Die Entwicklung von plattformübergreifenden Anwendungen auf mobilen Systemen wird in sechs verschiedene Ansätze kategorisiert.

Kompilierung (Compilation)

Pros

Cons

Komponentenbasiert (Component-Based)

Die Komponente besteht aus einem Paket oder einem Modul, dessen Funktionen und Daten untereinander in Relation stehen. Jede Komponente besitzt eine Schnittstelle, dass die Servicedienste spezifiziert, welche von anderen Komponenten genutzt werden können. Die Kommunikation findet ausschließlich über die Schnittstellen statt, so dass eine Komponente keinerlei Informationen über den Aufbau einer anderen benötigt.

Ein beispielhafter Lösungsansatz ist der Component-Based Mobile Web Application of Cross-Plattform. Dieser komponentenbasierte Ansatz versucht die Entwicklung mobiler Web-Apps dahingehend zu vereinfachen, dass durch das Konzept von Softwarekomponenten, die Kernfunktionalitäten modular aufgeteilt werden. Diese Module beinhalten Speichermanagement, Netzwerkkommunikation, Grafik, Dateisystem und die Systemdienstkomponenten. Dadurch erhalten die Komponenten eine Wiederverwertbarkeit und vereinfacht die Migration auf andere Plattformen. Jede Plattform kann dieselben Schnittstellen nutzen, benötigt jedoch eine eigene innere Implementierung für die Unterstützung.

Interpretierung (Interpretation)

Bei der Interpretierung übersetzt ein Interpreter (Dolmetscher) den Quellcode in ausführbare Anweisungen. Dies geschieht in Echtzeit mit Hilfe einer dedizierten Maschine. Hierbei existieren drei Unteransätze:

Virtuelle Maschine (VM)

Webbasiert (Web-based)

Laufzeit Interpretation (Runtime Interpretation)

Die bekannteste virtuelle Maschine ist die Java Virtual Machine (JVM). Diese verfügt über eine eigene komplette Hardwarearchitektur wie einer CPU, Stack, Register und ein korrespondierendes Befehlssystem. Die Grundidee hierbei ist es die mobile App mit einer plattformübergreifenden Sprache zu entwickeln, die auf der dedizierten, virtuellen Maschine läuft und auf entsprechenden Plattformen installiert ist.

Webbasierte Tools verwenden Technologien wie HTML(5), Javascript und CSS, die auf verschiedenen Plattformen ausführbar sind. Der Zugriff auf Hardwarekomponenten wie Kamera und Sensoren erfolgt durch Wrapper. Wrapper sind Adapter oder Schnittstellen, um auf die nativen APIs zugreifen zu können.

Bsp.: Phonegap

Die Laufzeit ist eine Ausführungsumgebung und eine Schicht, welche die mobile App auf der nativen Plattform lauffähig macht. Bei diesem Ansatz wird der Code in Bytecode umgewandelt und wird dann zur Laufzeit von einer virtuellen Maschine ausgeführt.

z.B. Titanium

Modellierung (Modeling)

Bei der Modellierung verwenden Entwickler abstrakte Modelle, um die Funktionen und / oder die Benutzeroberfläche der Anwendungen zu beschreiben. Diese Modelle werden für jede Zielplattform in entsprechenden Quellcode transformiert. Hierbei gibt es die Ansätze des Model-Based User Interface Development (MB-UID) und des Model-Driven Development (MDD).

\medskip

MB-UID wird genutzt, um die Benutzeroberfläche durch die formale Beschreibung von Aufgaben, Daten und Benutzern einer App automatisch zu generieren. Hierbei wird zwischen der Benutzeroberfläche und der App-Logik unterschieden.

Für die Generierung existieren zwei Strategien.

Eine Generation zur Laufzeit der App, die eine Websysteme adaptiert und basiert auf Anfrage- und Antwortprotokollen (request / response). Eingeschränkt wird dies durch die Voraussetzung einer dauerhaften Verbindung zu einem Server.

Die Generation während der Entwicklungszeit, also vor der Ausführung der Anwendung. Hier kann der Entwickler das generierte Interface überprüfen und zu jeder Plattform spezifische Funktionalitäten hinzufügen. Dabei kann die Funktion zur Verbindungsart festgelegt werden, also ob eine dauerhafte Verbindung bestehen soll oder zu einem selbst bestimmten Zeitpunkt synchronisiert werden soll.

