# Beyond Web/Native/Hybrid: A New Taxonomy for Mobile App Development

Robin Nunkesser
Hamm-Lippstadt University of Applied Sciences
Mobile Computing
Hamm, Germany
robin.nunkesser@hshl.de

## ABSTRACT

Currently, mobile operating systems are dominated by the duopoly of iOS and Android. App projects that intend to reach a high number of customers need to target these two platforms foremost. However, iOS and Android do not have an officially supported common development framework. Instead, different development approaches are available for multi-platform development.

The standard taxonomy for different development approaches of mobile applications is: Web Apps, Native Apps, Hybrid Apps. While this made perfect sense for iPhone development, it is not accurate for Android or cross-platform development, for example.

In this paper, a new taxonomy is proposed. Based on the fundamental difference in the tools and programming languages used for the task, six different categories are proposed for everyday use: Endemic Apps, Web Apps, Hybrid Web Apps, Hybrid Bridged Apps, System Language Apps, and Foreign Language Apps. In addition, when a more precise distinction is necessary, a total of three main categories and seven subcategories are defined.

The paper also contains a short overview of the advantages and disadvantages of the approaches mentioned.

## CCS CONCEPTS

• **Human-centered computing** → **Ubiquitous and mobile computing systems and tools**; **Smartphones**; **Tablet computers**; *Mobile computing*; • **Software and its engineering** → **Development frameworks and environments**; • **Information systems** → *Mobile information processing systems*;

## KEYWORDS

Mobile App, Native, Web, Hybrid, Android, iOS

**ACM Reference Format:**
Robin Nunkesser. 2018. Beyond Web/Native/Hybrid: A New Taxonomy for Mobile App Development. In *MOBILESoft '18: 5th IEEE/ACM International Conference on Mobile Software Engineering and Systems, May 27–28, 2018, Gothenburg, Sweden.* ACM, New York, NY, USA, 5 pages.
https://doi.org/10.1145/3197231.3197260


*MOBILESoft '18, May 27–28, 2018, Gothenburg, Sweden*

ACM ISBN 978-1-4503-5712-8/18/05…$15.00
https://doi.org/10.1145/3197231.3197260

## 1 INTRODUCTION

The terms web, native, and hybrid make perfect sense from a historic perspective. Back in 2007, there was no Android and the newly introduced iPhone only allowed third party developers to program Web Apps. Consider this famous line from Steve Jobs' WWDC 2007 keynote[1] where the term Web App was explicitly used:

> "The full Safari engine is inside of iPhone. And so, you can write amazing Web 2.0 and Ajax apps that look exactly and behave exactly like apps on the iPhone."

Fortunately, the iPhone was opened for native development in 2008. In the corresponding WWDC 2008 keynote by Steve Jobs[2] the term Native (App) was explicitly used:

> "With the SDK in iPhone 2.0 we're opening the same native APIs and tools we use internally."

The terms Web App and Native App were also used from the beginning of the iPhone Human Interface Guidelines [8].

With these two possibilities available, authors such as Christopher Allen, Shannon Appelcline [1], and Lee S. Barney [2] among others started mixing web and native development under the term hybrid development (an approach that is currently dominated by Adobe PhoneGap / Apache Cordova[3]).

Nowadays, this is still the typical taxonomy used: apps are often divided into the three categories Web Apps, Native Apps, and Hybrid Apps.

However, the advent of Android and numerous cross-platform tools since 2008 have led to several problems with this taxonomy. The term Native Android Apps commonly describes those written in Java (and since Android Studio 3.0 also those written in Kotlin) while strictly speaking it should be used to describe C/C++-Apps written with the Android NDK (Native Development Kit). In fact, when you consider the term Native Development Kit, this also reflects Google's terminology.

Being this strict in defining Native Apps does not make sense, because with this viewpoint, Native Apps would describe different approaches on iOS and Android (which is clearly not how most people use the term).

