一种 Android 平台混合应用运行环境①

宋小远 [1,3], 薛云志 [1,2]

[1](中国科学院软件研究所 基础软件国家工程研究中心, 北京 100190)
[2](中国科学院软件研究所 基础软件测评实验室, 北京 100190)
[3](中国科学院大学, 北京 100190)

摘 要: 随着智能终端设备以及移动互联网的发展, 智能手机等设备越来越普遍, 其上的应用也越来越丰富. 如何能够更快、更高效的开发智能终端上的应用成为开发者面临的巨大问题之一. 在分析了当前存在的跨平台混合应用开发的基础上, 设计并实现了一种 Android 平台混合应用运行环境, 利用该运行环境可以实现只用 HTML、CSS 和 JavaScript 开发 Android 应用, 为开发者带来极大的便利. 同时该运行环境兼容桌面系统上的混合应用运行环境, 利用该运行环境开发的应用经过不同屏幕适配同时可以运行在桌面系统上.

关键词: Android 平台; 混合应用; 运行环境; 跨平台; web 技术

Hybrid Application' Runtime Environment on Android Platform

SONG Xiao-Yuan[1,3], XUE Yun-Zhi[1,2]

[1](National Engineering Research Center for Fundamental Software, Institute of Software, the Chinese Academy of Sciences, Beijing 100190, China)
[2](Laboratory for Fundamental Software Testing, Institute of Software, the Chinese Academy of Sciences, Beijing 100190, China)
[3](University of Chinese Academy of Science, Beijing 100190, China)

Abstract: With the development of intelligent terminals and mobile internet, smart phones become more and more popular. Consequently, applications on them also become more and more diversified. Developers are facing a big problem of how to develop intelligent terminals' applications more quickly and efficiently. This essay analyzes and studies currently existing developments of cross-platform' hybrid application, based on which a hybrid applications' runtime on Android Platform is designed and implemented. Taking advantage of this runtime, developers can develop Android applications only via HTML, CSS, JavaScript. It brings much convinces to developers. This runtime environment is compatible with hybrid applications' runtime on desktop system. Applications developed via this runtime can be operated on desktop system after different screen adaptation.

Key words: Android platform; hybrid application; runtime; crosswalk platform; web technology

1 引言

随着移动终端硬件技术以及移动互联网的发展, 移动终端已经从功能性向智能性发展, 使得手机和平板电脑可以运行更多、内容更加丰富的应用. 对于应用开发者来说, 其关心的首要问题就是更加快速、高效的开发运行在这些设备上的应用[1].

但是由于各大公司的封闭, 各大主流的设备的操作系统之间互不兼容, 并且缺少统一的接口去实现跨平台应用的开发, 如果要开发一款运行在不同设备上的应用, 则需要针对不同的平台分别进行开发、测试和维护, 导致了资源浪费和成本提高等问题[2]. 因此, 一个跨平台的应用软件开发解决方案, 对提高智能设备应用的开发效率具有极其重要的作用.

目前, 对于跨平台开发应用软件的研究有很多, 像 Microsoft、Intel 公司都在进行跨平台开发方面的研究. 同时市场上也存在一些支持跨平台开发的解决方

① 基金项目:中国科学院先导专项(XDA06010600);国家自然科学基金青年基金(61402454)

收稿时间:2016-01-04;收到修改稿时间:2016-02-29 [doi:10.15888/j.cnki.csa.005352]

案和产品, 比较有名的有 NW.js、Crosswalk、PhoneGap 等[3]. 他们大多是采用 Web 技术来开发 Web 应用. 首先 HTML5 这类 Web 技术的学习成本低, 利用 Web 技术开发应用可以提升开发者的效率[4]. 同时 Web 技术具有天生的跨平台性, 可以很好地解决跨平台的问题, 现在利用 Web 技术来解决跨平台开发的问题是人们越来越热衷的解决方案[5]. 但是在移动平台, 利用 Web 技术开发移动应用还存在一些问题, 例如本地功能调用受限、读取用户数据的安全威胁、运行环境单一等等.

