

9

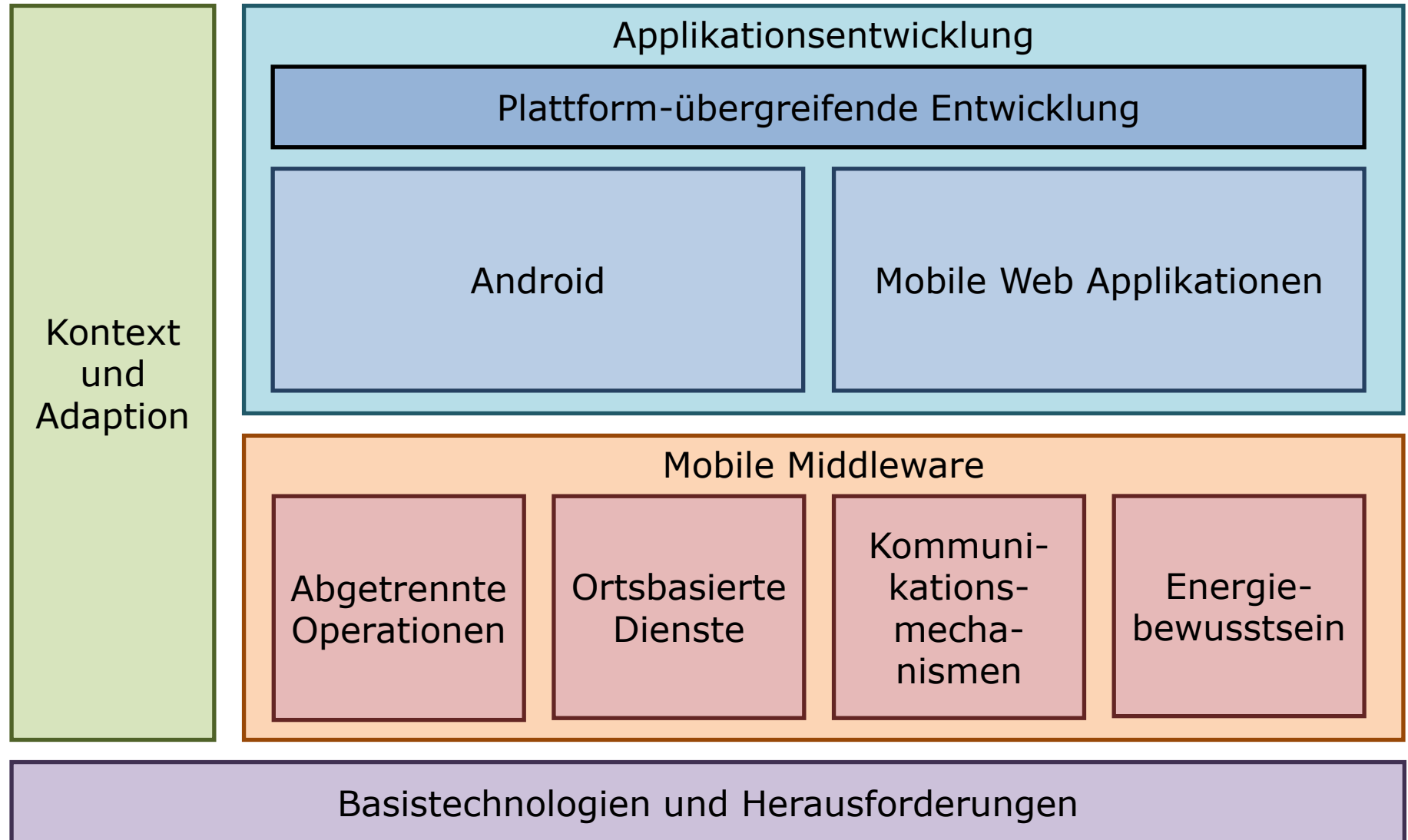
Web- und App-Programmierung

Plattform- übergreifende Entwicklung

mit Skriptmaterial von Dr.-Ing. T. Springer

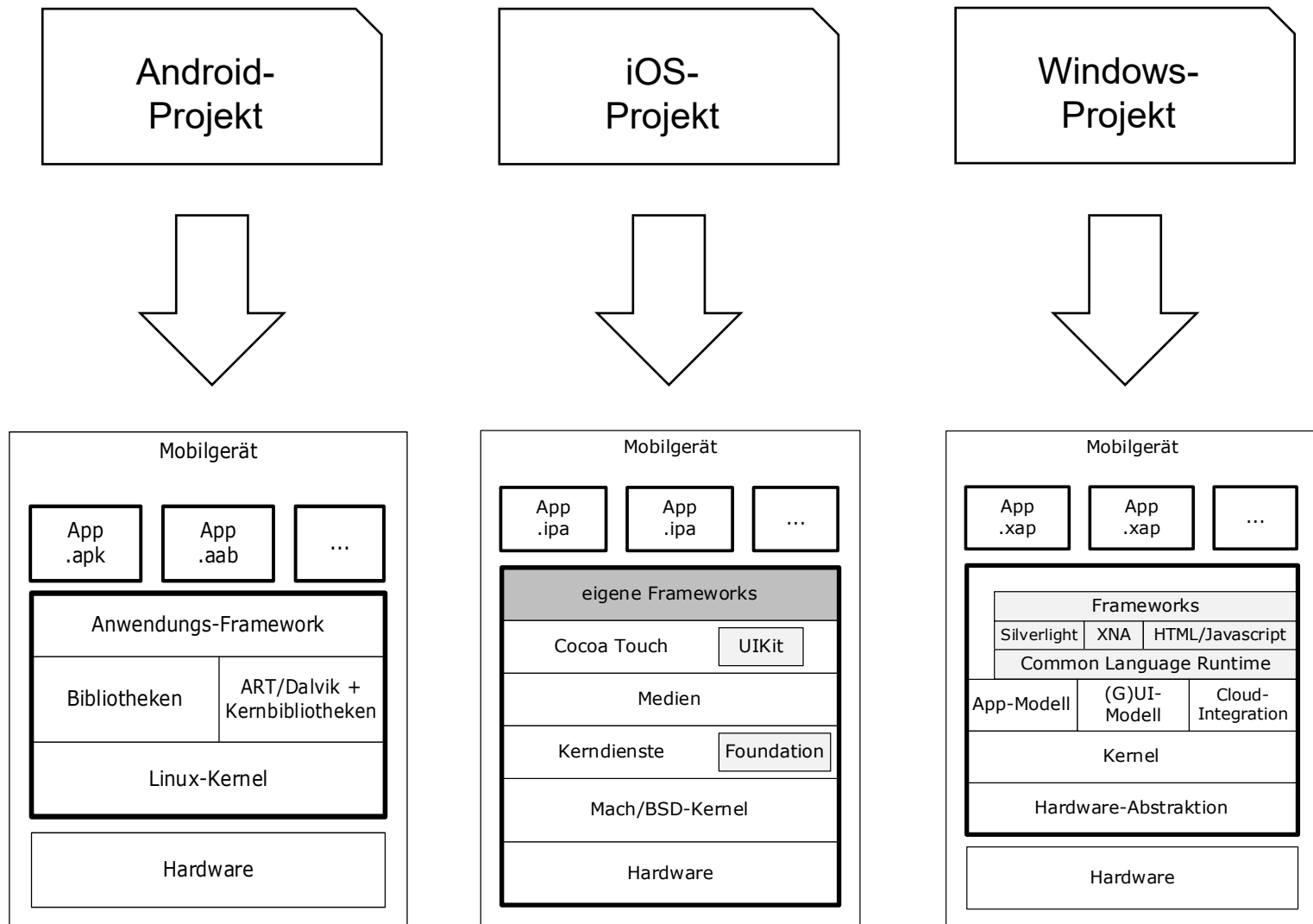
Prof. Dr.-Ing. Tenshi Hara
fragen@lern.es

Aufbau der Lehrveranstaltung



Motivation und Prinzipien

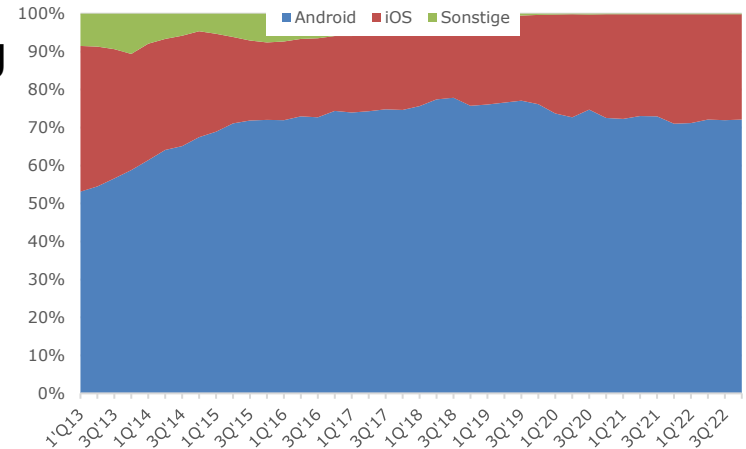
Warum plattformübergreifend entwickeln?