In contrast, Hybrid Apps as a term is used nowadays to describe a lot more than the mixture of web and native development. When Windows Phone 7 was introduced in 2010, up to five more

[1] http://events.apple.com.edgesuite.net/d7625zs/event/
[2] https://itunes.apple.com/de/podcast/wwdc-2008-keynote-address/id275834665
[3] https://phonegap.com and http://cordova.apache.org

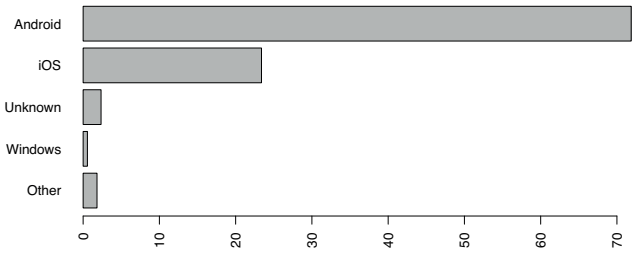


**Figure 1: Mobile Operating System Market Share Worldwide Feb 2018**

or less relevant smartphone OS existed simultaneously and cross-platform development became increasingly important with numerous tools on the market. These tools offer a lot of different approaches. The term hybrid is often also used for approaches that use web technology, for example, without building a strictly defined hybrid app. Beyond this, some cross-platform frameworks use approaches where neither the term web, nor native or hybrid fits.

Nowadays, the OS market is reduced to the duopoly of iOS and Android (see for instance Figure 1 for market share with regard to browser usage taken from http://gs.statcounter.com).

However, the problems of the terms web / native / hybrid with regard to Android and cross-platform tools are not changed by the duopoly we have at present.

Clearly, while it is useful to have only three different categories, they are no longer sufficient.

### 1.1 Related Work

Most papers accept the basic categories web / native / hybrid sometimes adding categories and subdivisions. Turau and Ohrt [15] are among the few commenting on difficulties with the term *native*. They propose the terms Integrated Apps and Nonintegrated Apps instead.

Among the first extended taxonomies is Behrens' taxonomy [3] presented at MobileTechCon 2010 (also published in his blog[4]). His taxonomy was later also used by Xanthopoulos and Xinogalos [17]

Some papers propose subdivisions of cross-platform (typically implicitly accepting the categories web / native / hybrid). Raj and Tolety [13] use Web, Hybrid, Interpreted, and Cross-compiled while Ribeiro and da Silva [14] propose Runtime, Web-to-native wrapper, App Factory, and Domain Specific Language. Perchat proposes Cross-Compilation, Model-Driven Engineering, and Interpreters (Virtual Machines and Web-Based). El-Kassas et al. [6] propose Compilation, Component-Based, Interpretation, Modeling, Cloud-Based, and Merged.

Further related work may be found in [4, 5, 9–12]

### 1.2 Behrens' Taxonomy

Figure 2 shows the taxonomy proposed by Behrens [3].

It incorporates the classic three categories web, native, and hybrid. In addition, he uses the terms interpreted and generated.

[4]http://heikobehrens.net/2010/10/11/cross-platform-app-development-for-iphone-android-co-—-a-comparison-i-presented-at-mobiletechcon-2010/

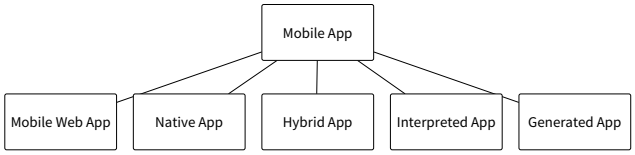


**Figure 2: Behrens categories Web, Native, Hybrid, Interpreted, Generated**

Behrens defines Interpreted Apps in the following words:

> "Interpreted apps use platform-specific native UI elements to interact with the user whereas the application logic is captured in a platform-independent way."

His definition for Generated Apps is the following:

> "The common idea of [...] [generated apps] is to produce truly native apps for each targeted platform with its respective programming language from a single code-base."

In 2013 Xanthopoulos and Xinogalos [17] used the same categories with more detailed explanations.

