Application Development for Mobile and Ubiquitous Computing

9. Cross-Platform Development

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

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
Why Cross-Platform Development?

- Strong fragmentation of market for mobile device platforms
- Market changes quickly
- High update rate
  - Device features
  - Platform APIs
- New form factors increase platform diversity, e.g. 7” and 10” tablets, Wearables, …
Why Cross-Platform Development?

Android Project

iOS Project

Windows Phone Project

Mobile Device

Application Framework

Libraries

ART or Dalvik VM
Core libraries

Linux Kernel

Hardware

App .apk

App .ipa

...
High development effort
- Know-how for multiple platforms
- High update rates of technologies
- Customers don’t pay per platform but for project

Short time to market
- Fast support of platform changes/new platforms

High maintenance effort
- Manage multiple code bases
- Avoid inconsistencies

Mobility/Adaptation Support
Write once, run everywhere?

Cross-Platform Project

Mobile Device
- Application Framework
  - Libraries
  - Dalvik VM + Core libraries
  - Linux Kernel
- Hardware

Custom Frameworks
- Cocoa Touch
- UIKit
- Media
- Foundation
- Core Services
- Mach/BSD Kernel
- Hardware

Frameworks
- Silverlight
- XNA
- HTML/JavaScript
- Common Language Runtime
- App Model
- UI Model
- Cloud Integration
- Kernel
- Hardware Foundation
- Hardware
How can we achieve it?

Cross-Platform Development

Mobile Device

Application Framework
- Libraries
- Dalvik VM + Core libraries
- Linux Kernel

Hardware

App .apk

App .ipa

Advisor

Frameworks
- Cocoa Touch
- UIKit
- Media

- Core Services
- Foundation
- Mach/BSD Kernel
- Hardware

Mobile Device

App .ipa

App .ipa

App .xap

Advisor

Frameworks
- Silverlight
- XNA
- HTML/JavaScript

Common Language Runtime

- App Model
- UI Model
- Cloud Integration

- Kernel
- Hardware Foundation

Hardware
A **device platform** is a combination of device hardware, operating system, runtime system, libraries, and frameworks forming a standard execution environment for applications on devices which run that platform.

A device platform has a characteristic set of features:
- Specific „look and feel“ (UI guidelines and interaction concepts)
- App runtime defining app anatomy and lifecycle
- Developer tool chain (programming languages, libraries, APIs)
- Specific app distribution and deployment process

**Native Code** is code which can be directly executed within the standard execution environment of the device platform. It has full access to the platforms libraries and frameworks, possibly also to operating system APIs and hardware features.
Mobile App Types

- Hybride App
- Native Code
- Cross-Platform App (+ Native Ext.)
- Cross-Platform API
- Interpreter
- Device Platform
- Interpreted App
- Web App
- Web browser
What to start with, model or source code?

Design Phase  Implementation Phase  Build Phase  Execution Phase

Model

Source Code

GUI
Logic

Executable Program

Mobile Device

Dr. Thomas Springer

Application Development - 9. Cross-Platform Development
Cross-Platform Development Approaches

- Model-driven Development

<table>
<thead>
<tr>
<th>Design Phase</th>
<th>Implementation Phase</th>
<th>Build Phase</th>
<th>Execution Phase</th>
</tr>
</thead>
</table>

- Model-to-Model-Transformation
- Model-to-Code-Transformation

Model

Source Code
- GUI
- Logic

Executable Program

Interpretation

Mobile Device
Cross-Platform Development Approaches

- Cross-compilation to native code

<table>
<thead>
<tr>
<th>Design Phase</th>
<th>Implementation Phase</th>
<th>Build Phase</th>
<th>Execution Phase</th>
</tr>
</thead>
</table>

- Model
  - Model-to-Model-Transformation
  - Model-to-Code-Transformation

- Source Code
  - GUI
  - Logic

- Cross-Compilation / JIT-Compilation

- Machine Code
  - Execution

- Executable Program
  - Interpretation
  - Mobile Device

Dr. Thomas Springer
Application Development - 9. Cross-Platform Development
Support for abstract runtime

Cross-Platform Development Approaches

Dr. Thomas Springer

Application Development - 9. Cross-Platform Development
Cross-Platform Development Approaches

- **Script languages**

| Design Phase | Implementation Phase | Build Phase | Execution Phase |

- Model-to-Model-Transformation
- Model-to-Code-Transformation

**Source Code**

- Cross-Compilation / JIT-Compilation
- AOT- / JIT-Compilation

**Byte Code**

- Compilation
- Interpretation

**Machine Code**

- Execution

**Executable Program**

- Mobile Device
Cross-Platform Development Approaches

- Translate to other programming languages

<table>
<thead>
<tr>
<th>Design Phase</th>
<th>Implementation Phase</th>
<th>Build Phase</th>
<th>Execution Phase</th>
</tr>
</thead>
</table>