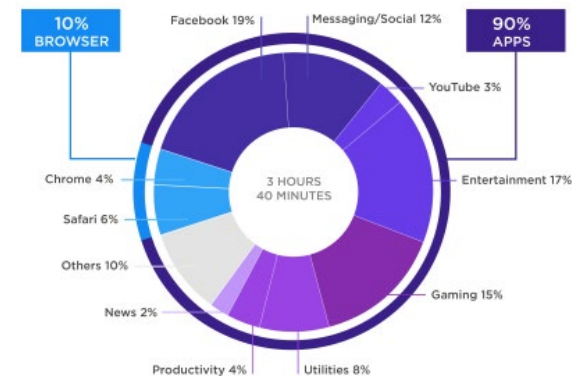
Warum plattformübergreifend entwickeln?

- hoher Entwicklungsaufwand
 - Know-how für multiple Plattformen nötig
 - schnelle Technologiefortschritte
 - hohe Plattformfragmentierung, insbesondere bei Android
 - Kunden zahlen nur Projekte, nicht für einzelne Zielplattformen
- hoher Wartungsaufwand
 - Verwalten multipler Codebasen
 - Vermeidung von Inkonsistenzen
- neue Formfaktoren verstärken Heterogenität
- kurze Markteinführungszeiten benötigt
→ Änderungen an Plattformen müssen zügig adressiert werden
- Unterstützung von Mobilität und Adaptivität

Marktanteil Android, iOS und Sonstige

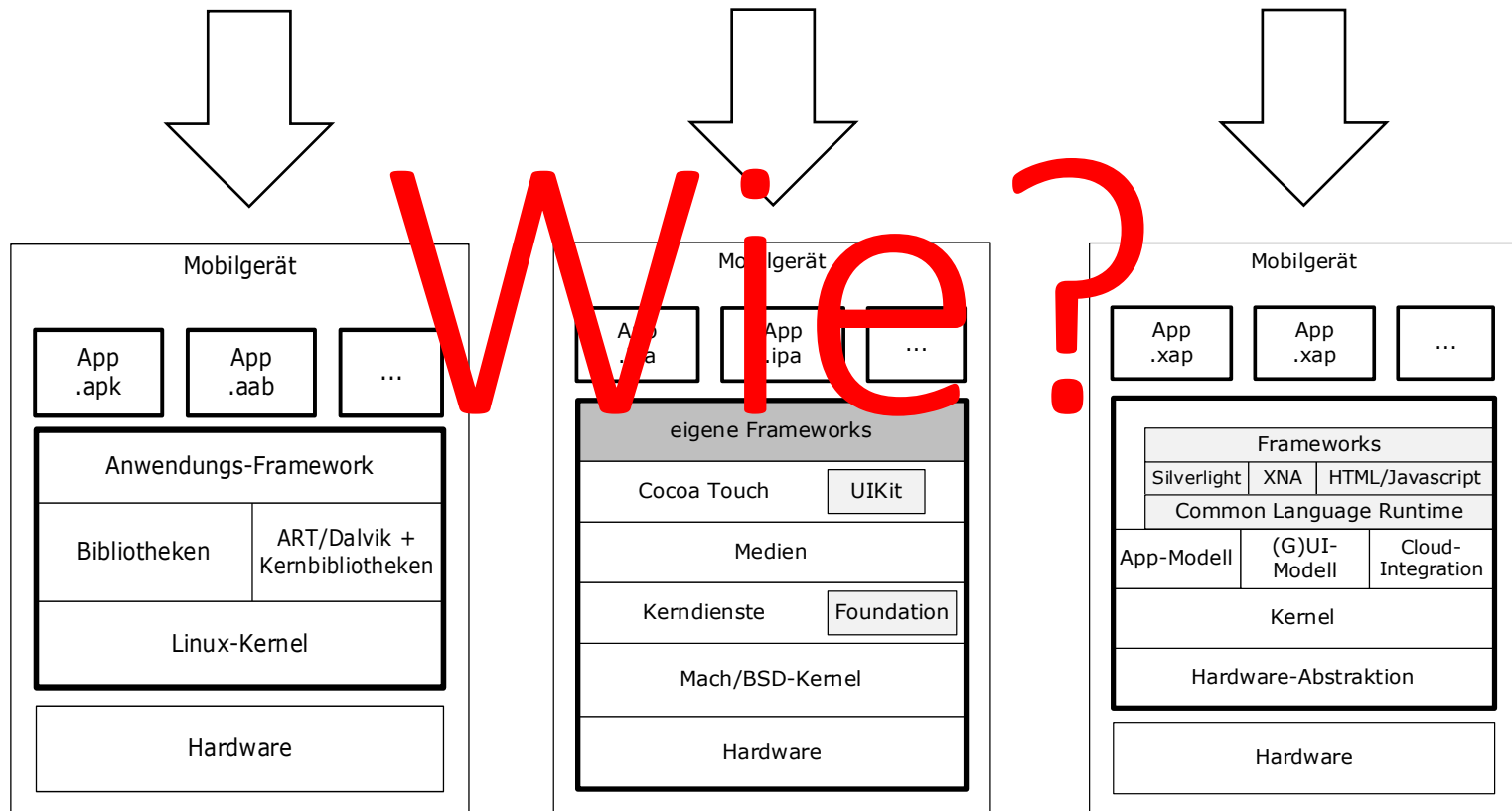


90% of Time on Mobile is Spent in Apps



Write once, run everywhere?

Plattform-übergreifendes Projekt



„Write once, run everywhere“-Prinzip – Herausforderungen

- heterogene Hardware
 - Formfaktoren
 - Ressourcen
 - Gerätefähigkeiten (Touch, physische Schaltflächen, Konnektivität, ...)
- heterogene Software-Plattformen
 - Ausführungsumgebung
 - Programmiersprachen
 - Plattform-APIs, Nutzerschnittstellenvorgaben
 - App-Anatomie und zugrunde liegendes Programmiermodell
 - Werkzeugketten
 - Bereitstellungs- und Verifikationsprozess
- Nutzererwartungen
 - natives Aussehen und Nutzungserfahrung
 - Design-Vorgaben

Geräteplattformen und nativer Code

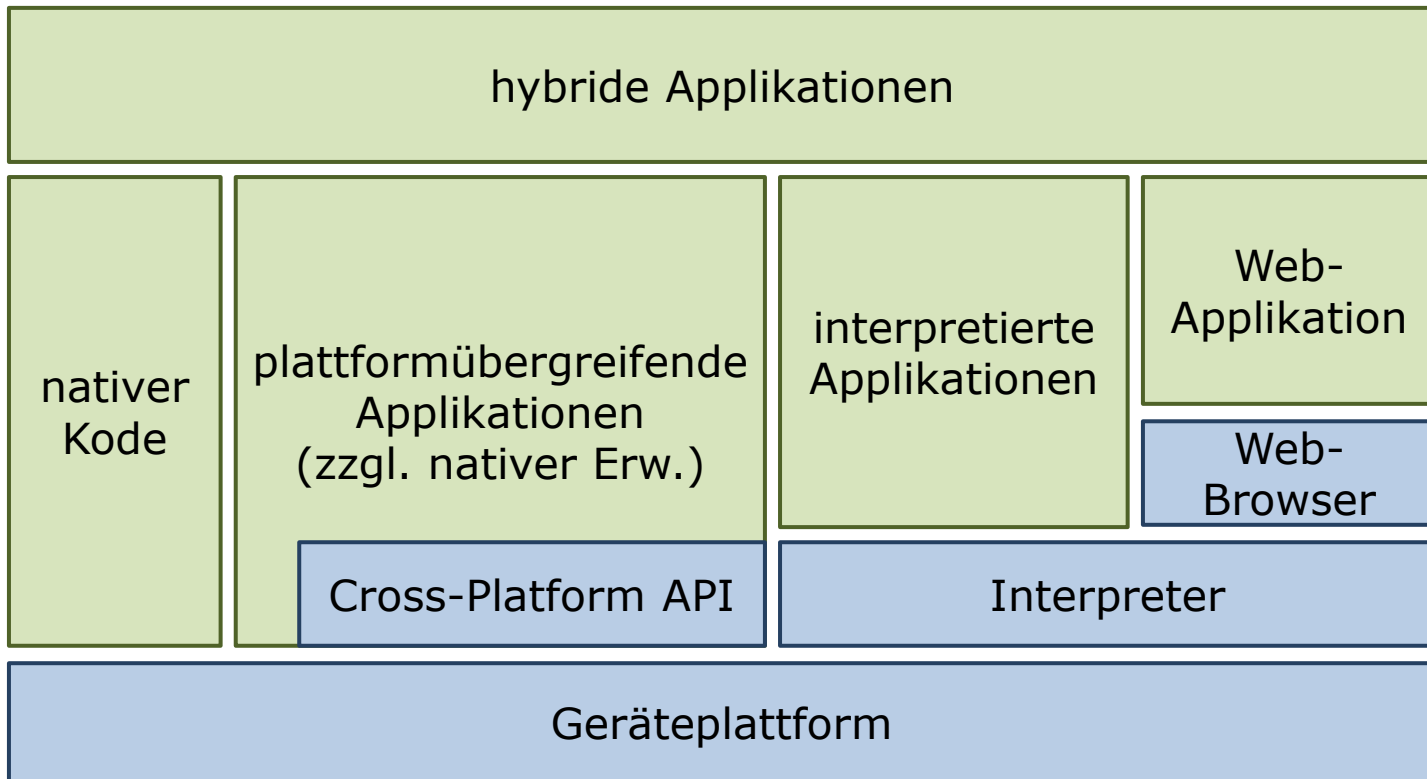
Eine **Geräteplattform** ist eine Kombination aus Geräte-Hardware, Betriebssystem, Laufzeitsystem, Bibliotheken und Frameworks, die eine standardisierte Ausführungsumgebung für Applikationen darstellt, die auf dieser Plattform ausgeführt werden. Sie hat bestimmte Eigenschaften:

- spezifisches Aussehen und Nutzungserlebnis ((G)UI-Vorgaben und Interaktionskonzepte)
- vom Laufzeitsystem definierte Applikationsanatomie und -lebenszeit
- Entwicklungswerkzeugkette (Programmiersprachen, Bibliotheken, APIs)
- spezifischer Applikationsbereitstellungs- und -veröffentlichungsprozess

Nativer Code ist Code, der direkt in der Standardausführungsumgebung der Geräteplattform ausgeführt werden kann. Er hat vollen Zugriff auf die Plattformbibliotheken und Frameworks, oft auch auf die Betriebssystem-APIs und Hardware-Eigenschaften.

