模型驱动的移动应用测试方法

冯 谷 李尼格
(全球能源互联网研究院信息通信研究所 南京 210003)
(信息网络安全国网重点实验室 南京 210003)

摘 要 移动互联网时代,移动智能终端和移动应用已经成为各领域竞相采用的祖尧解决方案。与此同时,移动终端硬件和平台的多样性和异构性,造成了移动应用开发和测试中存在大量工作冗余以及难以复用的情况,带来了跨平台问题,成为了学术界和工业界研究和实践的热点。文中提出使用模型驱动的移动应用测试方法。首先,使用UML状态机模型刻画移动应用的行为;然后,基于移动应用的行为模型,自动生成平台无关的测试用例;最后,将与平台无关的测试用例映射到多个移动平台,产生可执行的自动化测试用例。在此基础上,选择形如掌上电力的应用作为案例,分别实现IOS和Android平台上的自动化测试,验证了模型驱动移动应用测试方法在解决跨平台问题时的有效性。

关键词 移动应用,移动终端,跨平台,模型驱动,测试用例,平台无关,自动化测试
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.11.035

Model-driven Testing for Mobile Applications

FENG Gu LI Ni-ge
(Information Communication Research Institute, Global Energy Interconnection Research Institute, Nanjing 210003, China)
(State Grid Key Laboratory of Information & Network Security, Nanjing 210003, China)

Abstract In mobile Internet, the intelligent mobile terminal and mobile applications are widely used in all fields. At the same time, the diversity and heterogeneity of the mobile terminal hardware and platform have caused redundant work in mobile application development and testing. Cross platform issues arise and it becomes a hot area for study and practice in academic and industry. This paper proposed model driven testing for mobile applicaitons. Firstly, the method uses UML state machine to describe the behaviour of the application. Secondly, based on the behaviour model, the method generates platform independent test cases automatically. Lastly, the method maps the test cases which are unrelated to platform to multiple platforms and generates executable test cases. This paper chose a power application as an example and realized the automation test in both IOS and Android. The validity of the model driven testing method was verified in solving the cross platform problem.

Keywords Mobile application, Mobile terminal, Cross platform, Model-driven, Test case, Platform independent, Auto-testing

CNNIC发布的第三十七次《中国互联网发展状况统计报告》[1]显示,截至2015年12月,手机上网人数已有近6.20亿,目前其使用的主流平台有Android平台、IOS平台等。运行在移动智能终端上提供服务的软件系统,即用户手机安装的App的数量和种类都在高速增加[2],用户对其要求也越来越高,主要考虑该应用是否安全可信、友好、耗电等。而国家电网所提供的应用已经涉及到大多数用户,其可信度尤为重要。为保障某个移动应用的质量,一般在发布该应用之前对其进行全面的测试。由于多平台(Android,IOS)特性,对于同一个移动应用,开发和测试团队需要对不同的平台分别进行开发和测试,但由于缺乏跨平台测试技术,导致资源浪费。

当前移动应用开发团队在开发应用的过程中主要面临的问题有:1)对应用的功能测试问题;2)跨平台问题。

本文提出了模型驱动的移动应用开发和测试方法[3-4],希望能够在一定程度上解决上述问题。该方法首先,帮助开发团队实现测试时的自动化,不需要手动操作;其次,尽量保证测试用例完全,而并非只是完成部分;最后,解决跨平台的问题,针对不同平台上的同一应用,只需要一个团队即可同时进行不同平台上的测试。所提方法能够帮助公司节省大量的人力物力,也能够保证移动应用在上市之前更加安全。

经过研究,此次项目的过程为:首先根据样例应用的功能来刻画模型;其次开发者根据模型进行应用开发,为了让所开

到稿日期:2016-09-08 返修日期:2016-12-24 本文受国网公司科技项目:面向电力移动终端的应用测试技术研究(5455HT150029)资助。
冯 谷(1985-),男,高级工程师,主要研究方向为电力行业信息安全技术、攻防渗透技术,E-mail:fenggu@geiri.sgcc.com.cn;李尼格(1985-),女,高级工程师,主要研究方向为电力行业信息安全技术,E-mail:linige@geiri.sgcc.com.cn。

发的应用得到用户的好评，需在发布之前对其进行详尽的测试，因此需要通过之前得到的模型得出完备的测试动作序列；最后根据这些动作来自动生成测试脚本，从而完成对该应用的自动化测试，该步骤的关键点是本次研究的重点，即构造一个动作-脚本 API 映射表。上述过程的系统架构如图 1 所示。

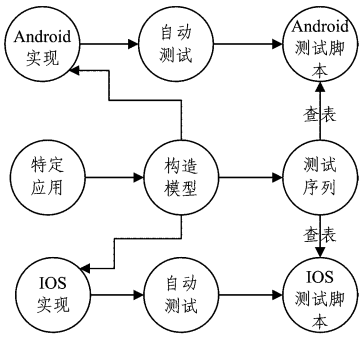

图 1　系统架构

本文第 1 节介绍相关技术背景和已有工作，其中包括 Android、IOS 系统、两大系统上基于模型驱动的移动应用的开发以及目前现有的测试方法和工具等；第 2 节具体阐述模型驱动的移动应用开发；第 3 节介绍模型驱动的移动应用自动化测试；最后总结全文并展望未来。

1　相关工作

本文主要针对两个主流平台（Android，IOS）进行模型驱动的移动应用开发和测试。随着技术的不断发展，目前已有较多的相关理论、工具来支持本次研究。

1.1　模型驱动的开发和测试

近年来，由于用户对软件或者应用的要求越来越多，软件的功能越来越多，整个应用越来越复杂，且在不停地扩展和更新以满足用户不断变化的需求。

模型驱动技术可以很好地帮助企业解决软件开发测试中的复杂性和可移植性问题。模型驱动中的模型指将现实世界中的事物进行抽象，然后对其自身的某些功能、结构特点、行为方式等进行形式化规范；平台无关模型（PIM）用于描述与具体的实现技术以及实现平台无关的系统功能与结构。平台相关模型（PSM）描述与具体的实现平台以及实现技术相关的、带有具体技术特征的系统功能和结构[4]。

