Application Development for Mobile and Ubiquitous Computing

8. Device Platforms

Dr. Ing. Thomas Springer
Technische Universität Dresden
Chair of Computer Networks
Lecture Structure

Adaptation and Context-awareness

Application Development

Mobile Business Applications

Cross-Platform Development

Mobile Web Applications

Android

iOS

Windows Phone

Mobile Middleware

Disconnected Operations

Mobile Databases

Location-based Services

Communication Mechanisms

Enabling Technologies and Challenges
ANDROID
Android Overview

- Created by: Open Handset Alliance, driven by Google
- First Release: Android 1.0 beta in November 2007
- Current version: 5.0 (Lollipop)
- Target devices: smartphones and tablets from different vendors
- Extensions for further device types: Android Wear, Android TV, Android Auto and Google Glass
- Operating System: Linux Kernel
- Approach: open source (Apache 2.0 license, some libs excluded (e.g. Google Maps), heterogeneous hardware
- Programming: Java
- Development: any hardware
- Development tools: Android Studio, Eclipse Plugin + Android SDK,
Android Architecture

- Application Framework allows reuse and exchange of components
- Libraries
  - Media Libraries supporting many popular formats, (MPEG4, H.264, MP3, AAC, AMR, JPG, and PNG)
  - SQLite - lightweight relational database engine
  - Google Maps support
  - Integrated Browser based on WebKit
  - Optimized graphics libraries (2D library, 3D library based on OpenGL)
- Android RunTime, Dalvik VM
- Linux Kernel (based on version 2.6)
  - threading
  - low-level memory management
  - hardware drivers
  - power management
### Android RunTime vs. Dalvik VM

#### Dalvik VM (previous runtime)
- Alternative Java runtime implementation
- no Sun/Oracle certification
- basically just the syntax of Java
- Optimized for mobile computers
  - memory management
  - every application runs in its own process
  - optimized for many parallel VMs

#### Android RunTime (ART)
- default runtime since Android 5.0
- Ahead-of-time-compilation (AOT)
  - overhead during installation (memory, time)
  - better runtime efficiency and performance
- Improved Garbage Collection and Debugging
- no modifications necessary for standard code
Anatomy of an Android application

- Major building blocks
  - Activities, Services, Intents
- Android Apps run in separate processes
  - Inter-process communication based on AIDL interfaces
  - Used components have to be declared in the Android Manifest file
**Activity**:
- a single screen of the application
- extends the Activity class
- consists of user interface elements (views) that respond to events
- may return a value to another activity
- When a new screen opens, the previous is put onto a history stack.
- Methods of activity reflect lifecycle
- Modular section of an activity, which has its own lifecycle, receives its own input events, and which you can add or remove while the activity is running
- Reusable in different Activities
### Service:
- Means for
  - Performing tasks in background (`startService`)  
  - Expose functionality to other apps (`bindService`)  
- Creation of new Thread in `onCreate()` method  
- Local and Remote Services  
- When connected, communication is done by an interface exposed by the service  
  - based on Java (local) or  
  - based on AIDL (Android Interface Definition Language) for access from other processes (remote).
Intent:
- passive data structure holding an abstract description of an event or an operation to be performed
- Used to
  - Publish events, trigger actions like starting Activity or Service
- consists of an action and data
  - ACTION_VIEW content://contacts/people/1 -- Display information about the person whose identifier is "1".
  - ACTION_DIAL tel:123 -- Display the phone dialer with the given number filled in.
- Intent Filter express ability of component to handle particular intent types
  - `<intent-filter>`
    `<action android:name="android.intent.action.VIEW" />`
  - `</intent-filter>`

BroadcastReceiver:
- receive intents sent by sendBroadcast()
- System broadcasts events to Apps (e.g. battery low, screen off, boot completed)
  - ACTION_BATTERY_CHANGED
  - ACTION_POWER_CONNECTED
Android Manifest

- **AndroidManifest.xml necessary for every application**
- Describes the application’s elements and when they should be initialized or activated
- Includes a list of permissions the application is offering or needing (e.g. for access to network or contacts data); so on installation, the user can grant or deny these.