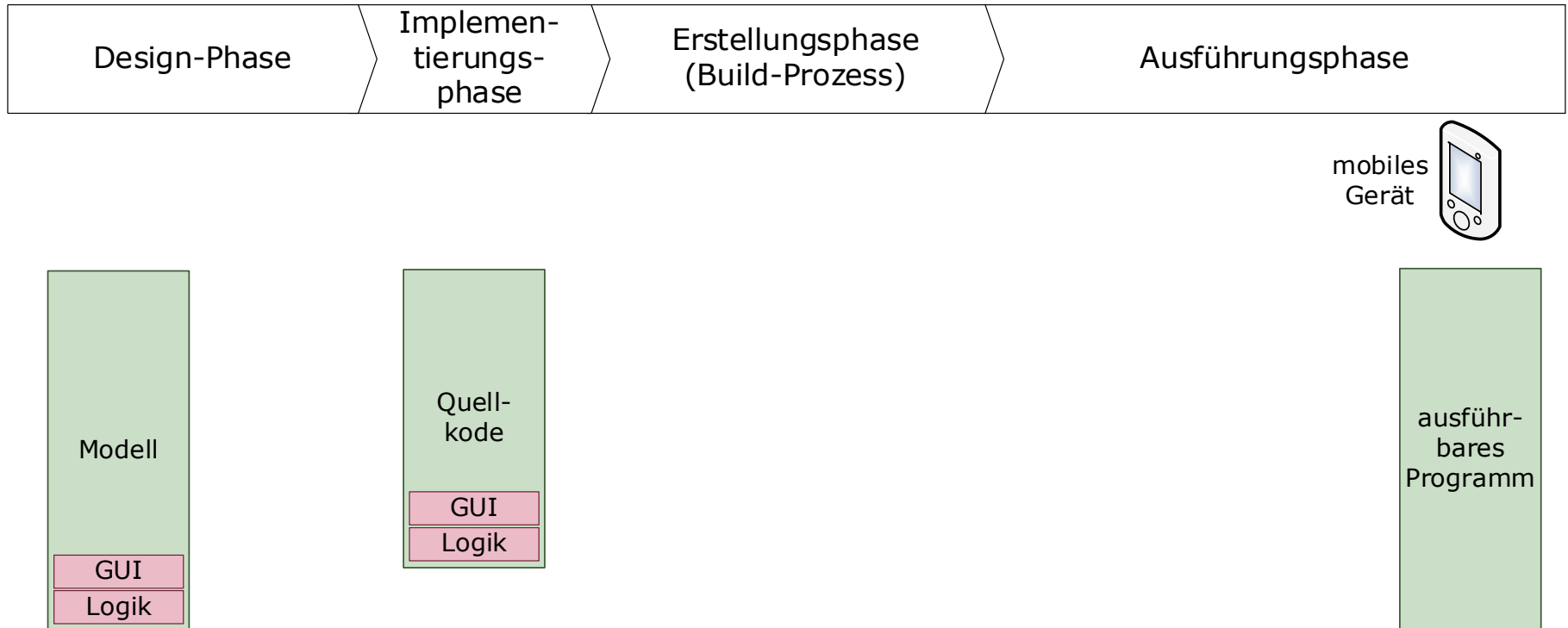
Typen mobiler Applikationen

Typen mobile Applikationen



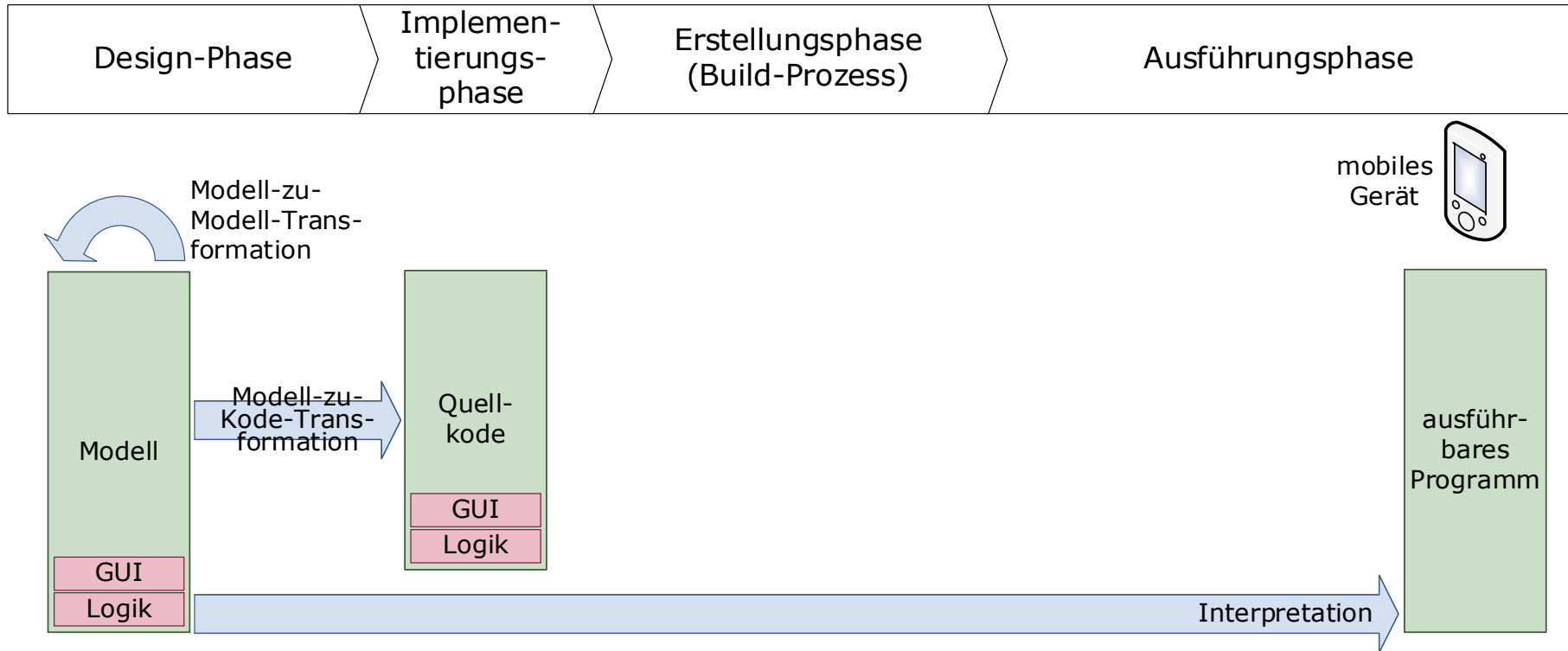
Ansätze der plattformübergreifenden Entwicklung

Womit beginnen? Modell oder Quellcode?