While this is a much more precise taxonomy than web / native / hybrid, it is very much focused on technical aspects. In addition, the current state of cross-platform development does not really suit the taxonomy. On the one hand, code generation for mobile apps was not as successful as Behrens had hoped (Behrens [3] as well as Xanthopoulos and Xinogalos [17] mention three frameworks for Generated Apps: XMLVM, iPhonical, and APPlause; none of the three is active anymore). On the other hand, some very successful cross-platform frameworks like Xamarin[5] do not fit in either of Behrens five categories or they use different approaches for different target platforms.

## 2 THE VISION: A NEW TAXONOMY FOR MOBILE APP DEVELOPMENT

One of the hardest parts of introducing a new taxonomy is the right wording. A major advantage of the terms native, web, and hybrid is the easy wording. Back in 2007 and 2008, it was easy to remember, accurate, and neutral.

Therefore, our proposed taxonomy uses a twofold approach with regard to wording. Firstly, we propose a clean division and subdivision and corresponding categories. Secondly, we propose to use only a subset of the possible divisions and appropriate wording.

The taxonomy we propose here, has three categories as a main division. The first category should contain what is at the moment imprecisely termed native: Apps that are developed with the SDKs of the OS vendors. We propose to call them Endemic Apps (from the Greek εν δήμος; "within people"). Let us for a moment use fitting terms for two other categories: Pandemic (from the Greek πᾶν δήμος; "all people") and Ecdemic (from the Greek εκ δήμος; "out people") Apps.

Figure 3 shows the reasoning behind this taxonomy.

[5]https://www.xamarin.com

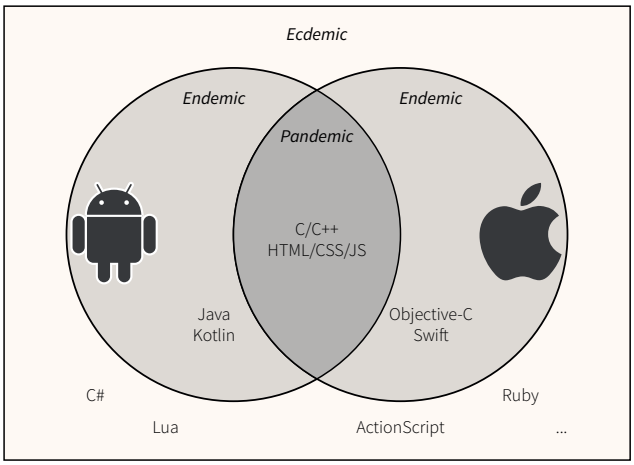


**Figure 3: Main categories Endemic, Pandemic, and Ecdemic**

## 2.1 Endemic Apps

All of the major mobile OS vendors also provide IDEs and SDKs for their operating systems. Apple's Xcode is shipped with the iOS SDK, which mainly makes it possible to program apps in Objective-C and Swift. Android Studio by Google and IntelliJ makes it possible to program with the Android SDK in Java and Kotlin. We propose using the term Endemic Apps for apps built with these SDKs.

## 2.2 Pandemic Apps

Some technologies are supported by every major mobile OS. The major mobile OS all support HTML, CSS, and JavaScript. Web pages that are optimised for mobile OS are typically called Web Apps, which is perfectly fine. Especially on Android Web Apps play a more prominent role nowadays under the term Progressive Web Apps.

Apache Cordova and Adobe PhoneGap mix the web app idea with an endemic app shell which leads to so called hybrid apps. We propose calling these apps Hybrid Web Apps to underline the point that it is a hybrid using web technology (as opposed to other possible hybrid approaches).

Frameworks like Appcelerator Titanium[6], React Native[7], and Flutter[8] use a slightly different hybrid approach. They also use the endemic JavaScript engines from an endemic App shell, but also allow to use endemic User Interface elements instead of HTML. We propose calling these apps Hybrid Bridged Apps (to distinguish them from Web Apps and underline the fact that a bridge from JavaScript to endemic languages is used).

Yet, the major mobile OS all support C/C++. This is used, for example, in game development with frameworks such as Unreal Engine[9] and Unity[10] which have C++ runtimes. Qt[11] also offers the possibility of writing mobile apps in C/C++. A fitting term for apps written in C/C++ is System Language Apps.

