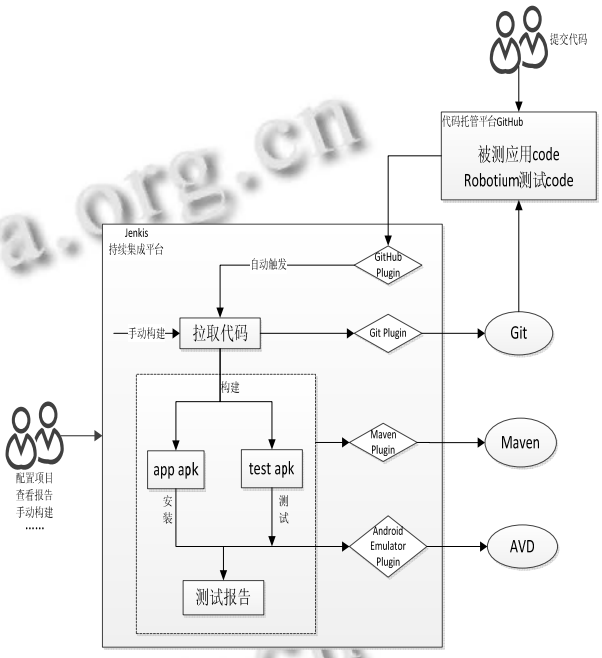

图 4 Android 持续集成环境架构

当源代码有新的提交时, GitHub 代码库通知 Jenkins 平台, Jenkins 平台通过 Git 拉取最新代码, 启动 Android 模拟器, 并使用 Maven 构建、安装和测试, 输出测试报告, 实现自动化测试. 这个测试过程中 Jenkins 平台使用的工具和软件, 都是通过插件的形式实现的.

Jenkins 安装插件需要在 Jenkins Web 页面的“系统管理”—>“插件管理”中配置. 本文研究内容使用到四个 Jenkins 插件: GitHub Plugin, 用来实现 Hook 功能, 当 GitHub 上的源代码有新提交时, 能通知 Jenkins 平台; Git Plugin, 用来调用本地的 Git 软件从代码库拉取代码; Maven Plugin, 用来调用本地的 Maven 软件, 用来构建、安装、测试 Android 应用, 并生成测试报告; Android Emulator Plugin, 用来启动本地的 Android 模拟器, 在测试之前准备好测试环境.

3 实例研究

本节首先以 Android SDK 自带的 NotePad 工程为例，采用上文所介绍的持续集成方案，完成测试环境搭建、测试任务新建、测试执行和测试报告输出，验证该自动化测试方案的正确性. 然后再在这个测试环境上，对作者正在开发的远程智能医疗项目 HealthCare APP 做全面的测试，给出其测试数据.

3.1 搭建 Android 测试持续集成环境

Jenkins 持续集成平台的安装非常简单，在官网 http://jenkins-ci.org/ 直接下载 jenkins.war 包(也可以选择各个系统平台下的本地安装包)，部署服务，本地访问 http://localhost:8080/jenkins/ 即可使用.

安装 Android SDK, Maven, Git 等工具，并且安装相应的 Jenkins 插件. 本文 Jenkins 部署在 Windows 系统上，平台如图 5.

图 5 Jenkins 平台

3.2 创建基于 Robotium 框架的 Android Maven 项目

快速创建一个 android maven 项目可以借助 android-archetypes 开源库，这里创建的是一个 android-with-test 原型，包括应用源代码和测试代码，创建命令:

```
mvn archetype:generate
-DarchetypeArtifactId=android-with-test
-DarchetypeGroupId=de.akquinet.android.archetypes
-DarchetypeVersion=1.0.11
-DgroupId=com.tonywang.test
-DartifactId=notepad-android-project
-Dpackage=com.example.android.notepad
```

创建的 android maven 目录结构如图 6 所示.

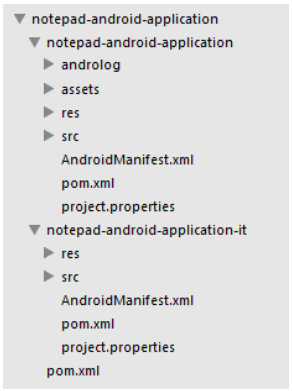

图 6 android-with-test maven 项目目录结构

其中 notepad-android-application 父目录下的 pom.xml 是整个 maven 项目的配置文件，在该文件中配置项目模块，管理项目依赖库，并使用 android-maven-plugin 开源项目的 maven plugin 来实现对 android 项目的管理. notepad-android-application 子目录是被测 android 代码，目录结构跟一般 Android 项目一致，该目录下的 pom.xml 用来定义它的 maven 配置，将 NotePad 代码和资源文件放入相应的文件夹. notepad-android-application-it 子目录是测试代码，测试依赖于 Robotium 库，而该库存在于 maven central repository 中，在该目录下的 pom.xml 的依赖项中添加以下代码，即完成对 robotium-solo.jar 的引入，在 src 中编写基于 Robotium 的测试脚本，就完成了测试代码.

```
<dependency>
    <groupId>com.jayway.android.robotium</groupId>
     <artifactId>robotium-solo</artifactId>
    <version>5.1</version>
</dependency>
```