企业在进行移动应用或软件开发时，首先对其能够完成的功能、逻辑联系、业务流程等进行分析，从而完成构建与具体技术无关的平台无关模型 PIM。由于在任何平台上需实现的功能是一致的，因此应用的执行过程、操作等都是一致的。接下来按照不同的平台特点以及实现方式产生对应的、合适的映射规则，这将上文构建的与技术平台等无关的 PIM 变成了 PSM 平台相关模型。

我们希望通过对一个移动应用进行建模来达到 4 个目标：1）能够让开发者更好地展现整个系统；2）允许开发者规定该系统的行为能力和一些特点；3）为开发者提供搭建该系统的模板；4）记录开发者的决策。随着技术的不断发展，现有的通用建模语言众多，如自然语言、图形语言、数学语言等，比较主流的是 UML 语言。

模型驱动的移动应用开发[5]步骤为：1）模型搭建；2）选择应用开发的平台，并据此选择合适的开发工具和开发语言；3）展开实际开发工作，将模型转换成适合特有平台的代码。模型驱动开发技术对于后期的更新维护具有较大的作用。

模型驱动的移动应用测试技术是较为有效的技术，给开发者带来了较大的便利。模型驱动测试的第一步在开发时已经完成，即构造抽象模型；第二步即依据模型覆盖测试标准来产生对应的、抽象的自动化测试用例；最后产生测试脚本并对应用进行测试，从而给出测试结果。

总体来看，模型驱动测试需要具备 3 个基本条件：模型、测试用例、测试脚本。

模型驱动测试技术具备的比较突出的特点有：1）对于移动应用开发者而言，模型驱动测试方法提高了测试的自动化水平和效率，大大节省了人力物力；2）模型驱动测试经常能发现其他测试技术很难发现的问题，从而能够确保软件的质量；3）生成的某个测试用例可以被多次利用。

建模是模型驱动测试技术的必要步骤，应用规模较大，为了满足用户不断变化的需求，企业开发者在开发应用之前将为该移动应用构造一个模型，以便开发。本文研究采用状态图来刻画应用，使得开发者对整个应用一目了然。

在建造模型的过程中，建模工具的选择十分重要，建模工具众多，且各工具有其各自的特色及适用范围，常见的有 Rational Software Archtect（RSA），PowerDesigner，Visio 等。

RSA 是直接从 UML 发展而来的设计工具。RSA 的主要优点在于能够较好地描述开发过程中的所有语义、对象、流程以及状态等。RSA 的角度比较宽泛，一般以全局的思想进行分析设计，因此能够使系统更加明确易懂，同时内部的各种结构也更加清晰。但是 RSA 对开发者的要求较高，需要开发者掌握 UML，对一般使用者而言并无优势。

PowerDesigner 最开始是一种数据库建模工具，经过不断发展才引入了对 UML 的支持。其主要优势在于能够较好地支持数据库建模，几乎对所有数据库均有效，通常采用它对数据库建模，但其不能较好地支持 UML 建模时使用到的各种图，因此较少使用它来进行 UML 的开发，但最近其对图的支持能力正逐渐得到加强。

UML 建模工具 Visio 原来仅仅是一种画图工具，但经过不断发展，目前其是最能够用图形方式来表达各种商业图形用途的工具，因此使用它来进行图形语义的描述较有优势。

模型建造并非本次研究中最主要的内容，只需用状态图的形式刻画应用即可，并不需要描述所有的 UML 元素（行为图、类图等）。基于此，Visio 工具更适合本项目，因此本次研究选择 Visio 作为建模工具。

1.2　移动应用开发

在开发一个移动应用之前，专业的开发者需确定关键点：1）确定该应用的功能及业务流程。为解决该问题，对应用需求进行分析，然后对功能进行抽象并建模，从而得到一般的状态转换图。2）开发平台、语言以及工具的选择。一般而言选择有多种，比较合适的选择将有助于整个工作的进行。其中开发语言和开发工具一般视开发平台而定，由于本次研究主

要考虑两个主流平台(Android,IOS),因此限定在这两个平台上进行开发。

1.2.1 Android 平台

Android 是基于 Linux 的操作系统,它由谷歌公司和开放手持设备联盟共同开发,目前较多智能终端都采用该系统。该系统对于开发者而言所具备的优点有:1)开源,这吸引了较多研究者来了解并完善它,大大加速了该平台的发展;2)兼容性好,智能终端间的差异较多,但 Android 系统能够兼容这些差异,因此较多的生产者和消费者都倾向于选择 Android 系统;3)敏捷开发,谷歌给开发者带来了自由的开发环境,从而使得众多开发者能创新地开发各种各样的应用。

Android 架构主要由 5 部分组成:Linux Kernel,Android Runtime,Libraries,Application Framework,Applications[6],如图 2 所示。

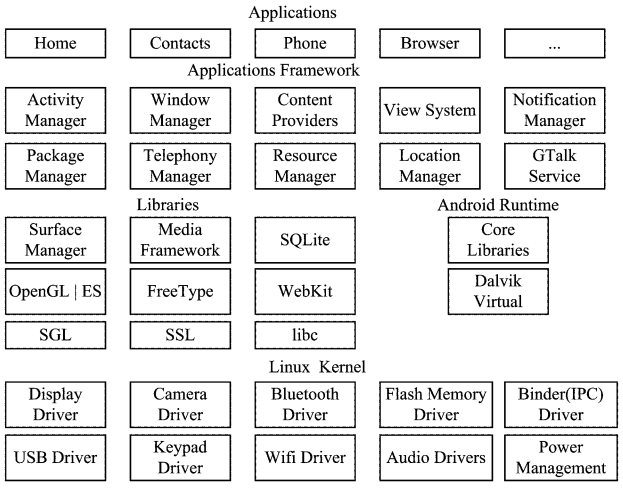

图 2 Android 系统架构图

为了更加清楚地了解 Android 系统,下面详细介绍各个层次。最下层为整个架构的核心层,主要为上面层次提供基础服务等;Runtime 中包含了核心库,提供了大部分在 Java 编程语言核心类库中可使用的功能;Library 中是一个 C/C++ 库的集合,供 Android 系统的各个组件使用;上面两层面向应用,框架提供了开发平台,使开发者开发出一些实用且新颖的应用;最上层为主要的已有的应用程序集合,如日历、地图等。

Android 开发语言是所有程序员都比较熟悉的 Java 语言,对应的环境为支持 Java 运行的环境。一般 Java 运行需要 JDK 以及 Eclipse 即可,而现在还要求具有 Android 应用需要的 android sdk 以及 Android 开发工具 ADT[7]。

