

Mobile Programming and Multimedia

Cross-Platform Frameworks

Prof. Ombretta Gaggi
University of Padua



Scenario



I have an exciting idea for a new app

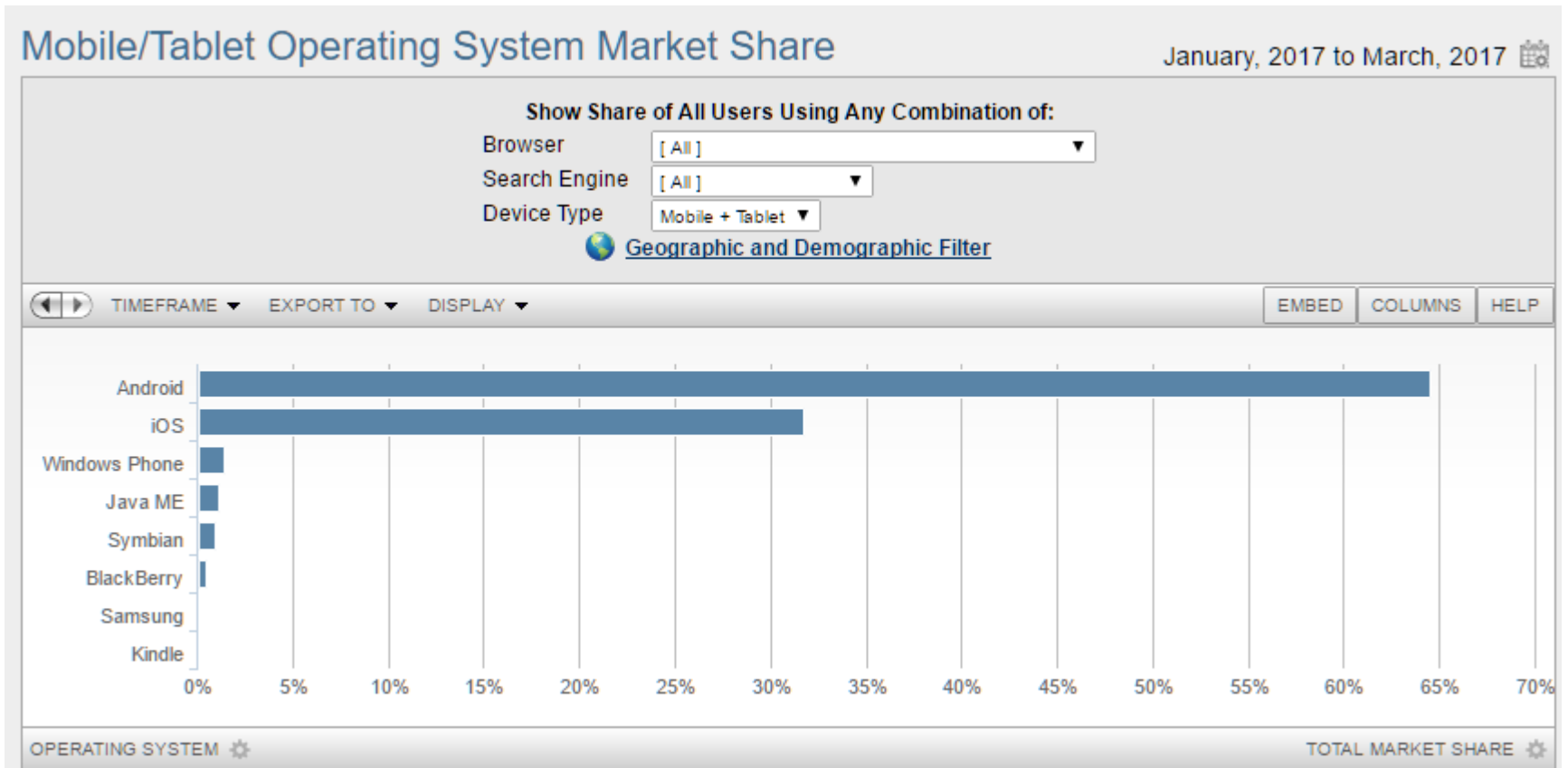