本文在分析了当前存在的跨平台解决方案及有关跨平台的产品的基础上, 设计并实现了一种 Android 平台混合应用运行环境—CAN(Chromium+ Android+ Node.js), 利用 CAN 运行环境实现跨平台应用软件的开发. 该 CAN 运行环境可以实现只用 HTML、CSS 和 JavaScript 开发 Android 应用, 充分利用了 Web 技术的学习成本低以及跨平台性的优点. 同时 CAN 运行环境兼容 NW.js 桌面系统混合应用运行环境, 所以利用 CAN 开发的混合应用, 经过不同屏幕的适配可以同时运行在 windows 系统、Linux 系统、Mac OSX 系统和 Android 系统上.

本文的组织结构如下: 第二节介绍有关跨平台开发应用软件方面的相关研究背景及意义, 第三节介绍混合应用运行环境 CAN 整体设计与实现, 第四节介绍利用 CAN 运行环境开发的应用过程及相关实验, 第五节是论文的总结, 提出了 CAN 运行环境当前存在的不足以及未来的展望.

2 研究背景及意义

现在人们使用的智能设备种类越来越丰富, 常用设备已经不仅仅局限于计算机, 还包括智能手机、平板电脑等. 但是由于各大公司的封闭, 不同种类的设备运行着不同种类的操作系统, 例如针对计算机的 Windows、OSX、Linux 等, 针对移动终端的 iOS、Android、Windows Phone 等. 不同操作系统之间互不兼容, 无法基于统一的接口去实现应用软件的开发, 如果要开发一款应用软件运行于不同的设备上, 就需要针对不同的设备分别进行开发、测试和维护, 极大的提高了开发成本, 造成了资源的浪费[6].

因此一种跨平台的应用软件开发解决方案是极其必要的. 它使得开发者开发一款应用软件不依赖于具体的操作系统或者硬件, 开发者的一套代码不用修改, 只需要少量的适配就可以运行在不同的操作系统上[7]. 同时开发的应用与本地应用在运行方式和使用形式没有任何的区别. 这样能够极大的提高开发者的效率, 缩小应用软件开发的成本, 同时用户可以获得更加丰富的应用.

现在针对跨平台开发的研究有很多, 针对移动智能终端跨平台方面的研究和产品则更加丰富, Microsoft、Google、Intel 等公司都在进行相关方面的研究.

近年来, 微软一直积极的开发 Windows Phone 兼容 Android 应用的功能. 现在谷歌的浏览器 Chrome 可以通过安装插件来运行 Android 应用, 这样 Android 应用就可以运行在安装有 Chrome 浏览器的桌面操作系统上. 因特尔公司的跨平台开源项目 NW.js 是 Github 上最活跃的开源项目之一, 目前已经有很多应用是基于 NW.js 进行开发的. 从中可以看出, 许多大公司都在进行跨平台方面的研究.

在大公司正在进行跨平台开发方面的研究的同时, 市场上也有很多支持跨平台开发的运行环境和开发框架, 它们大多数是采用 Web 技术来实现应用的开发, 充分利用了 Web 技术学习成本低、开发效率高和跨平台的优点[8]. 比较有名的有 NW.js、Crosswalk 和 PhoneGap 等等.

NW.js 是桌面系统上的混合应用运行环境, 它是将 Node.js 和 Chromium 相结合, 实现只用 Web 技术来开发本地桌面应用[9]. 现在 NW.js 只是支持 Windows、Linux 和 Mac OSX 操作系统, 不支持移动终端, 但是智能终端应用的开发是现在应用开发不可缺少的一部分.

PhoneGap(Cordova)是一套开源的、免费的移动终端跨平台开发框架. 它是根据系统自带的 WebView 视图控件异步调用本地 API 的原理来实现 JavaScript 与系统 API 的调用功能, 从而实现只用 Web 技术开发移动应用[10]. 但是 PhoneGap 依赖系统自身的 WebView 进行渲染, 不同版本的 WebView 的渲染性能和功能受到限制[11]. Crosswalk 是开源的智能移动终端混合运行环境, 它是将 Chromium 的内核进行封装, 在获取本地系统调用方面兼容 PhoneGap, 支持 PhoneGap 的插件扩展机制[12]. 但是 Crosswalk 现在只能开发移动终端和特定版本的 Linux 操作系统上的混合应用, 对于其他系统上的应用开发并不支持.

综上所述, 现存的一些跨平台开发的运行环境和开发框架都存在一些问题, 像支持环境不足、渲染引擎性能差等. 本文设计的CAN混合应用运行环境能够很好地解决上述问题, 利用CAN混合应用应用可以快速的开发移动应用, 同时开发的应用经过不同界面的适配, 可以运行在主流的桌面系统上(Windows + Linux + OSX). 同时 CAN 混合应用运行环境采用 Chromium 的 Blink 渲染引擎, Chromium 的 Blink 渲染引擎是公认的最快的渲染引擎之一, 可以极大的提高渲染引擎的性能.