Since Pandemic App is not an agreeable term for everyday usage, we propose using the terms Web App, Hybrid Web App, Hybrid Bridged App, and System Language App for the specific kinds of Pandemic Apps instead.

## 2.3 Ecdemic Apps

Cross-platform frameworks such as Xamarin take a language that is not endemic to the mobile OS and use it for development on the platform. In order to do this, different approaches exist. The language could be made compilable to an endemic virtual machine such as llvm or Android Runtime (ART) (which would offer the same possibilities as for endemic apps). There are three commonly used approaches to this so called cross language compilation: direct source to source translation, indirect source code to intermediate abstract language translation and direct compilation into a platform's binary format [16]. We propose calling the resulting apps (Endemic) VM Apps.

Another possibility is generating endemic code from the ecdemic language (this is the same case as Behren's Generated Apps) or with some kind of Model-Driven approach. The third possibility is an interpreter that gets shipped together with the app (this is the same case as Behren's Interpreted Apps).

We propose not differentiating between the three possibilities VM Apps, Generated Apps, and Interpreted Apps. In real frameworks a combination of these approaches is often used. Take Xamarin as an example. Xamarin Apps are interpreted on Android and run on the llvm on iOS. It is therefore better to simply use a single term. We propose calling ecdemic apps Foreign Language Apps.

## 2.4 Final Taxonomy

Figure 4 shows the final proposed categories to use: Endemic App, Web App, Hybrid Web App, Hybrid Bridged App, System Language App, and Foreign Language App.

## 2.5 Short Comparison of the Approaches Discussed

Behrens [3] also includes a comparison in his taxonomy. The comparison here uses similar criteria. Another approach with different criteria is presented in Heitkötter, Hanschke, and Majchrzak [7].

A major disadvantage of pure Web Apps is that Apple's Review Guidelines do not allow those apps to be distributed in the App Store. However, Hybrid Web Apps are admitted in the App Store if they incorporate enough differentiating functionality when compared to a corresponding Web App. All of the other approaches (Endemic Apps, Hybrid Bridged Apps, System Language Apps, and Foreign Language Apps) are allowed in the App Store. More precisely, Foreign Language Apps are not automatically allowed in the App Store, but a framework for Foreign Language Apps without the possibility of being distributed in the App Store makes no sense and may therefore be ignored.

A major disadvantage of Foreign Language Apps is vendor lock-in. If the framework you use is discontinued, all app investments may be in danger.

[6] https://www.appcelerator.com/mobile-app-development-products/
[7] https://facebook.github.io/react-native/
[8] https://flutter.io
[9] https://www.unrealengine.com
[10] https://unity3d.com
[11] https://www.qt.io/mobile-app-development/

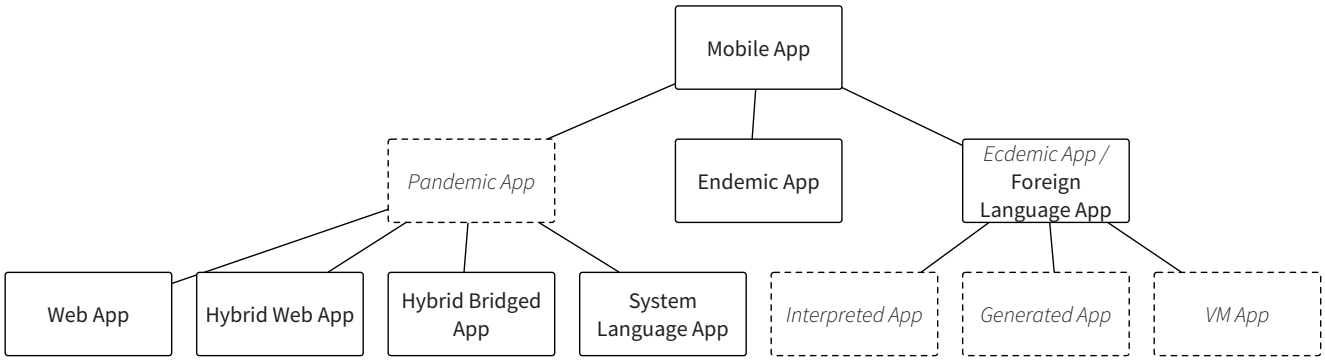


**Figure 4: Proposed Taxonomy Web App, Hybrid Web App, Hybrid Bridged App, System Language App, Endemic App, and Foreign Language App**