I want to develop this fantastic app

I have an iPhone so...
... I develop it in iOS



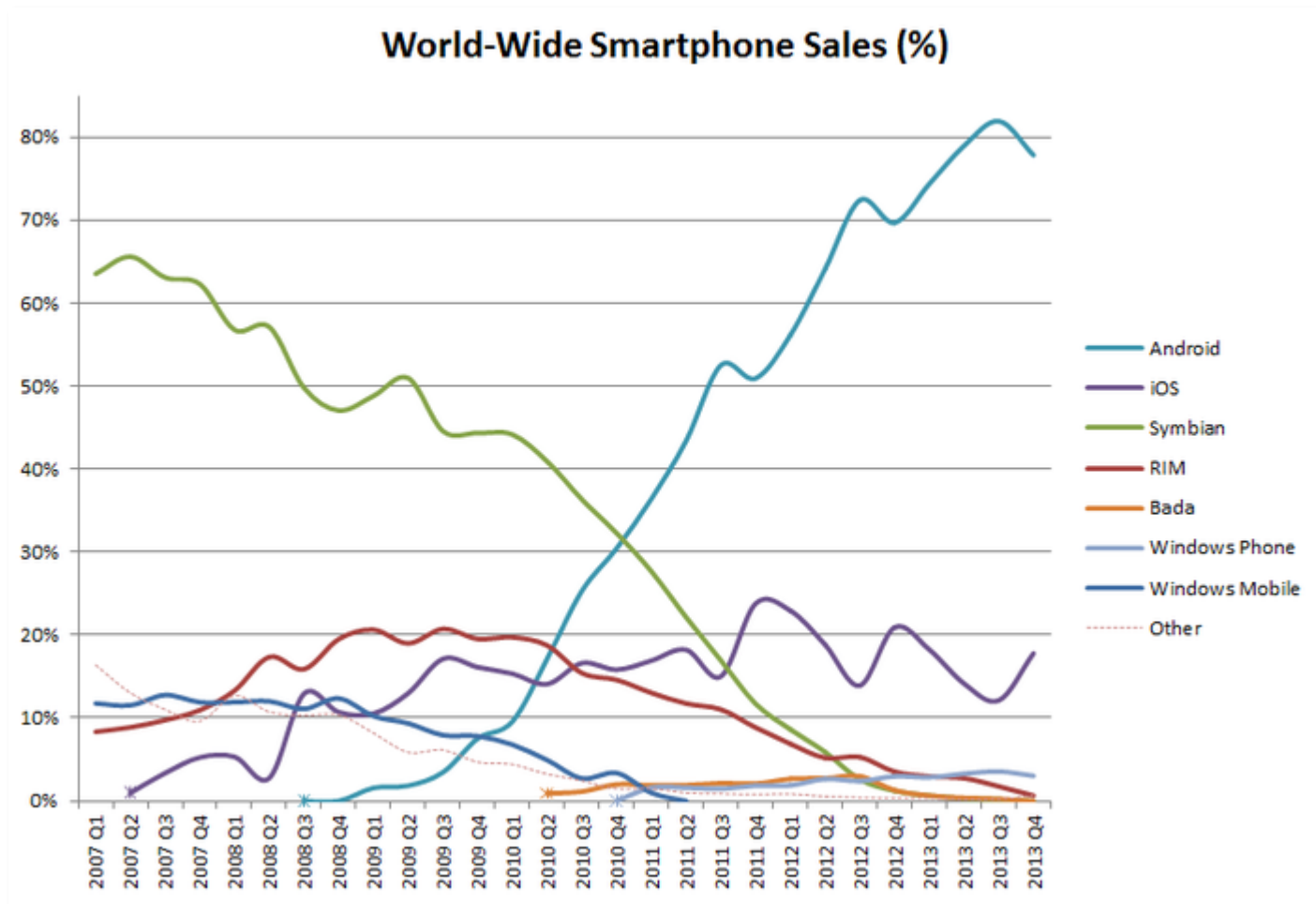
... but Android diffusion is higher than iOS!