3　运行环境设计与实现

实现 CAN 混合应用运行环境需要完成两个基本功能: 一、实现高效的 Web 界面渲染引擎, 能够渲染HTML、CSS页面, 同时还需要JavaScript解析引擎, 解析 Web 页面中的 JavaScript 语句; 二、封装不同平台的系统调用, 使得开发者能够通过统一的系统调用获取到不同平台的数据和系统调用[13]. 实现这两个基本功能之后开发者就可以通过一套代码, 少量适配, 打包成不同平台的应用并且在不同的平台上运行. 其运行框架图如图 1 所示.

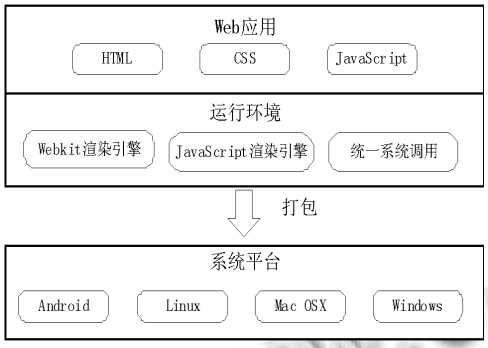

图 1　CAN 运行环境框架

下文将会首先介绍 CAN 运行环境的整体架构, 然后分别介绍实现 CAN 运行环境两个基本功能的解决方案, 以及在实现的过程中遇到的技术难点.

3.1 总体架构

CAN 运行环境采用将 Node.js 和 Chromium 的 Content 模块相结合的方式来实现运行环境的两个基本功能. Node.js 是 JavaScript 的本地运行环境, 它通过模块机制封装不同平台的系统调用, 使得开发者可以通过 JavaScript 语言获取本地系统数据和系统调用. Chromium 的 Content 模块采用 Blink 渲染引擎和 V8 解析引擎, 能够快速的渲染 Web 界面并解析 Web 界面中的 JavaScript 语句. 将两者结合, 可以利用 Chromium 的 Content 模块对 Web 页面进行渲染, 利用 Node.js 获取不同平台的数据和系统调用, 从而实现 CAN 混合应用运行环境的两个基本功能. 其总体架构图如图 2 所示.

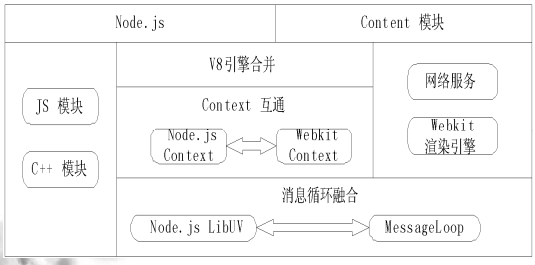

图 2　CAN 运行环境总体架构图

图 2 左侧是 Node.js 的组成部分, 包括 JavaScript 模块、C++模块、Node.js 中 JavaScript 运行环境以及 Node.js 的 LibUV 消息循环机制. 图 2 右侧是 Chromium 的 Content 模块部分, 包括提供的网络服务, Blink 渲染引擎, Blink 引擎中的 JavaScript 运行环境以及 MessageLoop 消息循环. 从图中可以看出完成 CAN 运行环境需要完成两个技术难点: 一、在 Render 进程中启动 Node.js, 同时将 Node.js 的事件循环与 Render 进程的事件循环机制融合. 二、将 Node.js 和 Content 模块的 V8 引擎合并为一个引擎, 并且将两者的 JavaScript 运行环境 Context 打通. 完成上述功能之后, 就可以实现只用 Web 技术开发 Android 应用, 同时通过少量的适配, 就可以将此应用打包成不同桌面系统上的应用.

3.2 Render 进程中启动 Node.js

Chromium 的 Content 模块是一个多进程的架构, 由 Browser 进程和 Render 进程组成, 其中 Browser 进程负责与操作系统相关联的部分, 如网络数据读取、文件访问等功能, Render 进程负责 Web 页面的渲染和 JavaScript 语句的执行[14]. Node.js 采用单线程异步非阻塞 I/O 模型, 主线程和线程池相结合的方式来运行, 其中主线程执行 Node.js 的逻辑部分, 线程池来完成 Node.js 的异步任务.