1.2.2 IOS 平台

IOS 是苹果公司开发的移动操作系统,近年来已成为主流系统,主要用于 iPhone,iPod,iPad,iTouch 等。因此公司在开发一个移动应用时必然需要一个团队在该系统上进行开发,但该系统是封闭的,相比 Android 系统而言有一定的难度。

IOS 系统可分为 4 个层次:核心操作系统层、核心服务层、媒体层、可轻触层,具体架构如图 3 所示。

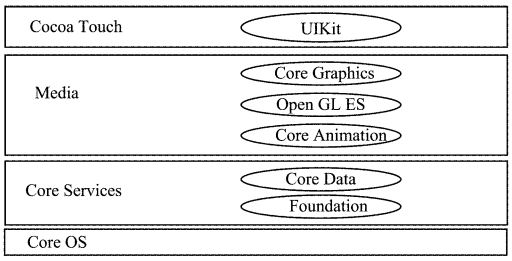

图 3 IOS 系统架构图

为了更加清楚地了解 IOS 系统,下面详细介绍各个层次。最底层的 Core OS 提供了整个系统的基础功能,如硬件驱动、内存管理、程序管理、线程管理、文件系统、网络和标准的输入输出等,这些功能都通过 C 语言的 API 来提供;Core Services 提供了比 Core OS 层功能更为丰富的基础服务;Media 层为 IOS 系统中的移动应用提供了图片、音乐、影片等多媒体功能;最上层 Cocoa Touch 由多个不同的框架构成,在移动应用开发中该层为核心,如 UIKit 负责启动和结束应用程序、控制界面等。

对于开发者而言,由于 IOS 移动应用的开发语言是一种特定的语言,不同于 C++,Java 等常见语言,因此开发语言成为难题。OC(Objective-C)[8]语言是一种通用的面向对象的语言,它是苹果的 OS X 和 IOS 操作系统,以及相关的 API,Cocoa 和 Cocoa Touch 的主要编程语言。该门新语言拥有较多特色,是 C 语言的超集,与 C 语言完全兼容,因此在一定程度上易于学习,在一个项目中可同时使用 OC 和 C++,还可采取另一种在项目中导入库文件的方式。

IOS 的特殊性已经一目了然,对应的 IOS 上的应用开发也需要特殊的工具。Xcode 是苹果公司向开发人员提供的集成开发环境(该环境是闭源的),它运行于 Mac 操作系统下。该工具随着 IOS 系统的不断更新而不停地发展,目前的最新版本为 6.2 版本[7]。

Xcode 对于开发者而言是一个很好的开发工具,因为它具有功能强大、易于学习等诸多优点。代码编辑器对程序员而言至关重要,较优的编辑器可以帮助程序员编写正确的代码,而 Xcode 的编辑器正是如此,如它能够自动补全代码、提示错误和警告信息等。另外,Xcode 中已经搭建好了 Interface Builder,因此开发者可以通过类似于画图的形式来设计测试应用的用户界面,然后通过图形的方式将界面关联到 Xcode 编辑器中的源。Xcode 还提供了模拟器,以便开发者能够在相应的模拟器中构建、安装、运行以及调试创建的应用,在模拟器中顺利运行之后再进行真机测试。

1.3 移动应用测试

在移动应用的开发过程中,测试[9-10]过程非常重要,目前较多公司忽略了这一点,只为了追求快速开发,从而产生一个应用未测试完全就上市的现象。未经过完整且全面测试的移动应用可能会带来灾难性的后果。

1.3.1 Android 应用测试

Android 是最早的主流平台,对其已经进行了较多的研究,因此目前已有很多较优秀的针对 Android 平台的自动化测试工具,它们有各自的特点及适用范围,比较常见的有

Monkey，MonkeyRunner 以及 Robotium 等[11-14]。

Monkey 是 SDK 提供的一种自动化的测试手段。Monkey 测试指系统产生伪随机的事件流并且驱动设备执行，这些事件流不仅数量大而且执行速度快。然而，Monkey 并不支持给定的事件序列，用户只能通过参数调整事件的数量和频率，即执行的强度，或者偏向于哪一类事件，如按键等。因此，Monkey 通常用于压力测试，即检测程序的鲁棒性，并不符合本次项目中的功能性测试。

MonkeyRunner 同样也是 SDK 自带的测试工具，其主要使用 Python 脚本作为输入。程序一般模拟用户操作应用包，比如向应用发送安装、卸载、模拟击键、触屏等事件。因此，MonkeyRunner 大多被用来进行回归测试和功能测试。

MonkeyRunner 的常用 API 有：1）waitForConnection，与设备进行连接；2）startActivity，启动 Activity；3）touch，发送触摸事件，参数为 XY 坐标；4）type，向键盘发送字符串消息。

MonkeyRunner 的优势在于 Python 脚本的编写十分简单易懂，甚至只使用记事本即可完成，灵活性较强，支持的测试功能也较全面，能够自定义事件流，但坐标点击方式可能不适用于本次项目的研究。

Robotium[11]是一款 Android 自动化测试框架，可以对 Android 平台的应用进行黑盒测试和白盒测试。它与 MonkeyRunner 的相似点在于，用户可以自己给定一个事件流来进行各种测试操作。

Robotium 的优点在于它对于测试人员而言较容易，对应的测试用例也较简单，在仅有 APK 的情况下测试人员可以对此进行黑盒测试，而在有源码的情况下可以对此进行白盒测试。

Robotium 主要用于功能测试，程序会按照测试包中的方法逐步执行，在测试方法中可以加入断言机制来判断最终结果是否符合预期，并向用户反馈。Robotium 的核心是 Solo 类，该类可以对设备发送输入文字、点击按钮等命令。

几种工具中，Robotium 更适合本次项目，因为本次项目主要从开发者的立场出发，因此本次研究采用 Robotium 作为 Android 平台上的自动化测试工具。

1.3.2　IOS 应用测试

随着技术的不断发展，当前已存在众多针对 IOS 平台的较完善的自动化测试工具[15-16]。但是对于新手而言，录放式 UI 测试工具将更易于使用，因为可以利用该工具先录制一些操作，然后再进行回放以完成测试，其中较常用的有 Instrument，FoneMonkey 等。Instrument 集成在 Xcode 工具中，更为简便容易，因此本次项目选择 Instrument 中的 UIAutomation 作为 IOS 移动应用测试工具。