<https://www.netmarketshare.com/>



Solution: go back to 2010 or before ;-)

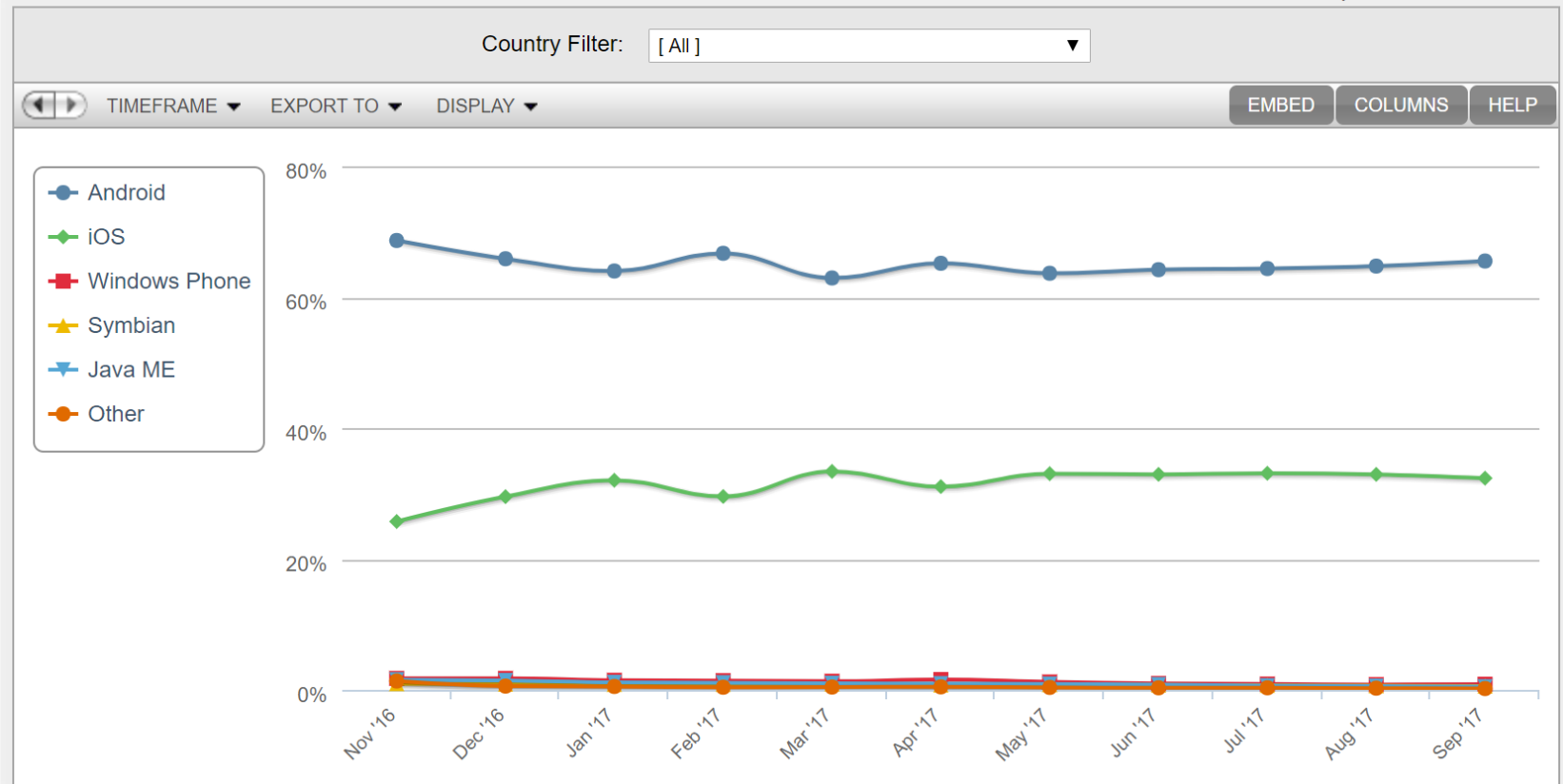