CAN 运行环境要获得 Android 系统的数据和系统调用, 需要使得 Content 模块中的 DOM 元素能够调用 Node.js 中的 JavaScript 对象和函数, DOM 元素在 Render进程中, Node.js的JavaScript对象和函数在自己

的进程中, 所以要想实现 Content 模块中的 DOM 元素调用 Node.js 中的 JavaScript 对象和函数, 需要在 Render 进程中启动 Node.js. 修改 Render 进程启动过程部分代码, 在创建 Render 进程的时候, 同时创建运行 Node.js 中 JavaScript 语句的 Context 执行环境和 Node.js 的 LibUV 循环.

Render 进程有自己的事件驱动机制, Node.js 利用 LibUV 来实现自己的事件循环机制, 在一个进程中只需要一个事件循环机制. 所以需要在 Render 进程中启动 Node.js 之后, 设计并实现一种消息循环机制, 将 Render 进程的消息循环和 Node.js 的事件循环合到一起, 使得两者的事件都能够得到及时地处理, 任何一方都不会处于饥饿状态.

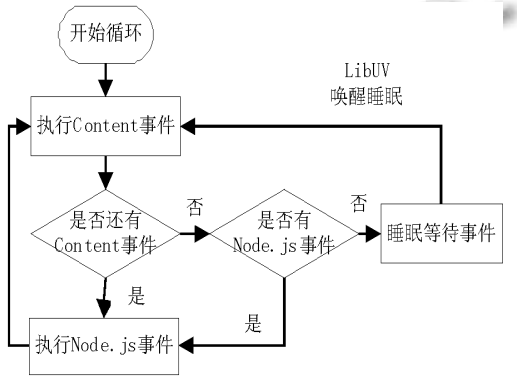

图 3 消息循环图

消息循环设计图如图 3 所示, 利用 Node.js 的 LibUV 事件驱动机制来实现 CAN 运行环境 Render 进程的消息循环, 每当 Render 进程执行完一个 Content 模块中的事件之后, 需要执行一次 Node.js 的事件循环来处理 Node.js 的事件. 当 Render 进程和 Node.js 都没有要处理的事件时, 该事件循环处于睡眠挂起状态. 当两者中有一个出现要处理的事件时, 事件循环就会启动, 其中循环的挂起和启动都是用 LibUV 来实现的.

3.3 Content 模块 Context 与 Node.js Context 互通

Context(执行上下文)是 JavaScript 中最重要的一个概念, Context 中定义了 JavaScript 变量和函数有访问权限的其他数据, 然后通过一个变量对象与 Context 关联起来, 不同的 Context 之间不能相互访问对方的对象和函数.

虽然上文已经将 Content 中的 DOM 元素和 Node.js 放到同一个 Render 进程中, 但是 Content 模块中 JavaScript 语句和 Node.js 中的 JavaScript 语句运行在不同的 Context 中, Content 模块中的 JavaScript 语句是不能调用 Node.js 中的对象和函数的. 为了使得两者的 JavaScript 语句能够相互调用, 并且降低 Content 模块和 Node.js 的耦合性, 我们采用将两者的 Context 打通的方式, 这样两者的 JavaScript 语句便可以相互调用.

首先 Content 模块和 Node.js 中都有 V8 解析引擎, 为了将两者的 JavaScript 运行环境打通并且考虑到节省空间和防止编译冲突的问题, 我们需要使得 Content 模块和 Node.js 使用同一个 V8 引擎. 考虑到跨平台的特性, 我们选取 Content 模块中的 V8 引擎作为两者的 JavaScript 解析引擎.

在 Content 模块和 Node.js 的 JavaScript 解析引擎合并为同一个 V8 引擎之后, 为了使得两者的 Context 互通, 需要为两个 Context 设置相同的 SecurityToken, 并且将 Node.js Context 中的全局对象挂载到 Content 模块中的 Context 中. 互通的过程如图 4 所示.

实现上述两部分功能之后, 可以在 Render 进程中启动 Node.js, 两者有共同的事件循环机制. 同时 Render 进程中的 Javascript 语句能够调用 Node.js 的对象和函数. 这样 CAN 运行环境便可以通过 Node.js 来调用 Android 系统的数据和系统调用.