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.my_domain.app.helloactivity">
    <application android:label="@string/app_name">
        <activity android:name="HelloActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN"/>
                <category android:name="android.intent.category.LAUNCHER"/>
            </intent-filter>
        </activity>
    </application>
</manifest>
```
UI is based on Screens
Activities create and control screens (one activity per screen)
- contain application logic, layout and views
Views as visible elements of UI
- Base class android.view.View
Layouts arrange views on screen
- Base class android.view.ViewGroup, (i.e. layout is group of views)
Android UI Creation

- XML-based description of UIs
- Alternatively UI creation in code
- Support of touch-based interactions
- Event-mechanism to handle interactions
  - E.g. `view.setOnClickListener(callback)`

```xml
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    < textView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="Auf dieser Seite erfassen Sie ei"

    <RadioGroup android:id="@+id/position"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical">
        <RadioButton android:id="@+id/stauAnfang"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Stauanfang" />
        <RadioButton android:id="@+id/stauEnde"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="Stauende" />
        <Spinner android:id="@+id/stauUrsache"
```
Java.net.* APIs can be used
- Provided as part of Android platform
- Base for HTTP connections is HTTPClient

Alternative libraries can be used
- e.g. Apache HttpComponents project

Android.net.ConnectivityManager
- Monitors connectivity state
- Sends broadcast intent if connection state changes
- Provides methods for accessing network state
  - getActiveNetworkInfo()
  - getAllNetworkInfo()
  - getNetworkInfo(int networkType)
Android Persistent Storage

- **SQLite**
  - Local database with SQL and transaction support
  - Database maintained in single file
  - Base class `android.database.sqlite.SQLiteDatabase`
  - execute SQLite queries using the `SQLiteDatabase query()` methods

- **Shared Preferences**
  - Base class `android.content.SharedPreferences`
  - Store private primitive data in key-value pairs

- **Internal Storage**
  - Store private data on the device memory as files.

- **External Storage**
  - Store public data on the shared external storage as files.
**ContentProvider**: Sharing of data between applications

- Abstraction layer on top of DB or files (interface android.content.ContentProvider)
- Content organized like on web server – content URIs for access
  - e.g. content://de.tudrn.exampleprovider/images/content_id
- Implement a standard set of methods for allowing other applications to store and retrieve data.
- ContentProvider implementation for common data types (calendar, contacts, ...)
- Access via ContentResolvers
Android Development Tools

Android Studio
- based on IntelliJ IDE

Eclipse plugin + Android SDK
- Project management
- Device emulator
- Debugger
Google Play Store for app provisioning
Development is free
Developer Registration One-time Fee 25$ for uploading Apps to Play Store
Transaction fee for selling applications in the android market
- 30% of the application price. For example, if you sell your application at a price of $10.00, the fee will be $3.00, and you will receive $7.00 in payment.
IOS
iOS Overview

- iOS (formerly iPhone OS)
- Created by: Apple Inc.
- First Release: June 2007
- Current Version: iOS 8.1.2
- Target devices: iPhone, iPad,
- Further device types: iPod touch, Apple TV, Apple Car Play
- Operating System: based on Mach/BSD Kernel
- Approach: closed source, restricted hardware
- Programming: Objective-C, C, (C++), Swift (since iOS 8)
- Development: Apple Hardware required “Develop for Mac on a Mac”
- Development tools: Xcode + iOS SDK
Apple iPad Air 2
Wi-Fi, BT, UMTS, LTE, GSM,
A-GPS, Accelerometer, Light sensor, Camera
Display 2048 x 1536 px, 264 ppi

Apple iPhone 6, 6+
UMTS, LTE, GSM, Wi-Fi, BT
A-GPS, Compass
Accelerometer, Gyro,
Light Sensor, Barometer
Display 1136 x 750 px, 326 ppi
1st Camera @ 8 MP
2nd Front Camera
Objective-C vs. Java

- Created in the early 80s by Stepstone
- Extends C with object-oriented constructs (Extensions based on Smalltalk)
  - Separate files for interface (header - .h) and implementation (.m) of classes