Market fragmentation



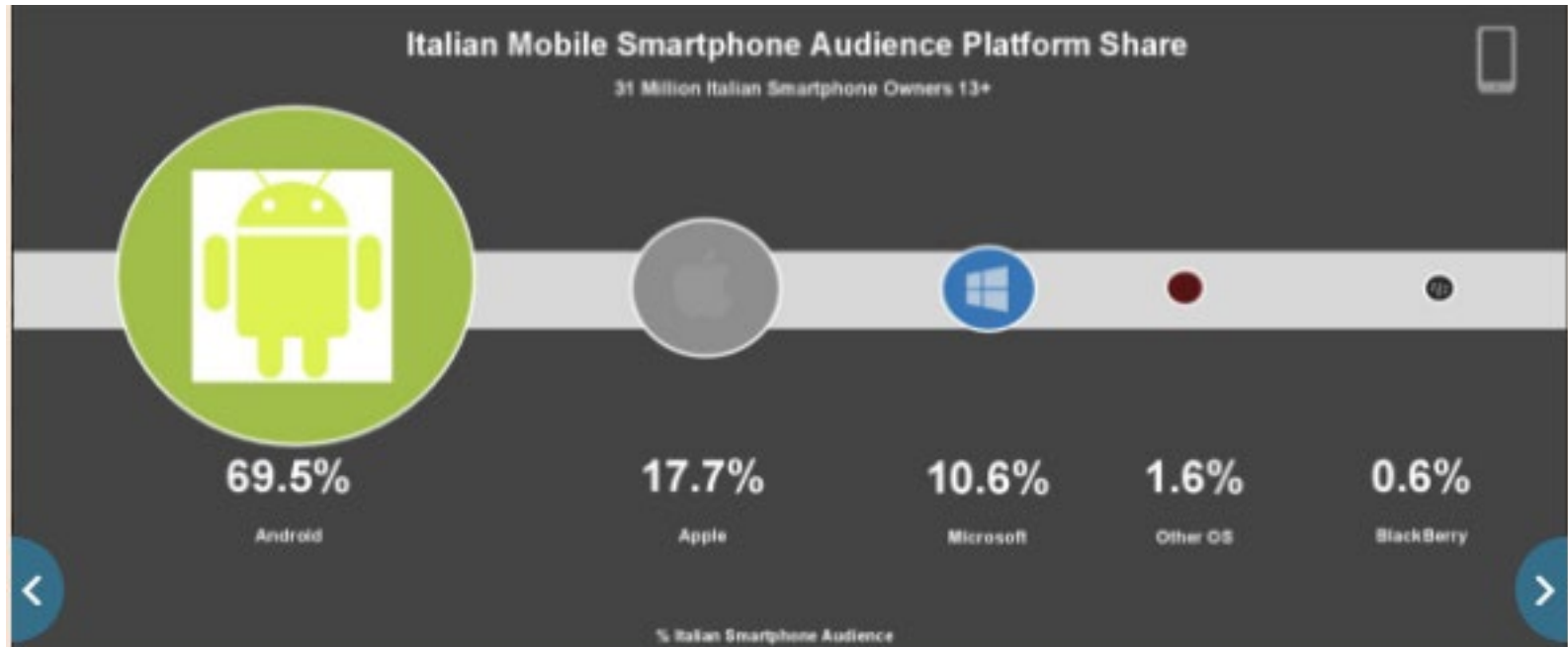
Mobile/Tablet Top Operating System Share Trend