UIAutomation 是苹果公司发布的一个测试框架，测试人员可使用它在真实的或模拟器上执行 IOS 应用的自动化测试。该框架支持 Javascript 语言，其本质是一个 Javascript 的类库，通过界面上的 UI 元素及其可访问性来完成一些相应的交互操作，从而进行一些基本的测试。因此对于开发者而言，在使用 UIAutomation 进行测试时需做到两点：1）找到界面上的一个 UI 元素；2）指定针对一个 UI 元素的操作。

新手可能不会编写 UIAutomation 的测试脚本，因为它拥有较多未知的 API。但该工具提供了录制回放功能，它会通过录制动作来自动生成一系列脚本，可能生成的脚本不完全正确，但已有大体框架，测试人员可以据此来学习脚本的编写，在掌握一定知识后直接编写脚本并测试。

综上所述，模型驱动技术均已得到了广泛应用，其极其需要能够对移动应用进行自动化测试的工具平台，现已开始不断尝试开发和改进各类测试工具，并已经取得了一些成果，为本文工具的设计与实现提供了借鉴与思路。

2　移动应用行为建模

模型理论现已经较成熟，本文希望通过模型驱动来进行移动应用的开发。通过构造移动应用的一组模型（用例图、状态图、顺序图等）对该应用进行开发。该技术有助于整个开发过程顺利进行，为后期对该应用的维护、更新等带来了便利。

2.1　实例

本文项目研究的目的为开发一个形如掌上电力的应用并对其进行自动化测试。

该应用包含如下大框架功能：用电查询、交费购电、网点导航、停电公告等，每个大框架下又包含较多小的功能，如网点导航又分为周边搜索和定制搜索，依此类推，将整个电力应用的功能不断进行细化。

图 4 给出了该应用的详细功能。

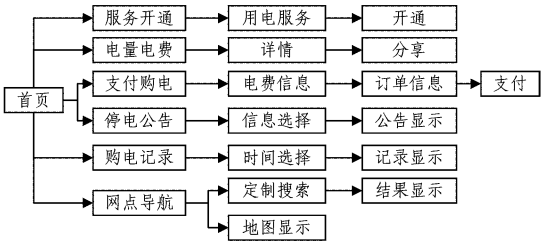

图 4　应用功能

本项目要求在两个平台（Android，IOS）上完成该应用的设计、实现及测试。

2.2　建模

UML[17-18]语言是主流的建模语言，本项目利用 UML 状态机模型来描述该电力应用的行为。

状态机图表示一个模型元素在其生命期间的状态：开始状态、响应事件、执行某些动作、发生状态迁移、在新的状态下继续操作、转移到另一个状态，如此不断发生状态改变，直到终结状态结束。首先需要确定状态图中的基本元素：状态、转移、事件等。

针对该电力应用，首先将其分为多个不同的状态，此处每个页面即为一个不同的状态，如网点导航界面为一个状态，用电查询为另一个状态等；然后定义不同的事件（如点击按钮、输入文字等）；最后描绘迁移，在某个事件的作用下从一个页面转移到另一个页面。

明确功能后，即可以利用 Visio 工具为该应用刻画用例图、状态机图、UML 类图以及顺序图，分别如图 5－图 8 所示。

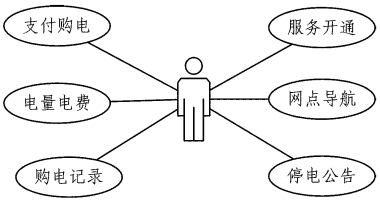

图 5 总系统用例图

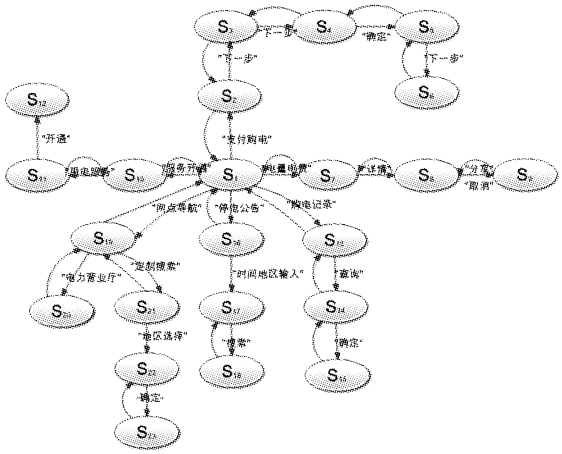

图 6 状态图

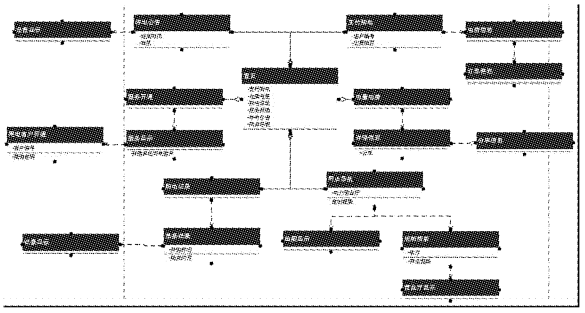

图 7 类图

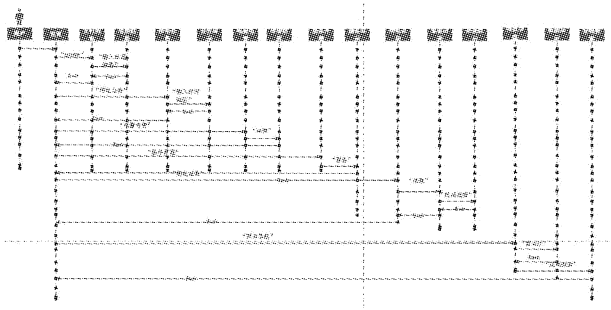

图 8 顺序图

2.3 移动应用实现

模型构建完成后即可进行应用的开发工作。如前文所述,在开始开发之前应做好相关准备工作,即确定开发工具、环境以及对应的语言。

上文介绍了当前两大主流平台(Android,IOS)的一些常见开发工具和语言。本次项目将选择 Xcode 为 IOS 平台的开发工具,开发语言为 OC 语言;选择 Eclipse 为 Android 平台的开发工具,开发语言为熟悉的 Java 语言(一种通用的、简单的、面向对象的程序设计语言,具有卓越的通用性、高效性、平台移植性和安全性)。