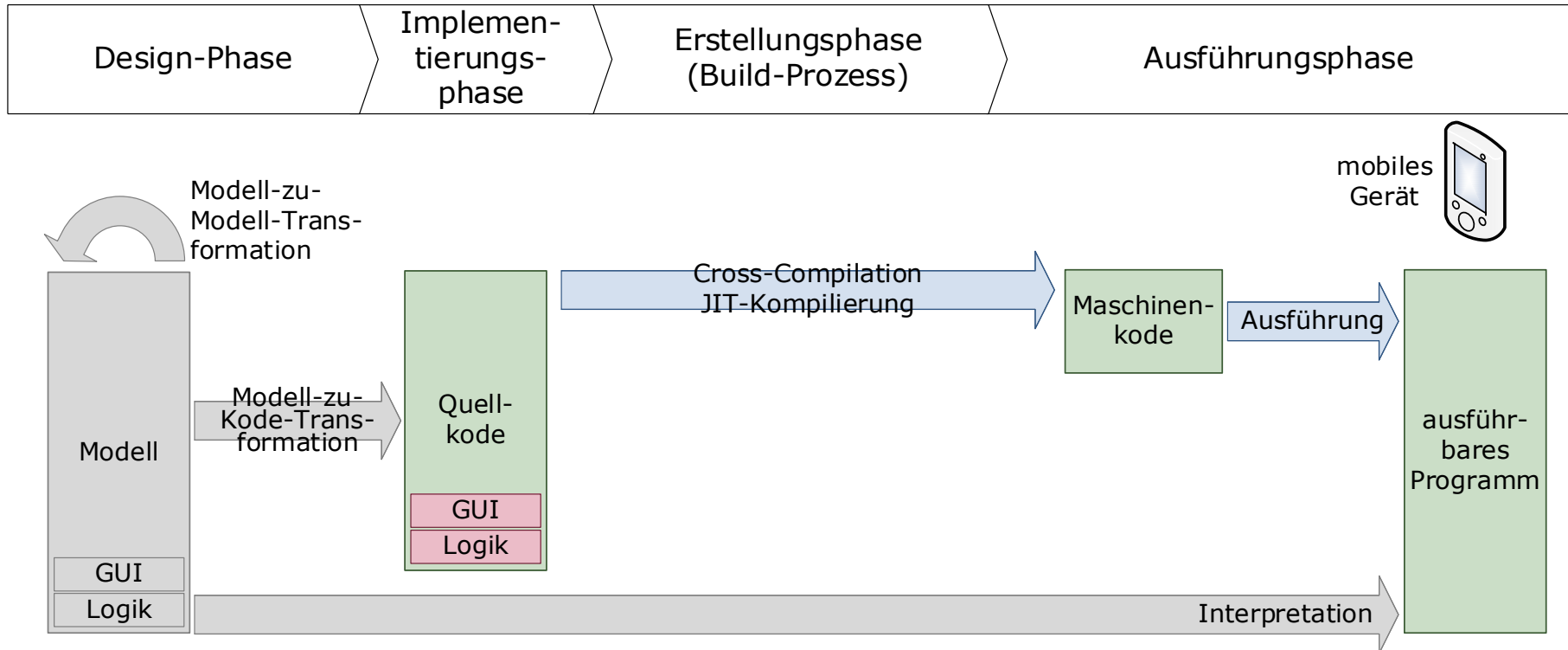
Ansätze der plattformübergreifenden Entwicklung

modellgetriebene Entwicklung



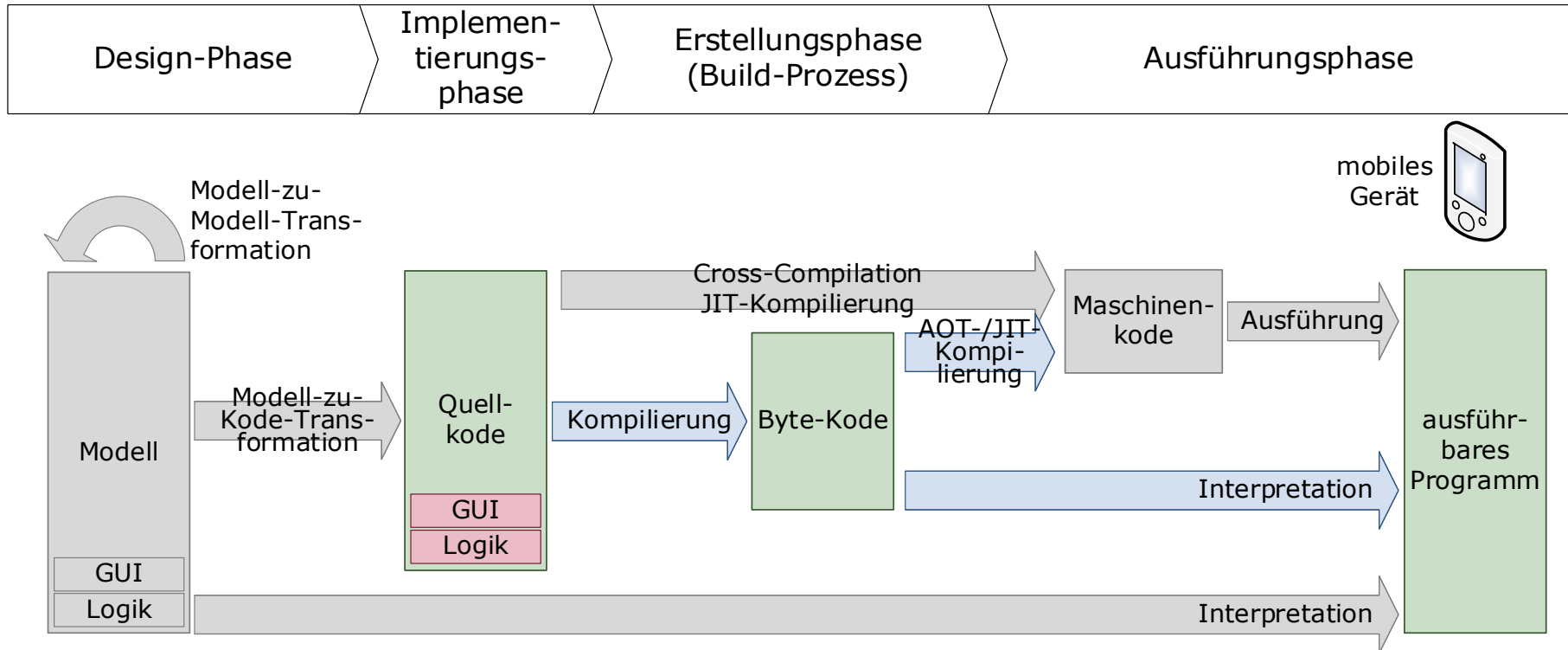
Ansätze der plattformübergreifenden Entwicklung

Cross-Compilation zu nativem Code



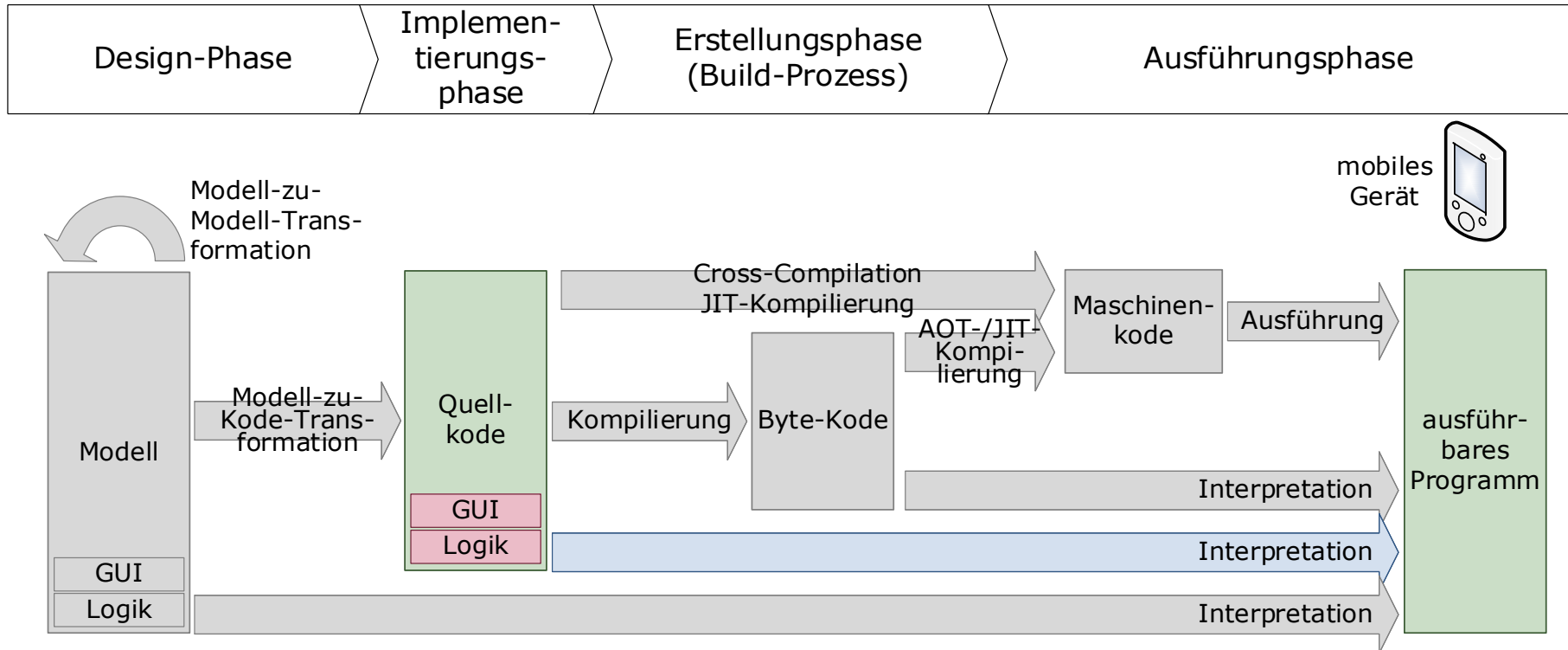
Ansätze der plattformübergreifenden Entwicklung