Model-to-Code-Transformation → Model

GUI Logic

Executable Program

Dr. Thomas Springer

Application Development - 9. Cross-Platform Development
Cross-Platform Development Approaches

**Design Phase**
- Model-to-Model-Transformation

**Implementation Phase**
- Source Code
  - Model-to-Code-Transformation
  - Source-to-Source-Translation (Code Transformation)

**Build Phase**
- Cross-Compilation / JIT-Compilation
- Compilation
- Byte Code
  - AOT- / JIT-Compilation
  - Interpretation

**Execution Phase**
- Machine Code
  - Execution
  - Interpretation

**Executable Program**
- Mobile Device

**Diagram Description**
- The diagram illustrates the phases and transformations involved in cross-platform development.
- It starts with the design phase, moving through implementation, build, and finally to execution.
- Key steps include transforming models into code, compiling and executing the code on a mobile device.
- The process emphasizes automation and optimization throughout the development cycle.
Translation Approach

Design Phase

Implementation Phase

Build Phase

Execution Phase

Model-to-Model-Transformation

Source Code

Model-to-Code-Transformation

Cross-Compilation / JIT-Compilation

Source-to-Source-Translation (Code Transformation)

AOT- / JIT-Compilation

Byte Code

Machine Code

Execution

Interpretation

GUI

Logic

Executable Program

Mobile Device
Interpretation Approach

Design Phase  Implementation Phase  Build Phase  Execution Phase

Model-to-Model-Transformation

Source Code

Model-to-Code-Transformation

Byte Code

Translation Approach

Source-to-Source-Translation (Code Transformation)

Cross-Compilation / JIT-Compilation

AOT- / JIT-Compilation

Machine Code

Execution

Executable Program

Mobile Device

GUI

Logic

GUI

Logic

Interpretation Approach

Dr. Thomas Springer

Application Development - 9. Cross-Platform Development
Web-based Approach
Hybrid Approach

Model-to-Model-Transformation

Model-to-Code-Transformation

Source Code

Translation Approach

Source-to-Source-Translation (Code Transformation)

Cross-Compilation / JIT-Compilation

AOT- / JIT-Compilation

Machine Code

Execution

Execution Phase

Build Phase

Implementation Phase

Design Phase

Model

GUI

Logic

Byte Code

GUI

Logic

Web-based Approach

Interpretation Approach
Model-driven Approach
Many tools and approaches
Example EMODE Model-Driven Approach

- Step-wise design and refinement
- Models for UI and application logic
- Integrated tool environment to consistently develop UI and application logic
• Qt C++ as source
• Virtual GUI toolkit
• Cross-platform libraries allow access to many device specific features (camera, contacts, Maps, GPS, Sensors, etc.)
• Mobile APIs /Android, iOS, Blackberry 10): access to hardware and mobile features (Sensors, Positioning, Bluetooth, NFC)
• Qt Quick allows declarative description of touch-based UIs
  • Interpreted in special runtime engine
- Java as common code base
- Parameterized code to customize code for different platforms
- Source-to-Source Translation to various programming languages
- Transformed code can be further edited
- Compatibility libraries per platform to bridge API-gap
- No IDE, but translation scripts, compatibility lib and cross-compiler

webMethods Mobile Designer Translation Approach

<table>
<thead>
<tr>
<th>Implementation-phase</th>
<th>Build phase</th>
<th>Execution phase</th>
</tr>
</thead>
<tbody>
<tr>
<td>JavaME-Sourcecode</td>
<td>Compilation</td>
<td>Interpretation</td>
</tr>
<tr>
<td>Java-Sourcecode</td>
<td>Compilation</td>
<td>Interpretation</td>
</tr>
<tr>
<td>Java-Sourcecode (Android)</td>
<td>Cross-Compilation</td>
<td>Interpretation</td>
</tr>
<tr>
<td>C++-Sourcecode</td>
<td>Compilation</td>
<td>Interpretation</td>
</tr>
<tr>
<td>C#-Sourcecode</td>
<td>Compilation</td>
<td>Interpretation</td>
</tr>
<tr>
<td>ActionScript</td>
<td>Compilation</td>
<td>Interpretation</td>
</tr>
<tr>
<td>HTML5, JavaScript</td>
<td>Compilation</td>
<td>Interpretation</td>
</tr>
</tbody>
</table>

- Code-Transformation, Cross-platform libraries
- Realisation in webMethods Mobile Suite
- Integration with Platform-SDKs