基本条件确定后,即可根据已经构造出的各种模型图来编写实际的代码从而开发移动应用。

无论是 Android 开发还是 IOS 开发,首先都是根据刻画出的 UML 状态机模型来刻画应用中的界面;然后根据状态之间的转换关系以及对应的平台语言规则来编写界面之间的跳转代码,直至整个应用完成。

图 9、图 10 给出了最终实现的电力应用(只给出了部分界面)。

图 9 应用界面一

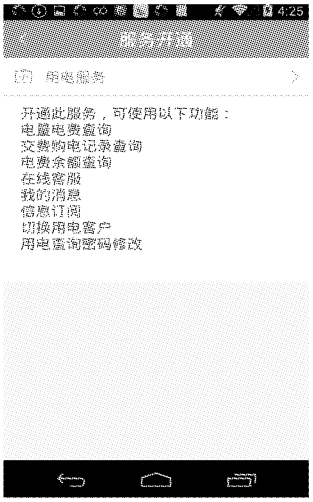

图 10 应用界面二

3 模型驱动的移动应用自动化测试

在应用发布前,对该应用的测试至关重要。本文对移动应用的自动化测试进行研究,从应用的模型出发构造完备的测试用例,并自动生成测试脚本对该应用进行测试[12-13,15-16]。该工作将为开发者的测试工作带来极大的便利。

3.1 基于模型构造的测试用例

在功能状态机已经刻画完成的基础上,从状态机的角度构造一系列的测试动作序列。

在模型驱动的移动应用测试过程中,用户需要通过与移动应用进行交互来对相关功能进行测试,而模型中的状态机已经描述了用户的动作事件,能够引起的移动应用的状态变化及用户和移动应用的预期行为,因此使用状态机模型来构造测试场景并产生测试用例是较合适的。

上述步骤是本次研究的重点,也是开发者亟需解决的问题,正是该问题的存在才导致较多开发者在未测试完全的情况下就发布了应用。面对一个大型的状态机,一定要确保得出的测试动作序列是完备的,而并非只是选择路径中的一条。在真实的开发测试中应做到完备测试,保证在每种可能的情况下都是正确的。若忽略了某种测试动作序列,则很可能会影响用户的使用。

3.1.1 路径算法

由于在实际的移动应用测试过程中希望快速地得到详尽的测试用例[19-22],因此在基于复杂度和覆盖能力的分析中选择全转换覆盖,以保证所有节点及转换都被至少遍历一次。本次项目选择的算法是基于广度优先的路径分析算法[23]。

首先从符号角度来定义状态机,一个状态机常被表示为 $SM=(S,S_0,\Sigma,Tran)$,其中 S 是一个状态的有穷集,$S_0\in S$

是初始状态；Σ 是输入符号的有穷集；*Tran* 是状态转换函数 $S_x\Sigma \rightarrow S$。

算法的输出是一组路径，每条路径形如 $p = init \rightarrow S_0 \rightarrow S_1 \rightarrow S_2 \rightarrow \cdots \rightarrow S_N$。

算法的基本思想为利用队列进行广度遍历。当转换函数中源点为起始状态时，初始化路径；当源点为非起始状态时，从已有路径中随机选取一条终点作为该源点的路径，然后再在该路径上添加终点即可；最后再对剩下的转换函数进行遍历。具体的伪代码如下：

```
Input: State machine SM = (S, S0, Σ, G, Tran)
Output: PathList, the set of generated path, each is a sequence of
        states and transitions;
Intermediate data: vq, queue to store the current states, in which visi-
ted nodes
if(t. from is reachable and t is not covered by PathList){
   Select a random path p, p's last node is t. from;
   new Path p' = p. append(t. from). append(t);
   PathList. add(p);
}
```

以图 11 所示的状态机为例（其中 0 为初始状态）来验证该算法的正确性。

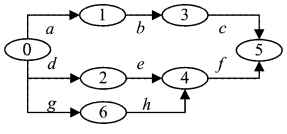

图 11　状态机样例

具体实验结果如下：

1）状态机输入如图 12 所示。

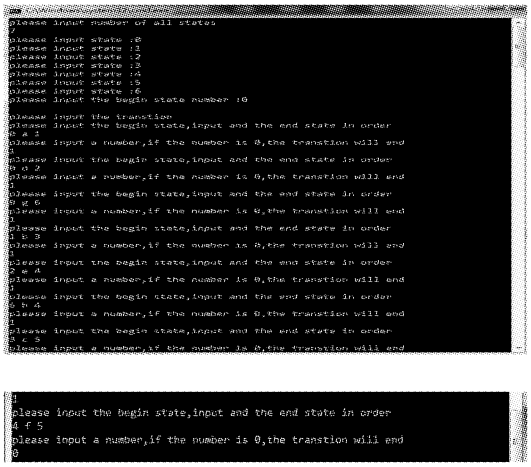

图 12　输入

2）路径输出如图 13 所示。

图 13　输出

3.1.2　测试用例生成

通过上述结果可知，无论终点是什么，利用该算法都能求出源点到该终点的所有路径。

针对该电力模型，即可利用该算法求出某一状态到另一状态的所有路径，对应的测试用例也将随之生成。

3.2　构造动作-脚本映射表

构造动作-脚本映射表较为关键，因为有了映射表才能自动生成测试脚本。对于测试样例中的某个动作（如点击按钮等），通过查表即可得出相应的 API。

该表的设计是这次研究的重点。首先选定测试工具，不同的测试工具所对应的脚本语言是不一致的，这次研究只针对某一个测试工具，而非所有的。为了得到不同动作的脚本 API，需要先熟悉该脚本语言的特色，进而思考如何编写该脚本。由于目前的测试工具都已具备录制回放功能，因此可以方便地利用该功能来了解该脚本语言，通过不断尝试各种动作得到相匹配的脚本 API，某些 API 可能还会包含一些参数，需要通过查阅各种资料来了解该参数的含义。若无录制回放功能，则需要通过不停地进行尝试来得出不同动作的 API。

通过上述方法构造出的动作-脚本API 映射表可能不是完备的，由于整个过程是通过查阅资料和尝试动作得出的，但其中已经包含了多数常见的动作，因此后期还需要不断的完善，每当遇到一个新的 API 和动作，就将其增加到该映射表中。映射表构建完成后，测试人员的测试工作将变得十分简单，所有测试工作都可以自动完成。