Unterstützung abstrakter Laufzeitumgebungen



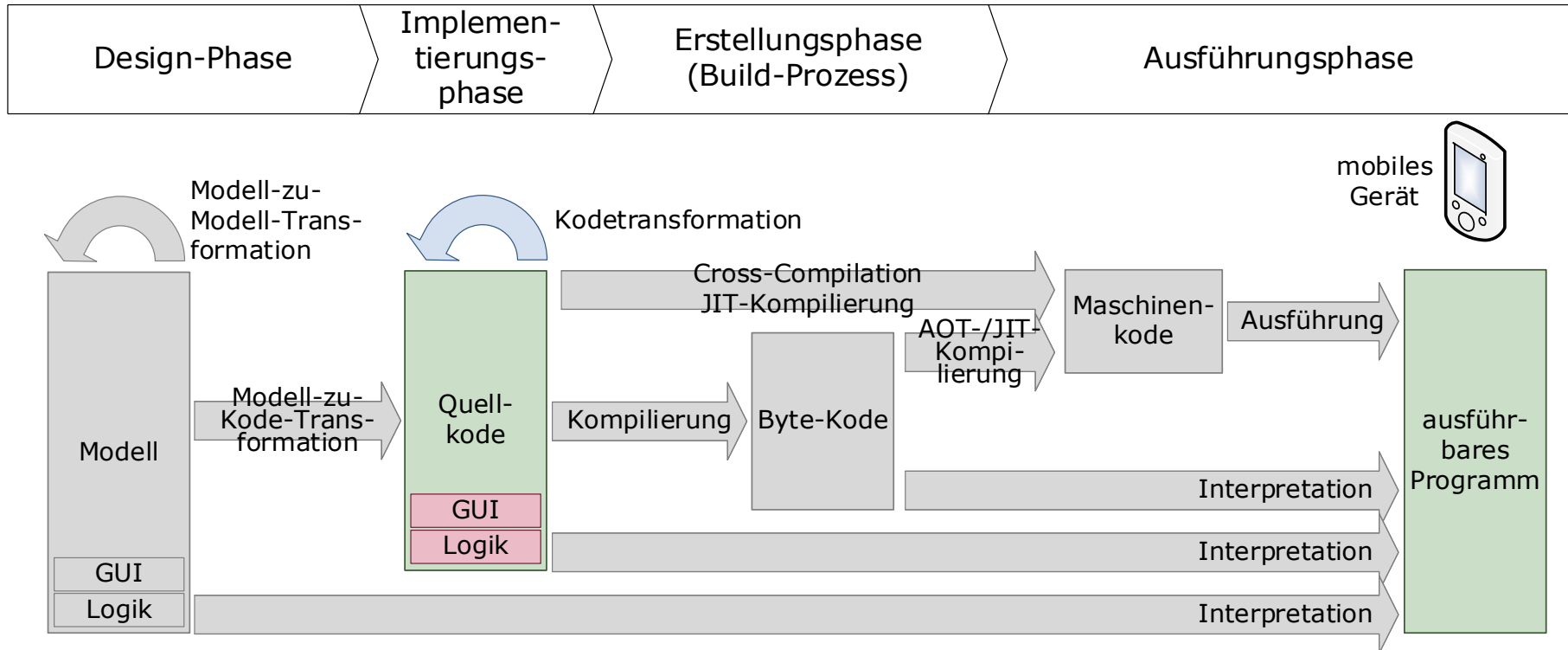
Ansätze der plattformübergreifenden Entwicklung