November, 2016 to September, 2017



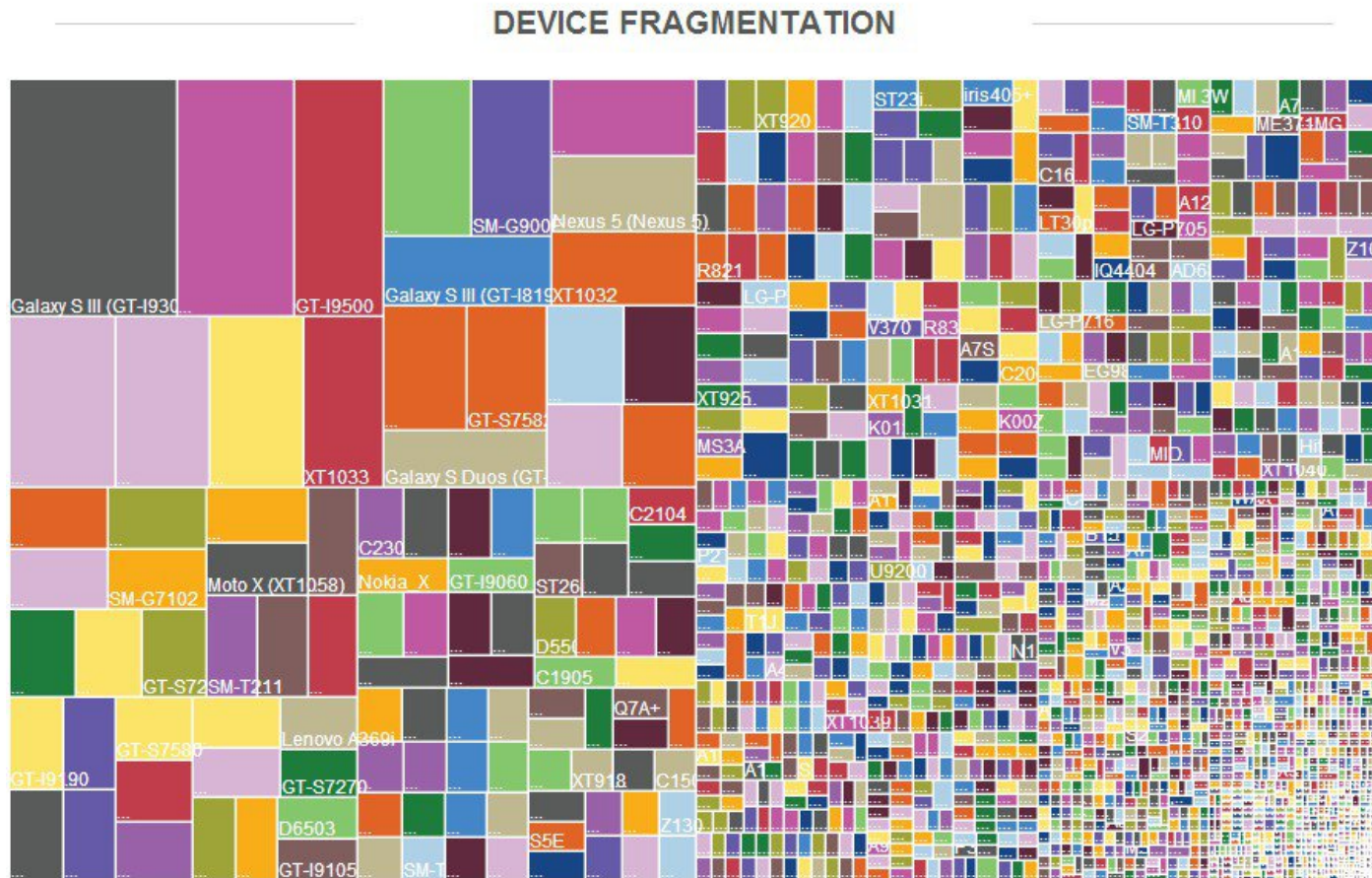
<https://www.netmarketshare.com/>

Italian market fragmentation



<http://www.infodata.ilsole24ore.com/2016/09/02/samsung-e-apple-guidano-il-mercato-smartphone-in-italia-ma-occhio-a-huawei-cresce-del-140/>

Android device fragmentation



<http://android.hdblog.it/2014/08/22/Android-2014-presentati-quasi-19000-device/>

Problem



Different OS => different languages



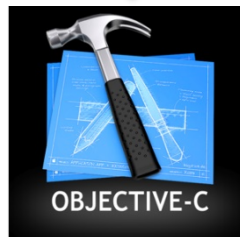
It is necessary to develop different apps (al the same) for several devices



How many?



One for each operating system?



Highly expensive!



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Which are the variables?