3.2.1　IOS 平台

IOS 的自动化测试工具是 UIAutomation，因此本研究的目标是构建 UIAutomation 工具下的动作-脚本 API 映射表，新入门人员需要先熟悉该工具下的脚本语言。

由于对较多 API 都不了解，因此先利用该工具的录制回放功能来熟悉更多的 API。

首先需要将工具切换到编写脚本的界面，然后点击中间下方的红色按钮即可开始录制，测试工具自动在模拟器中启动被测试的应用，此时在模拟器上进行一些操作，相应的脚本会在中间代码区域自动产生，从而可知每个动作所对应的 API，如在模拟器上点击一个按钮，中间代码将会对应地自动产生一句代码。

经过尝试之后发现，可能有些自动产生的脚本不是完全正确的，如在录制时针对特定动作产生的特定脚本 API 并不会考虑整个应用的连贯执行，但在实际的执行中动作之间还需要间隔时间等。因此在实践中有必要将自动产生的脚本再进行回放，以确认错误的 API。

经过各种动作的尝试之后，最终得出动作-脚本 API 映射表，如表 1 所列。由于其是人工尝试得出的，因此还不够完善，后期将不断对其进行完善。

表 1 中的参数说明如下：Target 代表目标程序，Window 代表当前窗口（大部分程序中的组件都是在窗口下操作的，因此若想获得组件，则须先获得窗口），nevigation_Bar 表示上方导航栏，tab_Bar 表示下方导航栏（导航栏和窗口是平级的），tableView 代表表格，slider 表示滑动条。

表 1 IOS 动作-脚本映射表

动作	脚本函数	参数描述
获取目标程序	UIATaget. localTarget()	
获取当前窗口	Target. frontMostApp(). mainWindow()	
普通按钮点击	Window. buttons()["xxx"]. tap()	"xxx"表示按钮在数组中的下标，也可以表示按钮的名称
等待系统反应(延迟执行下条指令)	Target. delay(x)	x 表示延迟的时间
等待系统反应(指定超时时间)	Target. pushTimeout(x)	x 表示指定的超时时间
等待系统反应(超时弹出)	Target. popTimeout()	
程序后台延时	target. deactiveAppForDuration("xxx")	"xxx"表示延时的时间值
获取上方导航栏	target. frontMostApp(). navigationBar()	
点击上方导航栏左侧按钮	nevigation_Bar. leftButton(). tap()	
点击下方导航栏右侧按钮	nevigation_Bar. rightButton(). tap()	
获取下方导航	Target. fronMostApp(). tabBar()	
下方导航按钮点击	tab_Bar. buttons()["xxx"]. tap()	"xxx"表示按钮的名字
获取下方导航中被点击的按钮的名字	Tab_bar. selectedButton(). name()	
获取当前窗口的表格	Window. tableViews()["xxx"]	"xxx"表示下标或名字
点击表格的某一单元	tableView. cells()["xxx"]. tap()	"xxx"表示下标
表格中的拖动操作	tableView. scrollDown() tableView. scrollUp() tableView. scrollLeft() tableView. scrollRight()	
定位到表格中的某个单元	tableView. cells()["xxx"]. scrollToVisible()	"xxx"表示需要定位到的单元的下标
利用 scrollView 定位(不确定名字)	tableView. scrollToElementWithPredicate("name begins with 'xxx')	"xxx"表示名字的开头
利用 scrollView 定位(确定名字)	tableView. scrollToElementWithName("xxx")	"xxx"表示名字
利用 scrollView 定位(值)	tableView. scrollToElementWithValueForKey("aaa""name")	"aaa""name"分别表示 k 和值
根据标题匹配组件	tableView. cells(). firstWithName("xxx")	"xxx"表示标题名称
根据 K 值匹配组件	tableView. cells(). firstWithValueForKey("aaa","name")	"aaa""name"分别表示 k 和值
根据名称查询组件	tableView. cells(). firstWithPredicate("name begins with 'xxx'")	"xxx"表示要查询的组件名称的前几个字母
获取当前窗口的滑动条	Window. sliders()["xxx"]	"xxx"表示下标
滑动条的拖动	Slider. dragToValue("xxx")	"xxx"表示拖动到的数据值
界面任意位置点击(单击)	Target. tap({x:a,y:b})	'a''b'分别表示坐标(x,y)的具体值
界面任意位置点击(双击)	Target. doubleTap({x:a,y:b})	'a''b'分别表示坐标(x,y)的具体值
界面任意位置点击(双手指点击)	Target. twoFingerTap({x:a,y:b})	'a''b'分别表示坐标(x,y)的具体值
给文本框等赋值	window. textFiels()[0]. setValue("xxx")	"xxx"表示赋的值
指定坐标处的放大	target. pinchOpenFromToForDuration(({x:a,y:b},{x:c,y:d},e))	(a,b),(c,d)分别表示坐标,e 表示指定时间限制
指定坐标处的缩小	target. pinchCloseFromToForDuration(({x:a,y:b},{x:c,y:d},e))	(a,b),(c,d)分别表示坐标,e 表示指定时间限制
拖拽坐标位置	target. dragFromToForDuration(({x:a,y:b},{x:c,y:d},e))	(a,b),(c,d)分别表示坐标,e 表示指定时间限制
快速拖拽坐标位置	target. flickFromTo({x:a,y:b},{x:c,y:d})	(a,b),(c,d)分别表示坐标
设置设备方向	UIATraget. setDeviceOrientation	
设置界面方向	UIATarget. interfaceOrientation	

3.2.2 Android 平台

本文研究使用 Robotium 作为 Android 平台的自动化测试工具。

经过资料查询得知，Robotium 作为一个测试框架进行 Android 自动化测试时与 IOS 上的 UIAutomation 工具不同，利用该框架进行测试时实际是新建一个测试工程，然后将该框架倒入工程中，从而通过新建测试类来进行自动化测试。