Script-Sprachen

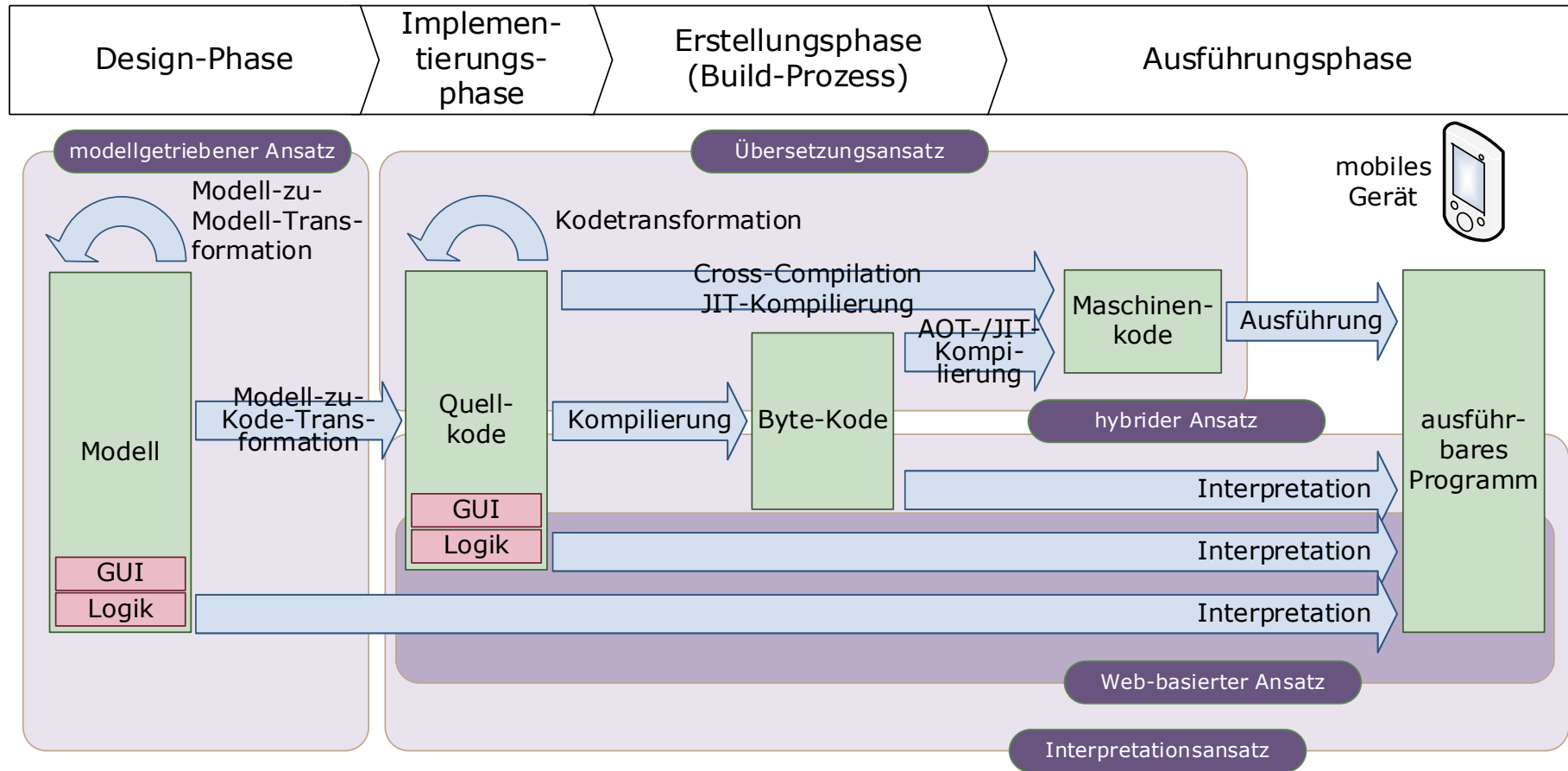


Ansätze der plattformübergreifenden Entwicklung

Übersetzung in andere Programmiersprachen



Ansätze der plattformübergreifenden Entwicklung

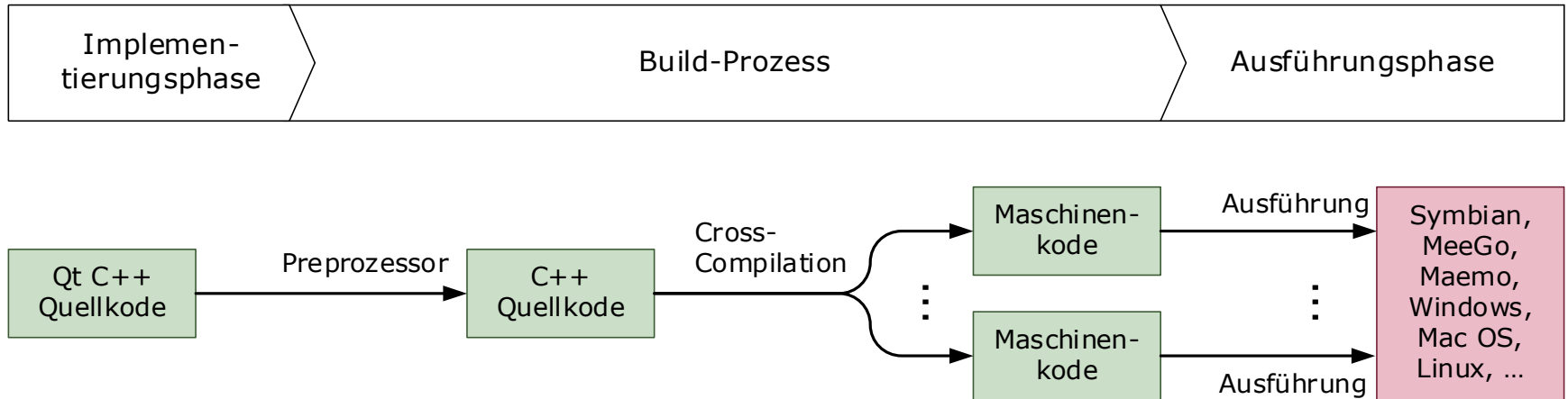


Werkzeug- und Ansatzdiversität

Werkzeuge & Ansätze



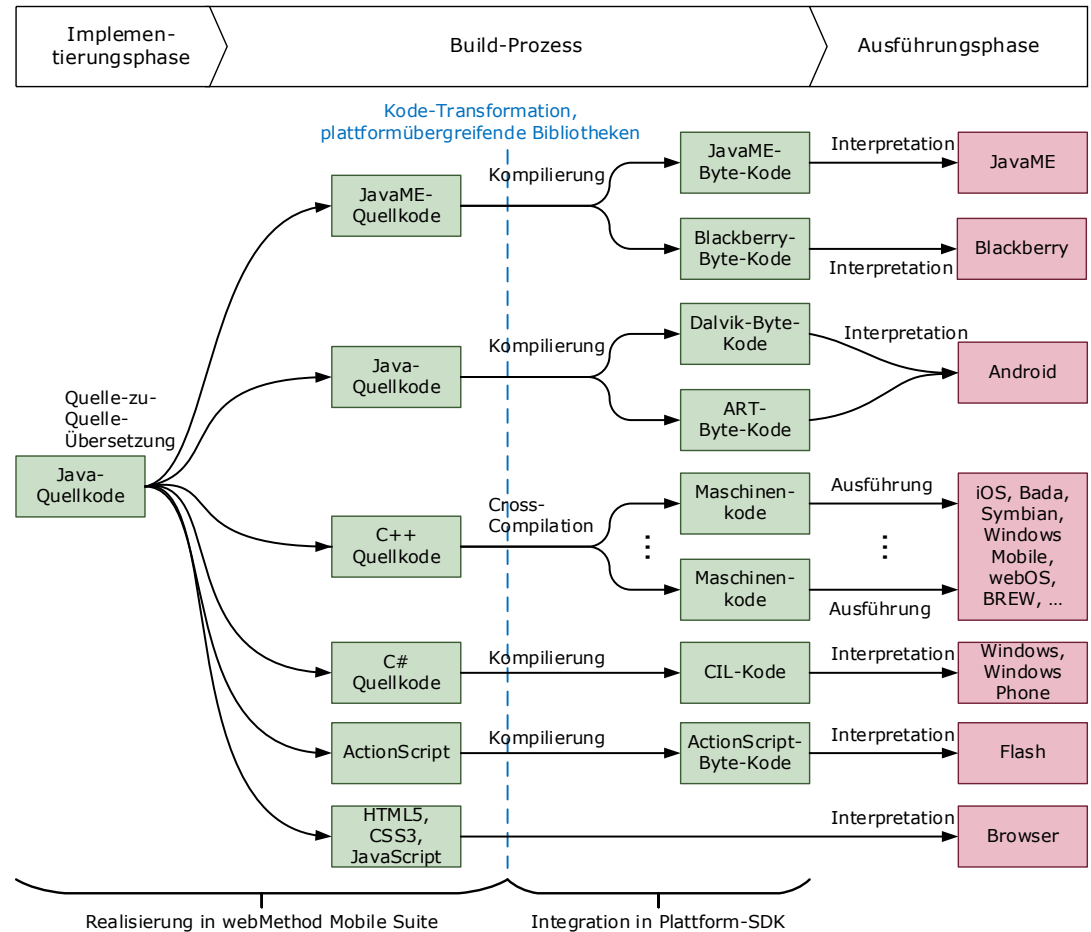
Beispiel: Qt – Übersetzungsansatz



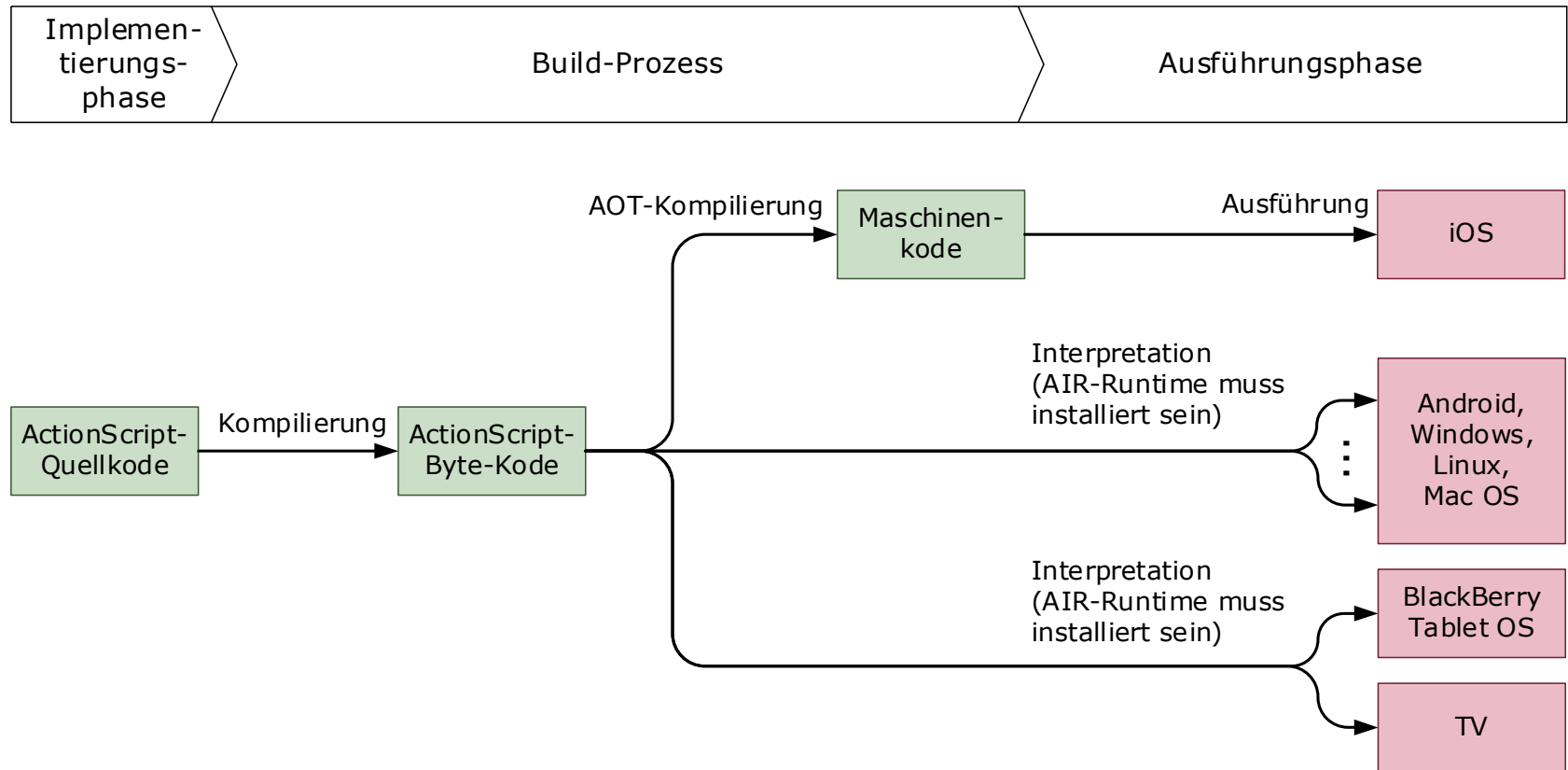
- Qt C++ als Quellen
- virtuelles GUI-Werkzeug
- plattformübergreifende Bibliotheken erlauben Zugriff auf (viele) gerätespezifische Eigenschaften (Kamera, Kontakte, GPS, Sensoren, ...)
- mobile APIs (Android, iOS, Blackberry 10): Zugriff auf Hardware und mobile Eigenschaften (Sensoren, Positionierung, Bluetooth, NFC, ...)
- via Qt Quick deklarative Beschreibung der Touch-basierten GUI
→ wird in spezieller Laufzeitumgebung interpretiert

Beispiel: webMethods Mobile Designer – Übersetzungsansatz

- Java als gemeinsame Kodebasis
- Quelle-zu-Quelle-Übersetzung zu diversen Programmiersprachen
- parametrisierter Code zur Anpassung des Codes an unterschiedliche Plattformen
- transformierter Code kann weiter bearbeitet werden
- Kompatibilitätsbibliotheken für jede Plattform schließen API-Lücken
- keine IDE, aber Übersetzungs-Skripte, Kompatibilitätsbibliotheken und Cross-Compiler



Beispiel: Adobe Air – Interpretationsansatz

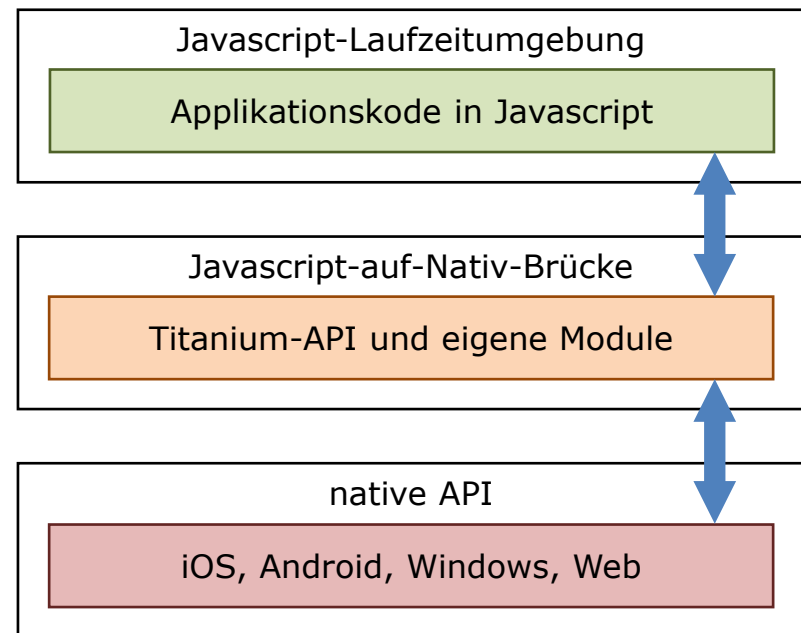


- spezifische AIR-Runtime und -Bibliotheken
- deklarative GUI-Beschreibung in MXML
(Erweiterungen für Touch-/Gesten-Steuerung vorhanden)
- GUI-Beschreibung wird zu ActionScript-Kode kompiliert