*2.5.1 Code Sharing.* The amount of code sharing is at its lowest in Endemic Apps. A major point of using pandemic or ecdemic approaches is to share code. Code sharing is also possible for Endemic Apps, however. You can, for example, share test code with a cross-platform UI testing tool such as Appium[12] or by generating parts of the code from models.

In practice, code sharing in pandemic apps tends to be more common than in ecdemic apps. The reason for this is that business logic and UI are typically written with shared code in pandemic apps, while ecdemic frameworks often offer to mix in more endemic code (especially for the UI). For apps of lesser complexity, however, code sharing could be nearly 100 % for both approaches.

*2.5.2 User Experience.* From the perspective of UX, Endemic Apps offer the best performance, full native UI and all of the possibilities of the OS. Pandemic apps (with the exception of Hybrid Bridged Apps), on the other hand, lack the possibility of using a native UI which may reduce UX. Performance may also be a problem for Web Apps and Hybrid Apps while System Language Apps may in some cases even outperform Endemic Apps.

Foreign Language Apps from the leading frameworks typically make it possible to use native UI and nearly all capabilities of the OS. Performance may be an issue when the code is generated or the apps are shipped with an interpreter. When the interpreter is shipped within the app itself, the download size of the app binary is also increased. In some cases, apps based on the same interpreter even redundantly install duplicate versions of this interpreter.

New features of the OS need to be made accessible by the framework vendors. This may not always be available as swiftly for Endemic Apps.

*2.5.3 Hardware Accessibility.* The accessibility of device hardware like sensors and actuators falls roughly under the same category of considerations as UX. Endemic Apps give full access, while Web Apps give only limited access. All the other approaches give varying possibilities depending on the specific vendor.

Progressive Web Apps are a current trend mainly pushed by Android to extend browser engines (and web widgets as well) with more and more traditional endemic features. The result could be a fully-fledged platform that may actually compete with Endemic Apps at some point in the future.

*2.5.4 Offline Content.* All apps apart from Web Apps may be distributed with static content and have access to the embedded SQLite database and the local or cloud file system of the devices. (Progressive) Web Apps may use Web Storage, IndexedDB, and Service Worker for storing content offline.

*2.5.5 App Updates.* Most content in Web Apps is dynamic while the other approaches need to incorporate backend communication for dynamic content. For iOS Apps it may even take weeks to change content because the review process must be incorporated.

## 3 WHY IS IT NEW?

To our knowledge, no publication apart from Turau and Ohrt [15] ever questioned the term *native*. While there are a lot of publications dealing with the categorisation of cross-platform approaches, none of them uses a hierarchical approach. In addition to the hierarchical categorisation, the approach proposed here also offers some flexibility and extensibility.

## 4 THE RISKS

The terms web / native / hybrid are used as standard terms for almost ten years now. It may be to late to replace a term like native that is so widely used. However, the least we can do is to draw attention to the cases where terms like native or hybrid are clearly used wrong or misleading.

## 5 NEXT STEPS

The classic taxonomy for mobile application development web / native / hybrid does not fit the current state of the mobile OS and cross-platform tools market. The duopoly of iOS and Android and the multitude of cross-platform tools with different approaches require a new taxonomy.

As a necessary next step this new taxonomy will need backing and marketing to be adopted. To make adoption easier, the taxonomy still has some flexibility and will hopefully spark a discussion.

[12] https://appium.io

## REFERENCES


[1] Christopher Allen and Shannon Appelcline. 2008. *iPhone in Action: Introduction to Web and SDK Development.* Manning Publications Co., Greenwich, CT, USA.