上述测试类是关键内容，与 IOS 上的脚本相同。测试类的编写过程如下。

1)该测试类要继承 ActivityInstrumentationTestCase2〈测试类类名〉，一般情况下该测试类类名都是 MainActivity；

2)创建 Solo 的私有对象 solo，对 UI 上的测试都是通过该对象来完成的；

3)构造方法中的 super("包名"，测试类类名. class)；

4)SetUp 方法中令 solo＝new Solo(getInstrumentation()，getActivity())；

5)执行 testUI 方法，调用 solo 的各种 API 来进行应用程序的自动化测试。

通过上文可知本研究的主要目标是 solo 类的 API 的整理，整理后的动作-脚本映射 API 表如表 2 所列。

表 2 Android 动作-脚本映射表

动作	API	参数描述
普通按钮点击	clickOnButton(int index) clickOnButton(java. lang. String name)	"index"表示按钮在数组中的下标，"name"表示按钮的名称
等待时间	Sleep(int time)	"time"表示延时的时间值，单位为 ms
模拟返回按键	goBack()	
清除输入框的内容	clearEditText(int index)	"index"表示输入框在数组中的下标
在输入框中输入内容	enterText(int index,java. lang. String text)	"index"表示输入框在数组中的下标，"text"为输入的内容
随机点击屏幕	clickOnScreen(float x,float y)	"x""y"分别表示屏幕上的坐标
点击内容	clickOnText(java. lang. String text)	"text"表示要点击的内容
点击视图	clickOnView(android. view. View view)	"view"表示要点击的视图

（续表）

动作	API	参数描述
点击输入框	clickOnEditText(int index)	“index”表示输入框在数组中的下标
点击菜单栏	clickOnMenuItem(java. lang. String text)	“text”表示要点击的菜单栏
拖动	Drag(float fromX, float toX, float fromY, float toY)	“fromX”“fromY”表示起始点坐标，“toX”“toY”表示终止点坐标
获取按钮	getButton(int index)	“index”表示按钮在数组中的下标
返回当前 activity	getCurrentActivity()	
返回 button 列表	getCurrentButtons()	
获取进度条	getCurrentProgressBars()	
获取滚动条	getCurrentScrollViews()	
获取滑动控件	getCurrentSlidingDrawers()	
返回最高级的 view	GetTopParent(android. view. View view)	“view”表示要返回的视图
向下滚屏	scrollDown()	
向上滚屏	scrollUp()	
向下滚屏到指定位置	scrollDownList(int index)	“index“表示指定位置
向上滚屏到指定位置	scrollUpList(int index)	“index“表示指定位置
水平滚动	scrollToSlide(int side)	“side”表示 right 或者 left
设置屏幕方向	setActivityOrientation(int orientation)	“orientation”表示屏幕显示的方向
设置进度条位置	setProgressBar(int index, int progress)	“index”表示进度下标，“progress”表示目标位置
设置滑动模块状态	setSlidingDrawer(int index, int status)	“index”表示滑动条下标，“status”表示目标状态
获取下拉菜单	getCurrentSpinners()	
拉下菜单项	pressMenuItem(int index)	“index”表示菜单下标
点击某个菜单项	pressSpinnerItem(int spinnerIndex, int itemIndex)	“itemIndex”表示点击的菜单项，其中正数表示向下，负数表示向上
查找按钮	searchButton(java. lang. String text)	“text”表示按钮的名字
查找文字	searchText(java. lang. String text)	“text”表示文字内容
等待一个对话框关闭	waitForDialogToClose(long timeout)	“timeout”表示等待的时间

3.3　自动化测试脚本的生成

通过上述两个步骤自动生成测试脚本。第一步得到了测试动作序列，第二步得到动作-脚本 API 的映射表，那么通过查表即可得到测试动作序列中的动作所对应的 API，自动产生了相应的脚本，具体过程如下：

测试动作序列→按次序取出单个动作→查映射表得到对应 API→将对应的动作更换为脚本 API→测试脚本

3.4　结果展示

以本研究选择的电力应用为例，利用提出的路径算法得到源点到某个终点的所有路径，路径上的即为动作序列（当一个状态机十分庞大时，可以先选取其中的一部分来测试），在所有的动作测试序列中选择一个测试动作序列如下：

点击“支付购电”→选择“时间地区”→点击“下一步”→点击“下一步”→点击“确定”

3.4.1　IOS 平台

通过查阅 IOS 平台上的动作-脚本映射表得到对应的 API，然后将这些 API 转换为连续可执行的脚本放入 Xcode 的集成框架 UIAutomation 中运行，从而该应用开始按照预期的执行顺序进行自动化测试，结果如图 14 所示。

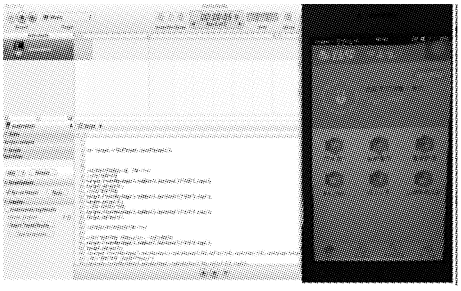

图 14　运行界面

3.4.2　Android 平台

与 IOS 平台上的执行过程一致，首先查阅 Android 平台上的动作-脚本映射表得到对应的 API，然后利用 Android 自动化测试框架 Robotium 生成执行脚本并开始运行，同时应用开始进行自动化测试，运行界面如图 15 所示。

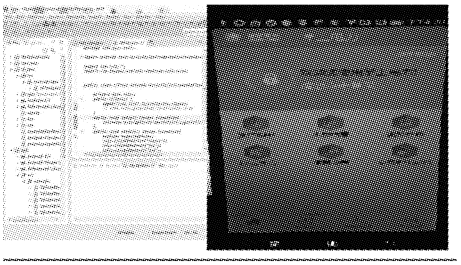

图 15　运行界面

3.5　小结

同样地，针对任何应用和平台，且也不局限于所举例子，只要给出应用对应的模型，即可得到测试动作序列。本次研究中已得出了动作-脚本映射表，因此只需通过查表即可得到对应的脚本 API，从而自动产生测试脚本，这为开发者的测试工作带来了极大的便利，也能够让该应用的质量得到提高。

结束语　本文主要研究模型驱动的移动应用开发和测试，从开发者的角度出发，能够让开发者根据模型对一个移动应用进行自动化测试，即不需要测试人员进行繁杂的手动操作等，这将方便开发者的测试，也使得移动应用质量得到提高。本研究提出了一个动作-脚本 API 映像表以及源点到终点的所有路径算法，针对任何应用只需根据模型即可得到完备的测试动作序列，从而可通过查表来实现脚本的自动生成，这将为移动应用的开发测试节省很多人力物力。本次研究还涉及了多个平台，包括现在主流的 Android 和 IOS 平台。

移动应用测试的自动化是移动应用测试的大趋势，越来越多的企业都有这样的需求，也有越来越多的企业将目光投向该领域，因为用户越来越多，而且用户对移动应用的要求越

（下转第 245 页）

Trust worthy Software Architecture[J]. Chinese Journal of Computers, 2010, 33(5): 890-899. (in Chinese)
赵会群，孙晶. 面向服务的可信软件体系结构代数模型[J]. 计算机学报，2010，33(5)：890-899.

