Application Development for Mobile and Ubiquitous Computing

8. Device Platforms

Dr. Ing. Thomas Springer
Technische Universität Dresden
Chair of Computer Networks
Application Development

Mobile Business Applications

Cross-Platform Development

Android
iOS
Windows Phone

Mobile Web Applications

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: 7.0 (Nougat)
- Target devices: smartphones and tablets from different vendors
- Extensions for further device types: Android Wear, Android TV and Android Auto
- Operating System: Linux Kernel
- Approach: open source (Apache 2.0 license, some libs excluded (e.g. Google Maps), heterogeneous hardware
- Programming: Java
- Development: on any hardware
- Development tools: Android Studio, Android SDK
- 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
  - threading
  - low-level memory management
  - hardware drivers
  - power management
Android RunTime vs. Dalvik VM

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

- **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
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
messaging object to publish events or request actions from other components
contains of action and data
  • ACTION_VIEW content://contacts/people/1 -- Display information about the person whose identifier is "1".

three fundamental use cases:
  • start a new instance of an activity
  • start a service
  • deliver a broadcast receivable by any app on the device

two types of intents
  • explicit – specify the receiver component by name
    o component referenced by class name is immediately started
    o e.g.: start a new activity in your app

  • implicit – no specific receiver but a general action to be performed
    o apps can declare intent filters in their manifest
    o if multiple app intent filters match, the user has to pick one
    o e.g.: request another app to show a location on a map
BroadcastReceiver:
- receive intents sent by sendBroadcast()
- System broadcasts events to Apps (e.g. battery low, screen off, boot completed)
  - CONNECTIVITY_CHANGE

```xml
<uses-permission
    android:name="android.permission.ACCESS_NETWORK_STATE" />

<receiver android:name=".receiver.ConnectivityReceiver">
    <intent-filter>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
    </intent-filter>
</receiver>
```

```java
public class ConnectivityReceiver extends BroadcastReceiver {

    public void onReceive(Context context, Intent intent) {
        Log.d(ConnectivityReceiver.class.getSimpleName(), "action: " + intent.getAction());
    }
}
```
**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
Building blocks - Fragments

- 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

Image from http://developer.android.com
### 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).
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>
```
Android UI Creation

- 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" />
    </RadioGroup>
    <Spinner android:id="@+id/stauUrsache"
```
- **Java.net.* APIs**
  - Provided as part of Android platform
  - Base for HTTP connections is HTTPClient

- **Alternative libraries can be used**
  - Google Volley
  - 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 DIE
- Project management
- Device emulator
- Debugger

Eclipse plugin + Android SDK
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 10
- 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: Swift (since iOS 8), Objective-C, C, (C++),
- Development: Apple Hardware required
- Development tools: Xcode + iOS SDK
Apple iPad Pro
Wi-Fi, BT, UMTS, LTE, GSM, A-GPS, Accelerometer, Light sensor, Camera
Display (12,9””) 2732 x 2048, 264 ppi

Apple iPhone 7, 7+
LTE, UMTS, GSM, Wi-Fi, BT A-GPS, Compass
Accelerometer, Gyro, Light Sensor, Barometer
Display (4,7””) 1136 x 750 px, 326 ppi
1st Camera @ 12 MP
2nd Front Camera
Objective-C vs. Swift

- **Objective-C vs. Java**
  - Created in the early 80s by Stepstone
  - Extends C with object-oriented constructs (Extensions based on Smalltalk)

- **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, SQLite, network, location services, threading, NS Object
- **Core OS**: OS X Kernel, BSD, Mach 3.0, file system, power management, security
  - Limited access for developers
iOS Application Anatomy

- 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”)

- Well structured 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
Delegate Design Pattern

- Principle for customizing: delegation over subclassing
  - Subclassing limits reusability of views

- Delegation
  - Tasks and responsibilities are forwarded – delegated – to a 3rd class
  - You “tell” the table view “who” has the knowledge on how to react on events
  - 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)
iOS Application Anatomy

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

- **Not Running**
  - not launched yet or terminated

- **Inactive**
  - running, but not receiving events

- **Active**
  - running and receiving events

- **Background**
  - in the background and executing code
  - app needs to register for allowance (location, audio, download)

- **Suspended**
  - In the background not running code
  - app remains in memory
- 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 (size classes)
- **Root Class: NSObject (defined in foundation lib)**

![Diagram showing the iOS libraries hierarchy](image-url)
iOS Network Communication

- Part of foundation library
  - `NSURLConnection`, `NSURLRequest`, `NSURLMutableViewRequest`
iOS Persistent Storage

- **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
Tools for iOS Development

Instruments

- Performance analysis tool (incl. graphical display)
  - Memory usage
  - Disk activity
  - Network activity
  - Graphics performance
iOS Developer Program

- Company ($299/year)
- Individual ($99/year)
- Without account you can test now with your own device

1. Develop
   Develop your application with the iOS SDK and a wealth of technical resources in the iOS Dev Center. [Learn more](#)

2. Test
   Test and debug your code on iPad, iPhone and iPod touch to finalize your applications. [Learn more](#)

3. Distribute
   Distribute your apps on the App Store and reach millions of iPad, iPhone, and iPod touch users. [Learn more](#)
<table>
<thead>
<tr>
<th>Platform Comparison</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Vendor</strong></td>
</tr>
<tr>
<td><strong>Current Version</strong></td>
</tr>
<tr>
<td><strong>Device hardware</strong></td>
</tr>
<tr>
<td><strong>OS</strong></td>
</tr>
<tr>
<td><strong>App runtime</strong></td>
</tr>
<tr>
<td><strong>Programming Language</strong></td>
</tr>
<tr>
<td><strong>Open source</strong></td>
</tr>
<tr>
<td><strong>Development restrictions</strong></td>
</tr>
<tr>
<td><strong>Developer program</strong></td>
</tr>
</tbody>
</table>
References

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