  - No namespaces/packages
  - Messages to invoke methods
  - Pointer syntax for explicit handling of pointers
  - No method overloading (same method name/different parameter type)
  - No garbage collection (iOS), explicit memory management required (supported by compiler)

Swift

- Programming language influenced by expressivity of scripting languages (elements from Javascript and Ruby)
- No distinction of interface and implementation files (.swift files only)
- No pointers

Swift, Objective-C and C code can be mixed within a project
iOS Architecture

• **Cocoa Touch**: Objective-C APIs for lower layers, e.g., multi touch, camera, web view, accelerometer,...
  ➢ That’s what you use mostly!

• **Media**: OpenGL ES, Core Audio, OpenAL, PDF, PNG, JPG, TIFF, Quartz 2D
  ➢ For performance optimisation

• **Core Services**: Address book, SQL lite, network, location services, threading, NS Object

• **Core OS**: OS X Kernel, BSD, Mach 3.0, file system, power management, security
  ➢ Limited access for developers
Cocoa Touch Frameworks in iOS:

- Foundation (NS... prefix)
  - Data types and structures (Strings, Array, Maps,...)
  - Services & functionality (Date, Calendar, Timer,...)
- UIKit
  - UI related objects (“views”)

Based on design patterns:

- Model-View-Controller – defines the overall app structure
- Delegation - facilitates the transfer of information and data from one object to another
- Target-action - translates user interactions with buttons and controls into code that your app can execute.
**Model View Controller**

- **Model**
  - Encapsulate data and basic behaviour
  - Stores application data (persistently)
- **View**
  - Present information to user
  - Allow users to edit model data
- **Controller**
  - Mediates access of views to models
  - Contains business logic for processing user input
  - Set-up and coordination tasks

![Diagram](image.png)
Delegate Design Pattern

- **Protocols**
  - Similar to interfaces in Java
  - Define methods that are to be implemented by other classes

- **Delegation = mechanism for customization and notification**
  - Principle for customizing: delegation over subclassing
  - Advantages
    - Easier switch at runtime (exchange delegate object vs. instantiation of a different view class)
    - Views are completely reusable
  - View holds reference to controller (defined as Outlet)
  - Controller implements methods of delegate-protocol
  - View invokes methods on its delegate object (controller)
Target-Action Design Pattern

- **Target-Action** – mechanism for notification
  - Actions represent events created by users interacting with the UI (e.g. button pressed)
  - Controller implements action handling
    - Defined by (IBAction)actionName
  - View dynamically invokes methods when actions happen
  - No return values (IBAction compiles to void)
UIApplication (Singleton)
holds reference to delegate
id <UIApplicationDelegate>.delegate
didFinishedLaunching {
init ViewController
}

UIViewController
UIView
defined in .xib or directly created in code

UIApplication Delegate <UIApplicationDelegate>

UIView
defined in .xib or directly created in code

UIViewController
 offsetof UIViewController

UIView
defined in .xib or directly created in code

UIViewController

set to UIWindow

- One Screen per ViewController
- Similar to Android Activities
- AppDelegate handles basic lifecycle calls of Application and inits ViewController for first View

Dr. Thomas Springer
Application Development - 8. Platforms
Based on Interface Builder
- .xib files for describing view hierarchies (.nib is binary form)
- One .xib describes typically one screen
- Created/Edited with Interface Builder
- No direct manipulation of .xib/xml

UIKit class library provides set of predefined Views, ViewControllers and Controls
Tools for iOS Development: Interface Builder

- Visual editor
  - Assembling UI
  - Nib-file generation
- Inspectors for
  - Identity
  - Size, position and layout
  - Attributes
  - Connections
Storyboard is part of InterfaceBuilders

- graphic editing of scenes (single screen)
- segue defines transition between scenes
- enable designing a single universal storyboard with customized layouts for both iPhone and iPad
iOS Libraries

- Root Class: NSObject (defined in foundation lib)
• Part of foundation library
  • `NSURLConnection`, `NSURLRequest`, `NSURLMutableRequest`