[9] ALVES A, ARKIN A, ASKARY S, et al. Web services business process execution language version 2. 0[EB/OL]. (2007-04-11)[2012-08-19]. http://docs. oasis-open. org/wsbpel/2. 0/OS/wsbpel-v2. 0-OS. html.

[10] SUN J, LI D F. Study of algorithm on evolution for BPEL structure[J]. Application Research of Computer, 2016, 33(9): 1-6. (in Chinese)
孙晶，李东方. 一种 BPEL 结构演化算法研究[J]. 计算机应用研究，2016，33(9)：1-6.

[11] ZHAO H Q, YIN C R. Research of model checking method and technology guided by testing purpose[J]. Application Research of Computer, 2015, 32(8): 2387-2390, 2394. (in Chinese)
赵会群，殷朝冉. 测试目的引导的模型检测方法与技术研究[J]. 计算机应用研究，2015，32(8)：2387-2390，2394.

(上接第 239 页)

来越高。希望本文所提方案能够缓解目前以手工为主的繁琐的测试过程中的由 Android 设备、IOS 设备系统和硬件的多样性造成的麻烦。

此外，在未来的工作中还需进一步完善已有的工作，在原有的基础上增加测试的项目，如安全性测试、可靠性测试，完善动作-脚本 API 映射表，对路径算法不断进行完善等，从而使之真正摆脱人工干预，功能更加强大。在研究工作中，可以将其不断扩展到资源泄漏测试、能耗测试等多个方面。

参 考 文 献

[1] 中国互联网信息中心. 第 37 次《中国互联网络发展状况统计报告》[OL]. http://www. cnnic. net. cn/hlwfzyj/hlwxzbg.

[2] LUO Z J, WU W J, YANG M. Mobile Internet: Terminal Device, Networks and Service[J]. Chinese Journal of Computer, 2011, 34(11): 2029-2051. (in Chinese)
罗军舟，吴文甲，杨明. 移动互联网：终端，网络与服务[J]. 计算机学报，2011，34(11)：2029-2051.

[3] ATKINSON C, KUHNE T. Model-driven development: a metamodeling foundation[J]. IEEE Software, 2003, 20(5): 36-41.

[4] MELLOR S J, CLARK T, FUTAGAMI T. Model-driven development: guest editors' introduction[J]. IEEE Software, 2003, 20(5): 14-18.

[5] BALASUBRAMANIAN K, GOKHALE A, KARSAI G, et al. Developing applications using model-driven design environments [J]. Computer, 2006, 39(2): 33-40.

[6] BUTLER M. Android: Changing the mobile landscape[J]. Pervasive Computing, IEEE, 2011, 10(1): 4-7.

[7] GOADRICH M H, ROGERS M P. Smart smartphone development: iOS versus Android[C]// Proceedings of the 42nd ACM Technical Symposium on Computer Science Education. ACM, 2011: 607-612.

[8] KOCHAN S G. Programming in Objective-C[M]. Addison-Wesley Professional, 2011.

[9] ABOGHARAF A, PALIT R, NAIK K, et al. A methodology for energy performance testing of smartphone applications[C]// 2012 7th International Workshop on Automation of Software Test (AST). IEEE, 2012: 110-116.

[10] ANAND S, NAIK M, HARROLD M J, et al. Automated concolic testing of smartphone apps[C]// Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering. ACM, 2012: 1-11.

[11] ZADGAONKAR H. Robotium Automated Testing for Android [M]. Packt Publishing Ltd, 2013.

[12] 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.

[13] HU C, NEAMTIU I. Automating GUI testing for Android applications[C]// Proceedings of the 6th International Workshop on Automation of Software Test. ACM, 2011: 77-83.

[14] MORZAEO N, MALEK S, PĂSĂREANU C S, et al. Testing Android apps through symbolic execution[J]. ACM SIGSOFT Software Engineering Notes, 2012, 37(6): 1-5.

[15] BORMAN M. Developing, and testing, a theoretical framework for inter-organisational systems (IOS) as infrastructure to aid future IOS design[J]. Information Systems and e-Business Management, 2006, 4(4): 343-360.

[16] PENN J. Test iOS Apps with UI Automation: Bug Hunting Made Easy[M]. Pragmatic Bookshelf, 2013.

[17] MELLOR S J, BALCER M, FOREWORD B J I. Executable UML: A foundation for model-driven architectures[M]. Addison-Wesley Longman Publishing Co., Inc., 2002.

[18] LODDERSTEDT T, BASIN D, DOSER J. Secure UML: A UML-based modeling language for model-driven security[M]// 《UML》2002—The Unified Modeling Language. Springer Berlin Heidelberg, 2002: 426-441.

[19] JENSEN C S, PRASAD M R, MØLLER A. Automated testing with targeted event sequence generation[C]// Proceedings of the 2013 International Symposium on Software Testing and Analysis. ACM, 2013: 67-77.

[20] SKIENA S. Implementing discrete mathematics: combinatorics and graph theory with Mathematica[M]. Addison-Wesley Longman Publishing Co., Inc., 1991.

[21] TARJAN R E. Fast algorithms for solving path problems[J]. Journal of the ACM (JACM), 1981, 28(3): 594-614.

[22] TARJAN R E. A unified approach to path problems[J]. Journal of the ACM (JACM), 1981, 28(3): 577-593.

[23] LEI B, WANG L Z, BU L, et al. Robustness Testing for Components Based on State Machine Model[J]. Journal of Software, 2010, 21(5): 930-941. (in Chinese)
雷斌，王林章，卜磊，等. 基于状态机模型的构件健壮性测试[J]. Journal of Software, 2010, 21(5): 930-941.