- Approach using \( m:1:n \) language translation
- XML as common language for byte code based on Java Bytecode
- Transformation based on XML Stylesheets
- Compatibility libraries used to port apps (even from Android to iOS)
- Specific AIR runtime engine + libraries
- GUI declaratively described in MXML (special extensions to support touch-based interactions)
- GUI description compiled to ActionScript code
Both hybrid and native HTML5 applications (both online and offline)
JavaScript and Ruby as common code base in one project
Cross-platform libraries
Application bundled with lightweight web server + Rhodes Ruby Framework + Ruby VM
Phone Gap
Hybrid Approach

- **Apache Cordova (formerly PhoneGap)**
  - Cross-platform tool to create mobile apps based on HTML, CSS and Javascript
  - Combination of Web-based and Translation approach
  - Supported platforms: iOS, Android, Windows Phone 7 and 8, BlackBerry OS and bada

- **Approach:**
  - web content wrapped in PhoneGap App
  - Native code to create native browser UI element (UIWebView (iOS) or WebView (Android))
  - and present locally stored web content
    - Runnable as App
    - Deployment via App Stores possible
Phone Gap Approach

- Web-technologies to create UI + logic
- Cross-platform PhoneGap Javascript API to access device specific features
  - Wrapped to native code
- Often used in combination with Mobile Web Application Frameworks like jQuery Mobile or Sencha Touch
- Native UI elements not supported
- **Start**: Cross-platform web project
- **Compilation with platform-specific IDEs**
  - Web service „PhoneGap build“ for creating Apps without native IDEs -> build.phonegap.com
• Cross-platform API provides common interface to access device specific features
  ➢ mapped to device OS by Javascript-to-native bridge

<table>
<thead>
<tr>
<th>Feature</th>
<th>iPhone / iPhone 3G</th>
<th>iPhone 3G and newer</th>
<th>Android</th>
<th>Blackberry OS 6.0+</th>
<th>Blackberry 10</th>
<th>Windows Phone 8</th>
<th>Ubuntu</th>
<th>Firefox OS</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerometer</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Camera</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Compass</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Contacts</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>File</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Geolocation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Media</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>X</td>
</tr>
<tr>
<td>Network</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Notification (Alert)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Notification (Sound)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Notification (Vibration)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Storage</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

✓ - supported feature
X - unsupported feature due to hardware or software restrictions

http://phonegap.com/about/feature
Goal: "Titanium apps should do more than just run on iOS and Android. When running on an iOS device, your app should feel like an iOS app. Your Android app should feel like an Android app."

"write once, adapt everywhere"
- Branching to use platform specific features

Mapping of Javascript code to precompiled native components accessible via JavaScript APIs
• Development based on C# for iOS, Android and Windows Phone
• Builds on Mono (open-source version of the .NET Framework)
• iOS
  • Xamarin’s Ahead-of-Time (AOT) Compiler compiles Xamarin.iOS applications directly to native ARM assembly code
  • MonoTouch runtime (memory allocation, garbage collection, underlying platform interop, etc.)
• Android
  • Xamarin’s compiler compiles down to Intermediate Language (IL), which is then Just-in-Time (JIT) compiled to native assembly when the application launches
  • IL code bundled with Mono for Android runtime which runs in parallel to Android RT/Dalvik VM
Xamarin Development