[2] Lee S. Barney. 2009. *Developing Hybrid Applications for the iPhone: Using HTML, CSS, and JavaScript to Build Dynamic Apps for the iPhone* (1st ed.). Addison-Wesley Professional, Boston, MA.

[3] Heiko Behrens. 2010. Cross-Platform App Development for iPhone, Android & Co. — A Comparison. (2010). MobileTechCon.

[4] Andre Charland and Brian Leroux. 2011. Mobile Application Development: Web vs. Native. *Commun. ACM* 54, 5 (May 2011), 49–53.

[5] Isabelle Dalmasso, Soumya Kanti Datta, Christian Bonnet, and Navid Nikaein. 2013. Survey, comparison and evaluation of cross platform mobile application development tools. In *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC 2013)*. Curran Associates, Inc. for IEEE, Red Hook, NY, 323–328.

[6] Wafaa S. El-Kassas, Bassem A. Abdullah, Ahmed H. Yousef, and Ayman M. Wahba. 2017. Taxonomy of Cross-Platform Mobile Applications Development Approaches. *Ain Shams Engineering Journal* 8, 2 (2017), 163 – 190.

[7] Henning Heitkötter, Sebastian Hanschke, and Tim A. Majchrzak. 2013. Evaluating Cross-Platform Development Approaches for Mobile Applications. In *Web Information Systems and Technologies: 8th International Conference, WEBIST 2012, Porto, Portugal, April 18-21, 2012, Revised Selected Papers*, José Cordeiro and Karl-Heinz Krempels (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 120–138.

[8] Apple Inc. 2007. *iPhone Human Interface Guidelines* (2007-09-27 ed.). Apple Inc., Cupertino, CA.

[9] Tim A. Majchrzak, Tor-Morten Grønli, and Andreas Biørn-Hansen. 2017. Comprehensive Analysis of Innovative Cross-Platform App Development Frameworks. In *Proceedings of the 50th Hawaii International Conference on System Sciences*. University of Hawai'i at Manoa, Honolulu, HI, 6162–6171.

[10] Ivano Malavolta. 2016. *Beyond Native Apps: Web Technologies to the Rescue! (Keynote).* ACM, New York, NY, 1–2.

[11] Ivano Malavolta, Stefano Ruberto, Tommaso Soru, and Valerio Terragni. 2015. End Users' Perception of Hybrid Mobile Apps in the Google Play Store. In *2015 IEEE International Conference on Mobile Services*. Curran Associates, Inc. for IEEE, Red Hook, NY, 25–32.

[12] Manuel Palmieri, Inderjeet Singh, and Antonio Cicchetti. 2012. Comparison of Cross-Platform Mobile Development Tools. In *2012 16th International Conference on Intelligence in Next Generation Networks, ICIN 2012*. Curran Associates, Inc. for IEEE, Red Hook, NY, 179–186.

[13] C P Rahul Raj and Seshu Babu Tolety. 2012. A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach. In *2012 Annual IEEE India Conference (INDICON)*. Curran Associates, Inc. for IEEE, Red Hook, NY, 625–629.

[14] André Ribeiro and Alberto Rodrigues da Silva. 2012. Survey on Cross-Platforms and Languages for Mobile Apps. In *Proceedings of the 2012 Eighth International Conference on the Quality of Information and Communications Technology (QUATIC '12)*. IEEE Computer Society, Washington, DC, USA, 255–260.

[15] Volker Turau and Julian Ohrt. 2012. Cross-Platform Development Tools for Smartphone Applications. *Computer* 45 (09 2012), 72–79.

[16] Robert Virkus. 2017. Going Cross-Platform. In *Don't Panic: Mobile Developer's Guide to the Galaxy* (17th ed.), Marco Tabor and Mladenka Vrdoljak (Eds.). Open-Xchange, Bremen, 78–85.

[17] Spyros Xanthopoulos and Stelios Xinogalos. 2013. A Comparative Analysis of Cross-platform Development Approaches for Mobile Applications. In *Proceedings of the 6th Balkan Conference in Informatics (BCI '13)*. ACM, New York, NY, USA, 213–220.