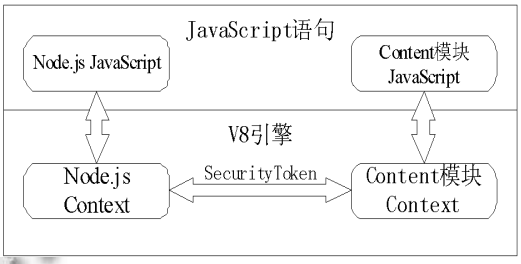

图 4 Context 互通

3.4 CAN 调用 Android 版本的 Node.js 模块

CAN 运行环境是通过调用 Node.js 模块来获取 Android 系统的数据和系统调用. Node.js 的模块分为两部分: JavaScript 模块和 C++模块. JavaScript 模块全部由 JavaScript 语言实现, 其解析执行是利用 CAN 运行环境 V8 解析引擎, 所以可以在 Android 平台上运行. C++模块是一个编译好的二进制动态库文件, 要想使得 C++模块能够在 Android 系统上运行, 需要将其编译成支持 Android 的 arm 二进制动态库文件.

利用 Android NDK 交叉编译的方法, 实现了一个将 C++模块编译成 Android 版本二进制动态库的工具. 在编译 C++模块时, 需要导入 NDK 的路径, 然后将编

译的指令需改为 NDK 中的编译指令, 这样 C++模块便可以编译出 Android 版本的二进制动态库文件, 然后被 CAN 运行环境调用.

4 实验分析

CAN 混合应用运行环境是让开发者使用 Web 语言开发 Android 应用, 同时开发的应用经过少量适配可以运行在桌面系统上. 利用 CAN 混合应用运行环境开发应用的流程图如图 5 所示.

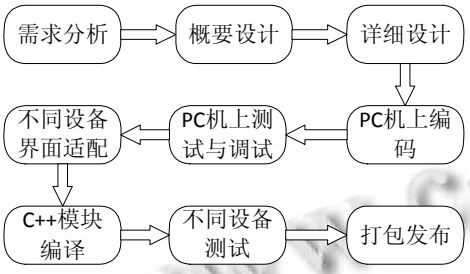

图 5　开发流程

从流程图中可以看出, 利用 CAN 运行环境开发一个应用时, 相关代码的编写、调试与测试主要是在 PC 机上完成, 当需要发布到不同设备, 不同平台的版本时, 只需要对界面进行相应的适配并且对用到的 C++模块进行编译, 同时为了安全, 还需要对应用进行二次测试, 最终打包成不同平台的应用.

在开发应用的过程中, 应用的主界面是在 index.html 中实现, 编写 HTML 文件用的 CSS 文件和 JavaScript 文件放在同一路径下的 css 文件夹和 js 文件夹下, 需要的 Node.js 模块放在 node_modules 路径下, 这样需要的文件都能通过路径找到. 在 PC 上调试好之后, 要打包成 Android 应用, 按照上述路径将源文件

图 6　Linux 系统监视器

放在 assets 文件夹下, 利用 Eclipse 打包即可,产生的 Android 安装包会放在 bin 的目录下.

本文根据上述过程开发了一个系统监视器, 利用此系统监视器可以获得操作系统平台, 架构以及当前系统的进程数等系统信息, 同时能够获得系统 CPU 每一个核的利用率以及内存的利用率信息. 在开发的过程中主要是在桌面操作系统 Linux 上进行编码调试, 然后分别打包成了 Android 版本和 Linux 版本. 最终系统监视器两个版本的应用截图如图 6、图 7 所示.

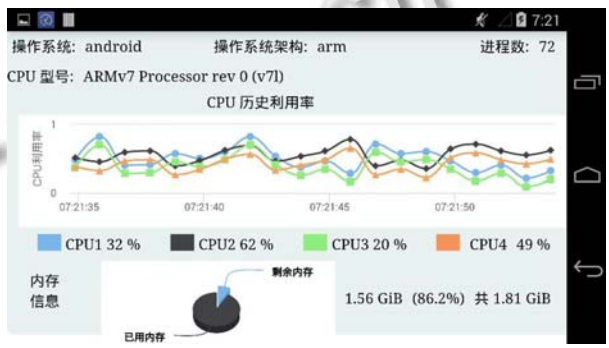

图 7　Android 版本系统监视器

5 总结

随着人们常用设备越来越多, 不同设备的操作系统之间互不兼容, 利用跨平台技术开发混合应用已经成为一种趋势. 由于 Web 语言的跨平台、高效的部署和分发、精密的 UI 布局等特点[15], 利用 Web 技术来实现跨平台的解决方案越来越流行. 现在利用 Web 技术开发跨平台混合应用的框架和运行环境正处于起步阶段[16], 市场上虽然有一些开发框架, 但是都存在一些问题.