- App structure according to MVC
  - Native UI development in C# based on Xamarin APIs (MonoTouch.UIKit APIs, Android.Views)
  - Cross-platform functionality for business logic and data layer
    - Reusable Code separated into a Core Library
<table>
<thead>
<tr>
<th>Tool</th>
<th>Supported Device Platforms</th>
<th>Programming Language</th>
<th>Approach</th>
<th>App Execution Env.</th>
<th>Generic App parts</th>
<th>Native GUI-Widgets</th>
<th>Supported Platform-Features</th>
</tr>
</thead>
<tbody>
<tr>
<td>webMethods Mobile Designer</td>
<td>Android, Antix Game Player, bada, BlackBerry, BREW, Flash, iOS (iPhone, iPad), JavaME, Nintendo DS / DSI / DSIware, Sony PSP / PSP Minis, Symbian, webOS, Windows Mobile, Windows Phone 7, Browser (HTML5)</td>
<td>Java (JavaME)</td>
<td>Source-to-Source-Translation to C++, Java, C#, ActionScript, JavaScript; Platform-SDK for Cross-Compilation</td>
<td>native</td>
<td>Logic, GUI</td>
<td>no</td>
<td>★★★★★</td>
</tr>
<tr>
<td>Qt</td>
<td>Symbian, Maemo, MeeGo, Windows, Linux, Mac OS</td>
<td>C/C++</td>
<td>Cross-Compilation</td>
<td>native</td>
<td>Logic, GUI</td>
<td>Yes</td>
<td>★★★★★</td>
</tr>
<tr>
<td>XMLVM</td>
<td>iOS, Android, .NET-Plattform, Java-Plattform, further languages: JavaScript, Python, C++, Objective-C</td>
<td>Java, Ruby, Programming Language of .NET-Plattform</td>
<td>Bytecode-Compilation</td>
<td>native</td>
<td>Logic</td>
<td>Yes</td>
<td>★★★★★</td>
</tr>
<tr>
<td>Adobe AIR</td>
<td>iOS (iPhone, iPad), Android, Blackberry Tablet OS, Desktop-PCS, TV</td>
<td>ActionScript, MXML</td>
<td>Bytecode-Interpretation / AOT-Compilation (iOS)</td>
<td>Engine / native</td>
<td>Logic, GUI</td>
<td>partly</td>
<td>★★★★★</td>
</tr>
<tr>
<td>Kony</td>
<td>Android, iOS, J2ME, BlackBerry, Symbian, Windows Mobile, Windows Phone 7, webOS, BREW; Tablets: iOS, Android, BlackBerry, webOS;</td>
<td>Lua</td>
<td>Bytecode-Interpretation / Source-to-Source-Translation, Cross-Compilation (iOS)</td>
<td>Engine / native</td>
<td>Logic, GUI</td>
<td>yes</td>
<td>★★★★★</td>
</tr>
<tr>
<td>Rhodes</td>
<td>Android, BlackBerry, iOS (iPhone, iPad), Windows Mobile, Windows Phone 7</td>
<td>Ruby, HTML, CSS, JavaScript</td>
<td>Interpretation of scripting languages Bytecode-Interpretation</td>
<td>Engine, hybrid</td>
<td>Logic, GUI</td>
<td>partly</td>
<td>★★★★★</td>
</tr>
<tr>
<td>Titanium Mobile</td>
<td>Android, iOS (iPhone, iPad), BlackBerry, Titanium Mobile Web SDK: Browser</td>
<td>JavaScript</td>
<td>Interpretation approach</td>
<td>Engine</td>
<td>Logic, GUI</td>
<td>yes</td>
<td>★★★★★</td>
</tr>
<tr>
<td>Sencha Touch</td>
<td>iOS (iPhone, iPad), Android, BlackBerry (incl. PlayBook)</td>
<td>JavaScript, (HTML, CSS)</td>
<td>JavaScript-Application-Framework</td>
<td>Web</td>
<td>Logic, GUI</td>
<td>No</td>
<td>★★★★★</td>
</tr>
<tr>
<td>Google Web Toolkit</td>
<td>Any device with Web browser</td>
<td>Java</td>
<td>Source-to-Source-Translation to HTML, CSS, JavaScript</td>
<td>Web</td>
<td>Logic, GUI</td>
<td>No</td>
<td>★★★★★</td>
</tr>
<tr>
<td>PhoneGap</td>
<td>iOS (iPhone, iPad), Android (Smartphones, Tablets), webOS, Symbian, BlackBerry, Windows Phone, bada</td>
<td>Native Code, HTML, CSS, JavaScript</td>
<td>Hybride Approach</td>
<td>hybride</td>
<td>Logic, GUI</td>
<td>no</td>
<td>★★★★★</td>
</tr>
</tbody>
</table>
Strong fragmentation of market for mobile device platforms
Cross-Platform frameworks try to bridge the gap
Large number of tools following manifold approaches
  • Methodology
  • Target platforms
  • Native features
  • App anatomy
  • User interactions
No tool fulfills all requirements
In sum: instead of „Write once, run everywhere“ it’s more „Write once, run many“

→ Possible strategy for projects
  1. **Web App** to support large set of platforms
  2. **Native Apps** in addition for most important platforms
- Dirk Hering: Analyse von Methoden und Werkzeugumgebungen zur plattformunabhängigen Entwicklung mobiler Applikationen, Diplomarbeit, TU Dresden, 2010
- Calvary, Gaëlle; Coutaz, Joëlle; Thevenin, David; Limbourg, Quentin; Bouillon, Laurent; Vanderdonckt, Jean: A Unifying Reference Framework for multi-target user interfaces. In: Interacting with Computers 15 (2003), Nr. 3, S. 289–308
- Adobe AIR: http://www.adobe.com/products/air
- Bedrock: http://www.metismo.com
- Kony Platform: http://www.kony.com/platform
- PhoneGap: http://phonegap.com/
- Qt: http://qt.nokia.com/products
- Rhodes: http://rhomoile.com
- Sencha Touch: http://www.sencha.com/products-touch
- Titanium Developer: http://www.appcelerator.com
- XMLVM: http://xmlvm.org