Key-Value Storage
- NSUserDefaults (simple Hash synchronized with file)
- Automatically synchronized by system
- stored in App sandbox

Framework Core Data
- Abstraction layer for storing objects persistantly
- Core data objects can be mapped to SQLite or file

SQLite

Files in Sandbox

iCloud
- Key-Value Storage (Hash) with automatic synchronization to iCloud
- Data objects derived from UIDocument, can easily be synchronized with iCloud
iOS Development Tools: XCode

- IDE for Mac and iOS development
  - Manage projects
  - Code editing
  - Building (on device & simulator)
  - Debugging (on device & simulator)
  - Repository management
  - Performance tuning
- Performance analysis tool (incl. graphical display)
  - Memory usage
  - Disk activity
  - Network activity
  - Graphics performance
iOS Developer Program

- Company ($299/year)
- Individual ($99/year)
- Limited to 100 devices
History of Microsoft’s mobile operating systems

Windows CE → Windows Mobile 6.5 → Windows Phone 7

Windows NT → Windows Phone 8 → Windows Phone 8.1
Current Version: 8.1
Created by: Microsoft
First Release: 2012
Target devices:
• primary Smartphones
• Tablets (max. 6 inch)
• unified platform for Smartphones, Tablets and Desktop planned with Windows 10
Operating System:
• Windows NT kernel
Approach: closed source, hardware restrictions defined
Programming: various languages (C#, C++, VB, Javascript)
Development: any hardware, Windows OS required
Development tools: Microsoft Visual Studio
Windows Phone Supported Hardware

- Hardware from multiple vendors supported
- Resource requirements defined by Microsoft
  - CPU
    - Snapdragon CPU
  - Memory
    - 512MB, 1GB or 2GB RAM
  - Resolution
    - 1080p, 720p, WXGA, qHD, FWVGA, WVGA resolution
    - multi-touch display
  - Sensors
    - accelerometer, proximity, ambient light sensors
    - rear-facing camera (VGA)
  - mobile data, WiFi, Bluetooth 4, A-GPS
  - Set of Keys
    - mechanical power, vol+, vol- buttons
    - start, back and search (hardware ore virtual)
Windows Phone 8.1:
- backward compatibility with old App models
  - WP 7: .Net Compact Framework, Silverlight
  - WP 8: Silverlight App model added supporting .Net and XAML
- new App models introduced with WP 8.1
Universal Apps
- new project type for developing Apps for both Windows and Windows Phone
- sharing of code, styles, controls, and other assets
- based on Windows Phone 8.1 API
- set up on Windows Runtime (WinRT)
Windows Phone Silverlight Apps
- Silverlight runtime upgraded from WP7
- WP7 Silverlight Apps runnable
- possibility to retarget old projects to WP 8.1 Silverlight
- Access to almost all APIs of Windows Phone 8.1 API
- several Silverlight exclusive APIs (e.g. clipboard access, VoIP, running under lock screen)
WinJS Apps
- App development based on HTML, Javascript, CSS, etc.
- based on Windows Library for Javascript (WinJS) -> access to native features
Windows Runtime

Windows Runtime Object

- Metadata (winmd)
  - Projection to CLR
  - Projection to Chakra

- C++ App
  - Projection to CLR

- C#/VB App
  - Projection to CLR

- HTML App
  - Projection to Chakra
Windows Programming Model

Windows Runtime

Input, Interaction, & Manipulation

DirectX, Media, & Composition

Windows Kernel Services

C / C++

XAML

C# / VB

HTML / CSS

JavaScript

http://channel9.msdn.com/Events/Build/2014/2-507
Windows Phone App Model

- appx packages
  - zip directories
  - after publishing in the store packages are optimized (e.g. code for fast startup)
- appxbundle
  - include specific packages for each language, resolution, etc.
  - allows users to download minimal package for their device from store
An app’s lifecycle

[Diagram showing the lifecycle of an app: Activated → Running → Suspended → Resuming → NotRunning → Running → Activated]

Windows Phone UI Creation

- Metro design
  - Originated in Windows Media Center and Zune
  - Inspired by public transport signs
  - Objectives
    - Clean, Light, Open, Fast
    - Content, not Crome
    - Integrated hardware and software (integration of phone keys)
    - Soulful and Alive (personalized, continuously updated)
  - Style guides for UI design
Windows Phone UI Creation

- Frame contains several pages for App
  - Web-like navigation
  - All viewed pages on stack (from multiple apps)
  - User can navigate back and forth
UI Definition based on Extensible Application Markup Language (XAML)