本文在分析研究了当前存在的跨平台混合应用开发的基础上, 提出并设计实现了一种 Android 平台的混合应用运行环境—CAN, 它是将 Node.js 和 Chromium 的 Content 模块结合, 利用 Content 模块对应用界面进行渲染, 利用 Node.js 获取 Android 系统的数据和系统调用, 充分利用了 Content 模块中 Blink 引擎的渲染功能与性能优势和 Node.js 大量开源的模块. 利用 CAN 运行环境可以实现只用 Web 语言开发 Android 应用, 同时开发的应用经过少量的适配可以运行在桌面系统上.

但是现在 CAN 运行环境还存在一些不足, 利用 CAN 运行环境开发的 Android 应用会给应用开发包带来额外的体积, 同时 CAN 运行环境对 Android 系统的

系统调用还受到一些限制. 在以后的工作中需要对 CAN 运行环境进行压缩, 实现运行环境共用功能, 将 CAN 运行环境和利用运行环境开发的应用分开打包. 将CAN运行环境作为Android系统的启动项单独安装, 在 Android 上 CAN 运行环境和利用 CAN 运行环境开发的应用的关系就像 office 软件和 word 的关系一样, 这样能够极大地减少应用软件的体积. 同时扩展 CAN 运行环境获取 Android 系统调用的方法, 借鉴 PhoneGap, 实现CAN运行环境中JavaScript和Android系统调用的异步调用功能, 使CAN运行环境的能够获得更多的系统调用, 功能更加完善.

参考文献

1 施伟,王硕苹,郭鸣,等.跨平台移动应用中间适配层设计与实现.计算机工程与应用,2014,(16):39–44.

2 Heitkötter H, Hanschke S, Majchrzak TA. Evaluating cross-platform development approaches for mobile applications. Web Information Systems and Technologies. Springer Berlin Heidelberg, 2012: 120–138.

3 Kao YW, Lin CF, Yang KA, et al. A cross-platform runtime environment for mobile widget-based application. 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC). IEEE. 2011. 68–71.

4 潘晓梦,邓建华,苏厚勤.一种跨平台移动应用方案的研究与实践.计算机应用与软件,2013,30(1):180–182.

5 Pervinder S, Esperanza P, Birgit S, et al. Development of a cross-platform biomarker signature to detect renal transplant tolerance in humans. Journal of Clinical Investigation, 2010, 120(6): 1848–1861.

6 张露.移动多平台跨平台开发工具集的设计与实现[硕士学位论文].武汉:华中科技大学,2013.

7 Humayoun SR, Ehrhart S, Ebert A. Developing mobile apps using cross-platform frameworks: A case study. Human-Computer Interaction. Human-Centred Design Approaches, Methods, Tools, and Environments. Springer Berlin Heidelberg, 2013: 371–380.

8 Johanson D, Andersson K. A cross-platform application framework for HTML5-based e-services. 2014 IEEE 11th Consumer Communications and Networking Conference (CCNC). IEEE. 2014. 194–198.

9 王文睿.node-webkit:HTML5 桌面应用运行环境.程序员, 2014,(1):127–129.

10 http://phonegap.com.

11 潘春华,李俊杰,向花,等.基于 PhoneGap 的智能手机跨平台应用.计算机系统应用,2014(7):106–109.

12 https://crosswalk-project.org.

13 Redda YA. Cross platform Mobile Applications Development: Mobile Apps Mobility. Institutt for Datateknikk Og Informasjonsvitenskap, 2012.

14 徐巍.跨平台移动开发框架的比较分析与实例开发[硕士学位论文].长春:吉林大学,2014.

15 Amatriain X, Arumi P, Garcia D. A framework for efficient and rapid development of cross-platform audio applications. Development of Magnetic Nanomaterials & Devices for Biological Applications, 2008, 14(1): 15–32.

16 Allen S, Graupera V, Lundrigan L. Pro smartphone cross-platform development: iPhone, blackberry, windows mobile and android development and distribution. Apress, 2010.