3.3 Jenkins 上创建该 Android 测试任务

创建一个新任务，选择“构建一个 maven2/3 项目”，然后在任务配置中:

(1)添加源码管理，配置 Git 远程 repository url

图 7 Jenkins 源码管理配置

(2)添加构建，配置 Maven 命令和 pom.xml 文件路径:

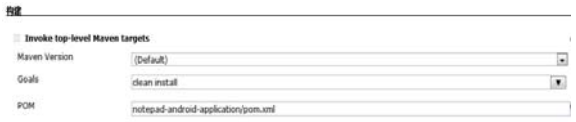

图 8 Jenkins 构建配置

(3)添加构建环境，配置测试之前启动 AVD:

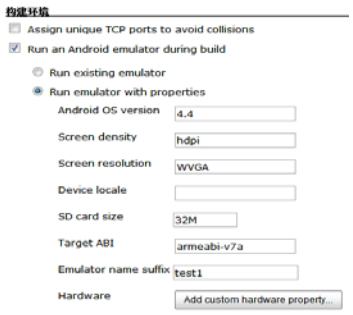

图 9 Jenkins AVD plugin 配置

3.4 构建和测试报告

支持手动构建和自动构建，自动构建可以选择当 GitHub 中源代码有提交时触发构建，这样每次提交后就可以实现自动测试，测试开始之前会自动启动 Android 模拟器. 在测试完成后生成测试报告，如图 10 所示.

图 10 NotePad 测试报告

3.5 HealthCare 测试

远程智慧医疗项目 HealthCare 的 Android 客户端采用 Holo 风格设计，功能模块强大，使用了多个 Android 组件和控件，并且与服务器有大量的交互，需要进行严密的测试.

用 Maven 管理整个项目，在编写 HealthCare 源代码的同时编写基于 Robotium 的测试代码，采用上文搭建的测试环境，实现 HealthCare 的自动化测试. 如图 11 所示，源代码质量持续改进，从最开始的 13 个测试用例中有 12 个失败，到最后全部通过. 通过这个持续改进的过程，项目管理者对项目的进展和质量有了直观的把控. HealthCare APP 的最后测试报告如图 12.

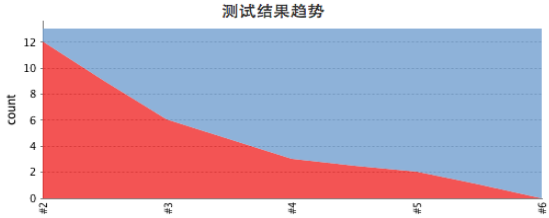

图 11 HealthCare 测试结果趋势图

图 12 HealthCare 测试报告

4 结论

Robotium 是一个非常强大的开源 Android 测试框架，它能够非常简单快速地实现 Android APP 的集成测试或用户场景测试. 然而，整个测试过程应该包括被测应用的构建、被测应用的安装、测试代码的执行、测试报告的输出等一系列活动，并且这些活动应该自动执行.

持续集成平台是一个容器，在软件项目的开发和测试过程中，各个阶段的控制都可以集成在这个平台上. 本文使用 Git 进行被测应用和测试代码的版本控制，GitHub 托管源代码;使用 Maven 构建和安装 Android 被测应用，执行基于 Robotium 框架的测试代码;应用安装和测试执行在 Android 模拟器上进行;测试报告输出并展示给用户. 这所有的流程都集成到 Jenkins 持续集成平台上，实现 Android 测试的完全自动化.

参考文献

1 蔡增柱.基于 Android 移动平台测试相关技术研究[硕士学位论文].广州:华南理工大学,2012.
2 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.计算机应用与软件,2012,29(12):6–9,68.
3 侯菊敏.基于 Android 的关键字驱动自动化测试框架研究[硕士学位论文].广州:中山大学,2012.
4 郭柏廷.Android 应用程序之 GUI 自动化测试方法[硕士学

位论文].台北:台北科技大学,2013.

5 Baek TS, Lee WJ. A GUI testing technique based on activity lifecycle for android applications. Journal of IEMEK, 2013, 8(6).

6 Amalfitano D, Fasolino AR, Tramontana P. A GUI crawling-based technique for android mobile application testing. Software Testing. Berlin, IEEE. 2011: 252–261.

7 Milano DT. Android Application Testing Guide. 1th ed. Birmingham: Packt Publishing Ltd, 2011.

8 Meyer M. Continuous integration and its tools. IEEE Software, 2014, 31(3): 14–16.

9 Geiss M. Continuous Integration and Testing for Android. Berlin Institute of Technology (TU-Berlin), 2012.

10 周文哲.基于表述性状态转移的持续集成平台研究与应用[硕士学位论文].长沙:湖南大学,2012.

11 刘悦之.基于 Git 的分布式版本控制系统的设计与实现.科技传播,2012,22:197–198.

12 Spinellis D. Git. IEEE Software, 2012, 29(3): 100–101.

13 Spinellis D. Software builders. IEEE Software, 2008, V25 (3): 22–23.