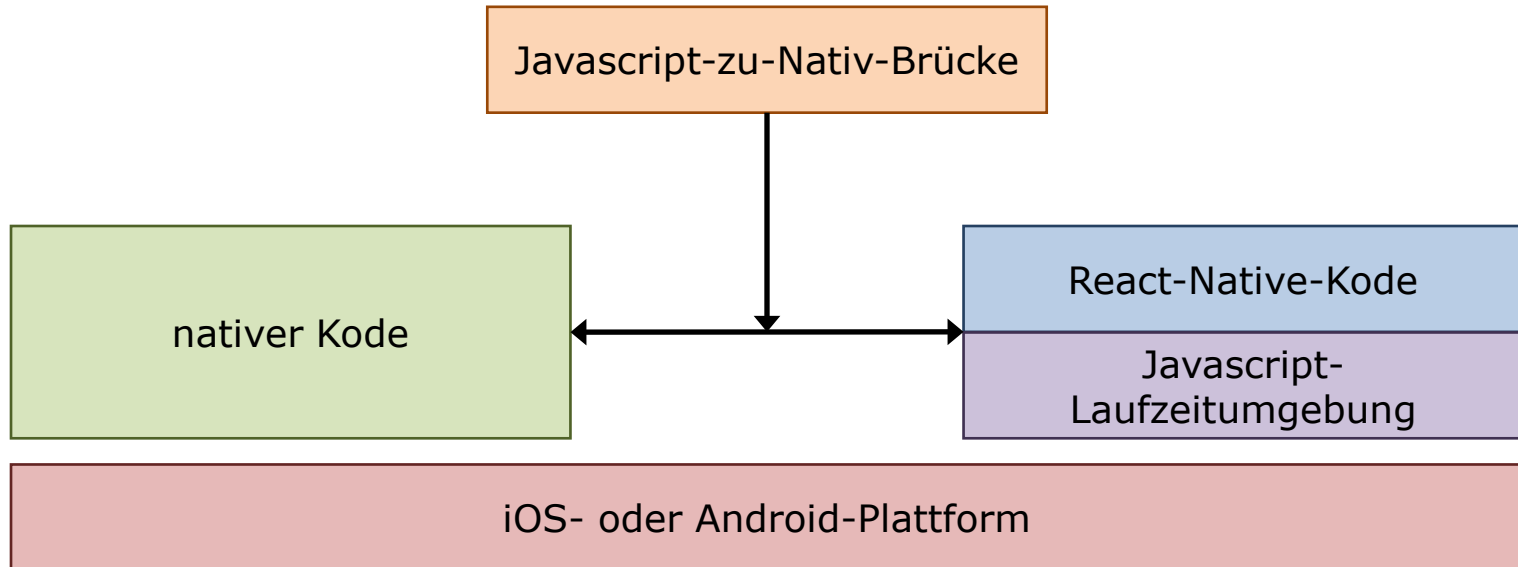


Beispiel: Appcelerator Titanium – Interpretationsansatz

- Javascript als Programmiersprache
- plattformübergreifende Bibliothek zur Übersetzung von Titanium-API-Aufrufen auf native API-Rufe
- Module enthalten Kernfunktionalität der API
→ eigene Module für Erweiterungen
- in jeder Applikation wird nativer Code mit Javascript-Laufzeitumgebung verknüpft
- Javascript-auf-Nativ-Brücke
- React Native nutzt ähnlichen Ansatz



Beispiel: React Native – Interpretationsansatz



- Programmiersprache: Javascript
- Javascript-zu-Nativ-Brücke bildet native APIs in Javascript ab

weitere VM-basierte Ansätze und Cross Compilation

- WebContainer
<https://blog.stackblitz.com/posts/introducing-webcontainers/>
- Web Components
<https://www.heise.de/blog/SPA-ohne-Framework-Web-Components-als-Ersatz-fuer-React-Co-4850959.html>
- Felgo als Erweiterung von Qt
<https://felgo.com>
<https://www.heise.de/developer/meldung/Cross-Plattform-Felgo-3-4-0-fuehrt-QML-Hot-Reload-ein-4686158.html>
- Flutter → nächste Folie



Beispiel: Flutter – Cross-Compilation-Ansatz

- von Google entwickelte Cross-Plattform-Lösung für Android, iOS, Linux, macOS, Windows und Web
- Entwicklung in Dart
- Deployment per
 - AOT-Compiler direkt für Zielplattform,
 - AOT-Compiler zu JavaScript für Web-Anwendungen oder
 - JIT-Compiler mit Ausführung in Dart-VM
- Widgets folgen den Designvorgaben der jeweiligen Zielplattform (Material, iOS, macOS, Modern UI)