```
<Page
    x:Class="HelloWorldApp.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Background="{ThemeResource ApplicationPageBackgroundThemeBrush}"
>

<Page.BottomAppBar
    <CommandBar>
        <AppBarButton Label="Hello" Click="BtnClicked" Icon="World"/>
    </CommandBar>
</Page.BottomAppBar>

<Grid VerticalAlignment="Center" HorizontalAlignment="Center">
    <TextBlock
        x:Uid="HelloWorldTxt" x:Name="TextBlock1"
        Style="{ThemeResource GroupHeaderTextBlockStyle}"
        Foreground="{ThemeResource PhoneAccentBrush}"
        Text="Hello world!" />
</Grid>
</Page>
```
Development Environment

- Visual Studio 2013 Update 2 (IDE provided by Microsoft)
- Windows 8.1 required
Windows Phone Store

- Developer Account
  - Individual – 19$/year
  - Company – 99$/year
  - free for students
## Platform Comparison

<table>
<thead>
<tr>
<th>Vendor</th>
<th>Android</th>
<th>iOS</th>
<th>Windows Phone</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vendor</td>
<td>Google Inc.</td>
<td>Apple Inc.</td>
<td>Microsoft</td>
</tr>
<tr>
<td>Current Version</td>
<td>5.0 (Lollipop)</td>
<td>8.1.2</td>
<td>8.1</td>
</tr>
<tr>
<td>Device hardware</td>
<td>not restricted, various vendors (smartphones, tablets)</td>
<td>Restricted to Apple devices (iPad, iPhone, iPod, Apple TV)</td>
<td>Smartphone and tablets, Various vendors, hardware requirements</td>
</tr>
<tr>
<td>OS</td>
<td>Linux Kernel</td>
<td>Mach/BSD Kernel</td>
<td>WP7 - Win CE, WP8 - Windows NT</td>
</tr>
<tr>
<td>App runtime</td>
<td>Dalvik VM, Android RunTime (ART)</td>
<td>Native code on Mach/ BSD Kernel</td>
<td>Windows RT</td>
</tr>
<tr>
<td>Programming Language</td>
<td>Java</td>
<td>Objective-C, C (C++), Swift</td>
<td>C#, C++, VB, Javascript</td>
</tr>
<tr>
<td>Open source</td>
<td>YES (Apache 2.0 license, some libs excluded, e.g. Google Maps)</td>
<td>NO</td>
<td>NO</td>
</tr>
<tr>
<td>Development restrictions</td>
<td>Any hardware and OS</td>
<td>Apple Hardware and OS required</td>
<td>Any hardware, Windows</td>
</tr>
<tr>
<td>Developer program</td>
<td>one-time fee 25$, development free</td>
<td>Individual $99/year, Company $299/year</td>
<td>Individual 19$/year Company 99$/year free for students</td>
</tr>
</tbody>
</table>
Android

- Arno Becker, Markus Pant: Android 2 – Grundlagen und Programmierung. Dpunkt Verlag, 2. aktualisierte Auflage, 2010
- http://developer.android.com
- http://code.google.com/android

iOS

- Bill Dudney, Chris Adamson: Entwickeln mit dem iPhone SDK. Oreilly, 2010

Windows Phone 7/8

- msdn.microsoft.com/de-de/library/cc656764.aspx
- Patrick Getzmann, Simon Hackfort, Peter Nowak: Entwickeln für Windows Phone 7 – Architektur, Frameworks, APIs, Microsoft Press, 2011