Das Hauptkonzept von MDD ist die Generierung von plattformspezifischen Versionen, basierend auf dem plattformunabhängigen, abstrakten Modell. Das Modell wird durch Unified Model Language (UML) oder Domain-Specific Language (DSL) beschrieben.

Cloudbasiert (Cloud-Based)

In diesem Ansatz geschieht die Verarbeitung der Anwendung nicht lokal auf dem Gerät, sondern auf einem Cloudserver. Dabei werden einige Cloudeigenschaften verwendet, wie Flexibilität, Virtualisierung, Sicherheit und dynamisches Management. Die Clientanwendung ist dabei weitest möglich reduziert, da diese nur Basisprozesse benötigt. Dies wird Thin-Client genannt, da nur Ein- und Ausgabe verarbeitet werden müssen und dadurch als potenziell energieeffizient gelten.

Vereinigung (Merged)

Dieser Ansatz versucht die besten Eigenschaften verschiedener Ansätze zusammenzuführen, von den jeweiligen Vorteilen zu profitieren und Nachteile zu minimieren.

Ein unbenannter Lösungsansatz vereinigt den komponentenbasierten Ansatz mit dem Cross Compiler und einer darauf angepassten Universalsprache. Um die nativen Hardwarefunktionen wie Kamera und GPS, sowie native Softwareeigenschaften wie Buttons und andere Interaktionsfelder anzusprechen, wird eine Sammlung an Komponenten erstellt. Implementierungen dieser Komponenten können durch gemeinsame Schnittstellen für jede Zielplattform erfolgen. Dieses Framework soll dem Entwickler ermöglichen, Applikationen zu entwickeln, die auf nativen Code und der Universalsprache basieren. Diese Sprache wird der App als zusätzliche Kommunikationsschicht und Schnittstelle hinzugefügt, um die Komponenten und deren Methoden anzusprechen. Der Entwickler implementiert nur eine minimale Struktur der App auf nativer Basis, welche die Benutzerschnittstelle und Navigation beinhaltet. An welcher Stelle und auf welche Weise die Komponenten integriert werden, wird durch die Universalsprache definiert. Das Framework regelt die Codeintegrierung innerhalb des nativen Codes. Bei diesem Lösungsansatz ist es erforderlich die Benutzerschnittstelle für jede Plattform manuell zu definieren. Dabei liegt der funktionale Fokus auf allgemeingültigen Funktionen, ohne Berücksichtigung der plattformspezifischen.

Integrated Cross-Platform Mobile Development (ICPMD) ist eine weitere Lösung die auf dem Merge-Prinzip aufbaut und drei Verwendungsszenarios unterstützt. Diese Szenarios sind abhängig von dem gegebenen Input.

Der Entwickler hat…

…bereits ein bestehendes Projekt (z.B. Windows Phone) und möchte dies auf weitere Plattformen (z.B. iOS und Android) ausweiten.

…definierte Anforderungen und möchte daraus, auf bestimmte Zielplattformen, eine mobile App erzeugen.

…ein Projekt basierend auf dem abstrakten Modell und möchte dies aktualisieren und dann speichern oder daraus, auf bestimmte Zielplattformen, eine mobile App erzeugen.

Entwicklung mobiler Applikationen ohne den Schwerpunkt Spieleentwicklung

In der Abschlussarbeit „Plattformabhängige und –unabhängige Entwicklung mobiler Anwendungen am Beispiel von Geo-Wikipedia-App“ wird ebenfalls die plattformübergreifende Entwicklung analysiert, jedoch liegt hier nicht der Fokus auf Game-spezifischen Applikationen und Entwicklungswerkzeugen. Hier wird in Kapitel 2.2 auch auf die verschiedenen Herangehensweisen eingegangen und deren Resultate klassifiziert. Weiterhin wird eine Auswahl der bekannteren Cross-Plattform Entwicklungstools (PhoneGap, Xamarin, Appcelerator) analysiert. Diese Arbeit setzt den Schwerpunkt auf Frameworks und Engines zur Spieleentwicklung und geht von daher nicht weiter auf die genannten und vorangegangenen Entwicklungswerkzeuge ein.