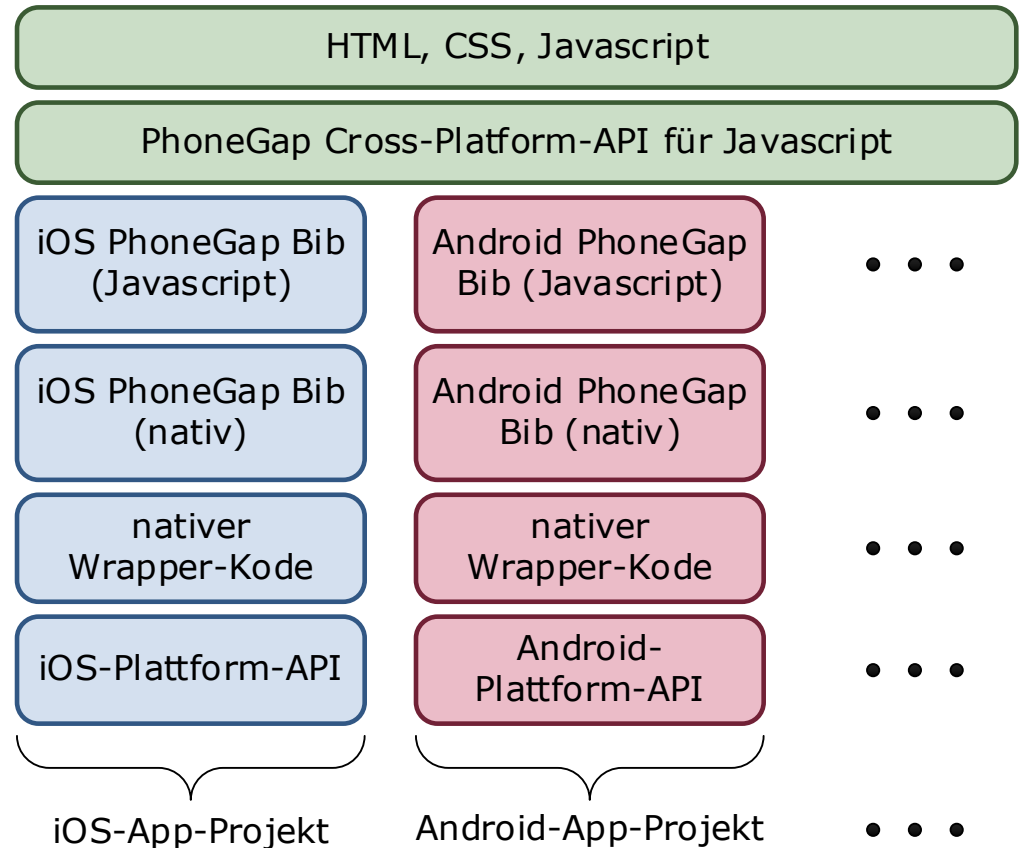
→ <https://flutter.dev>

Beispiel: Apache Cordova – hybrider Ansatz

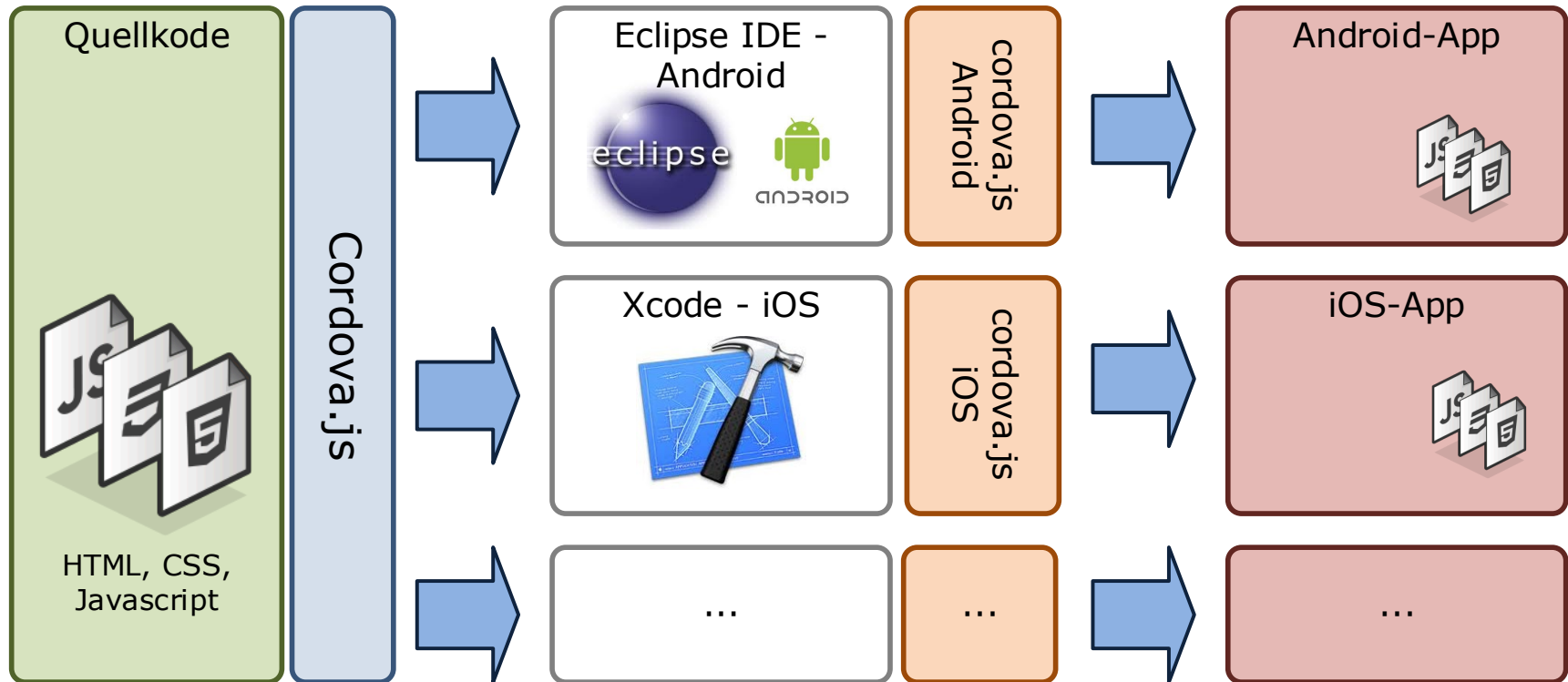
- Kodebasis von PhoneGap
- plattformübergreifendes Werkzeug zur Erstellung mobiler Apps auf Basis von HTML, CSS und Javascript
- Kombination der webbasierten und übersetzungsbasierten Ansätze
- unterstützte Plattformen: iOS, Android, Windows Phone 7 und 8, BlackBerry und Bada
- Web-Inhalte
- läuft als native App
- Bereitstellung über Stores möglich
- Ansatz
 - Web-Inhalt wird in PhoneGap-Applikation gekapselt
 - nativer Code zur Erstellung der GUI-Elemente des Browsers (UIWebView (iOS) oder WebView (Android))
 - Anzeigen lokal gespeicherter Inhalte

Beispiel: Apache Cordova – hybrider Ansatz

- GUI und Logik erzeugt durch Web-Technologien
- Zugriff auf gerätespezifische Eigenschaften durch plattformübergreifende PhoneGap-Javascript-API
- gebunden als nativer Code
- oft in Kombination mit Web-Frameworks verwendet, bspw. jQuery und Sencha Touch
- native GUI-Elemente werden nicht unterstützt
→ Nachahmung via CSS



Beispiel: Apache Cordova – hybrider Ansatz



- Start: plattformübergreifendes Web-Projekt
- Kompilierung mit plattformspezifischen IDEs
- Web-Dienst (PhoneGap build) zum Erstellen von Applikationen ohne native IDEs → build.phonegap.com

Beispiel: Apache Cordova – hybrider Ansatz

- plattformübergreifende API bietet gemeinsame Schnittstelle für Zugriff auf gerätespezifische Eigenschaften
- abgebildet auf Gerätebetriebssystem durch Javascript-auf-Nativ-Brücke
- generell (iOS und Android) unterstützte Eigenschaften
 - Beschleunigungssensor
 - Kamera
 - Kompass
 - Kontakte
 - Benachrichtigungen (Alarm, Ton Vibration)
 - diverse sonstige sensorspezifische Eigenschaften
 - Dateien
 - Lokation
 - Medien
 - Netzwerk
- des Weiteren (teilweise eingeschränkte) Unterstützung von Blackberry, Windows Phone

Beispiel: Capacitor – Hybrider Ansatz



- folgt der Philosophie von Apache Cordova und Adobe PhoneGap
- erzeugt universelle Apps basierend auf Web-Technologien
→ somit auch universelle Progressive Web Apps
- ermöglicht nativen API-Zugriff
(nativer Kern für jede Plattform: „Core Native Plugin“)
 - Kamerazugriff
 - Dateisystem
 - Geolokation
 - Beschleunigungssensoren
 - Benachrichtigungen
 - Netzwerkzugriff
 - haptische Interaktion
 - weitere, eigene Plugins möglich

→ <https://capacitorjs.com>

Beispiel: .NET MAUI – hybrider Ansatz (1/2)

- Entwicklung basieren auf C# für iOS, Android und Windows Phone
- baut auf .NET auf (Mono-Kodebasis funktioniert auch)
- unterstützt diverse MVVM-Frameworks (MVVM-Toolkit von Microsoft, ReactiveUI, Prism u.v.a.m.)
- Windows
 - AOT-Kompilierung wie bei „normalen“ .NET-Anwendungen
 - x86, x64, ARM und Modern als Target möglich
 - MAUI-Laufzeitumgebung (Speicherzuweisung, Garbage Collection, Integration mit Windows Core Components etc.)
 - spezielle Bibliotheken möglich (bspw. zur ASLR-Steuerung bei Multimedia- oder Gaming-Anwendungen)

Beispiel: .NET MAUI – hybrider Ansatz (2/2)

- iOS
 - AOT-Kompilierung von iOS-Applikationen direkt in ARM-Assembly
 - MAUI-Laufzeitumgebung (Speicherzuweisung, Garbage Collection, Interoperabilität mit zugrundeliegender Plattform etc.)
- Android
 - Kompilierung in **Intermediate Language** (IL), auf deren Basis beim Programmstart JIT-Kompilierung in native Zusammenstellung erfolgt
 - IL-Kode wird mit MAUI-Laufzeitumgebung für Android gebündelt, welche parallel zu ART/Dalvik läuft
- Windows
 - Kompilierung als native Binary für WPF und MSIX
 - ständige Änderungen durch Microsoft

<https://www.heise.de/news/NET-MAUI-Microsofts-neue-GUI-Bibliothek-fuer-Windows-Android-iOS-und-macOS-7121152.html>

<https://www.heise.de/blog/Windows-Forms-lebt-4987733.html>

Zusammenfassung und Aufgaben

Zusammenfassung

- starke Fragmentierung der Geräteplattformen
- plattformübergreifende Frameworks versuchen, Lücke zu überbrücken
- viele Werkzeuge mit unterschiedlichen Ansätzen
 - Methodik
 - Zielplattformen
 - native Eigenschaften
 - Applikationsanatomie
 - Nutzerinteraktionen
- kein Werkzeug erfüllt alle Anforderungen
 - „Write once, run everywhere“ geht nicht; eher „Write once, run many“

mögliche Projektstrategie

1. Web-App mit Unterstützung vieler Plattformen, dann
2. darauf aufbauende native Apps für die wichtigsten Zielplattformen

Aufgaben

- Diskutieren Sie mit Ihren Kommilitonen die unterschiedlichen Ansätze der Entwicklung plattformübergreifender Applikationen.
→ Was sind besonders wichtige Vor- und Nachteile?
- Weshalb ist „Write once, run many“ manchmal gegenüber nativen Applikationen zu bevorzugen? Wann nicht und wieso?
- Lesen Sie aufmerksam eins der drei Tutorials bei heise und versuchen Sie anschließend, das ausgewählte Tutorial nachzuvollziehen
 - Flutter: <https://www.heise.de/hintergrund/Flutter-Cross-Plattform-a-la-Google-4480141.html>
 - Kotlin Multiplatform Mobile:
<https://www.heise.de/hintergrund/Kotlin-Multiplatform-Mobile-Native-Apps-entwickeln-mit-Multiplattform-Technik-7155149.html>
 - Vue.js: <https://www.heise.de/hintergrund/Vue-js-3-Reactivity-System-und-Composition-API-unter-der-Lupe-6032080.html>

Referenzen

Dirk Hering: Analyse von Methoden und Werkzeugumgebungen zur plattformunabhängigen Entwicklung mobiler Applikationen, Diplomarbeit, TU Dresden, 2010

Kramer, Dean ; Clark, Tony ; Oussena, Samia: MobDSL: A Domain Specific Language for multiple mobile platform deployment. In: 2010 IEEE International Conference on Networked Embedded Systems for Enterprise Applications (NESEA), 2010, S. 1–7

Hamann, Thomas ; Hübsch, Gerald ; Springer, Thomas: A Model-Driven Approach for Developing Adaptive Software Systems. DAIS 2008, Proceedings Bd. 053 (LNCS), pp. 196–209, 2008

Kassinen, Otso ; Harjula, Erkki ; Koskela, Timo ; Ylianttila, Mika: Guidelines for the Implementation of Cross-platform Mobile Middleware. In: International Journal of Software Engineering and Its Applications (IJSEIA) 4 (2010), Nr. 3, S. 43–57

Referenzen

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>

Google Web Toolkit: <http://code.google.com/intl/de-DE/webtoolkit>

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>