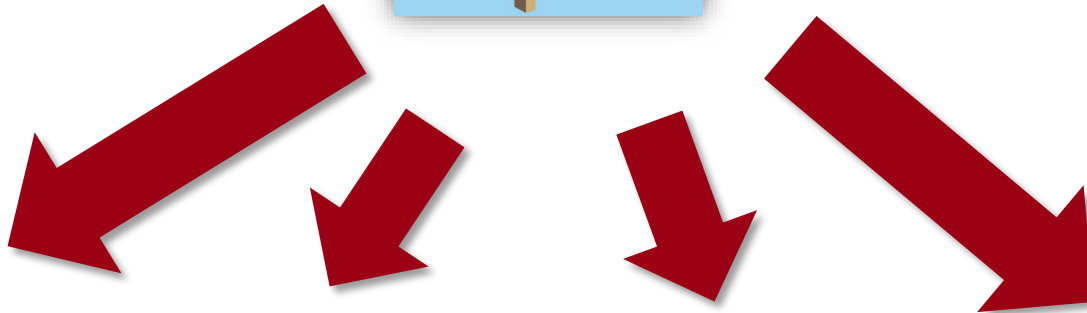


- Operating system
- Programming language
- Development tools (IDE, simulators, etc.)
- API
- Sensors/equipment
- Screen size
- Computational capacity
- ...

Goal



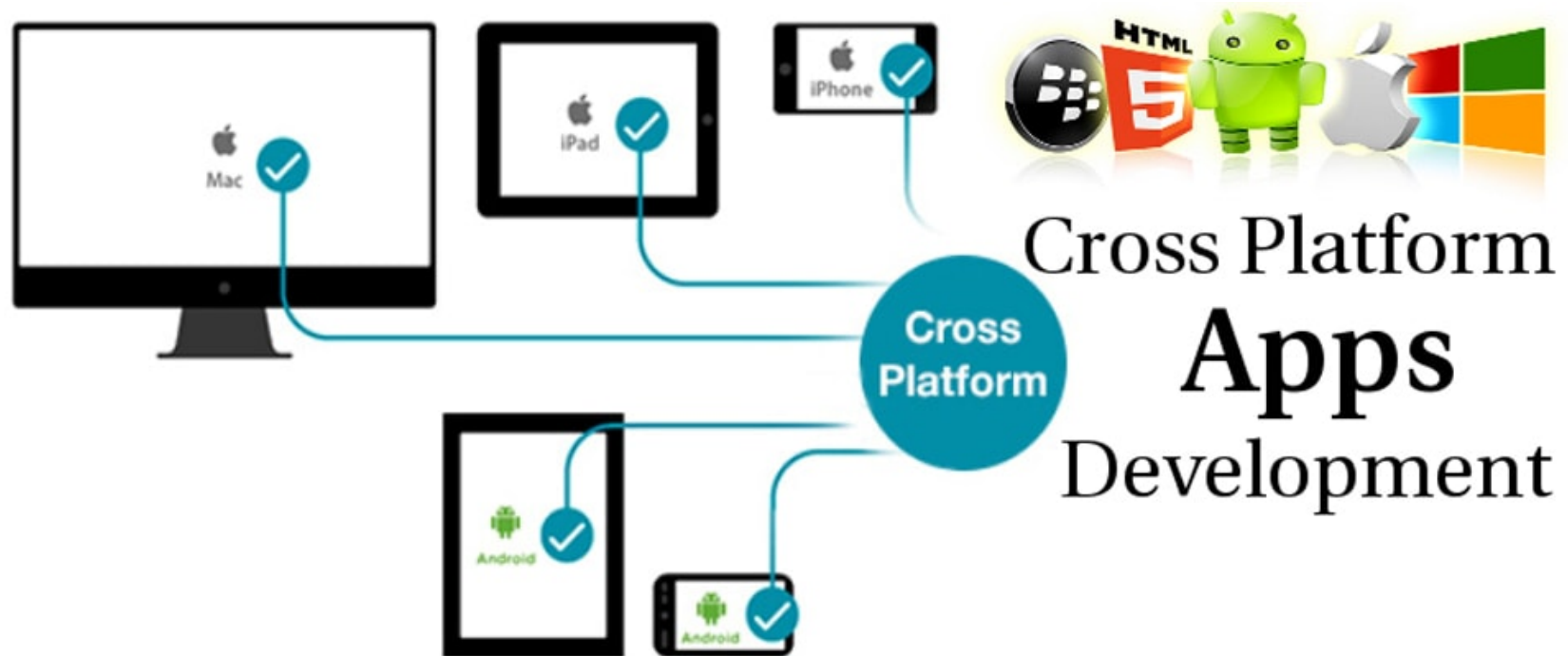
UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Solution



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



A new approach



Cross-platform frameworks for mobile development reduce the negative effects of market fragmentation

«Write *once*, distribute *everywhere*»

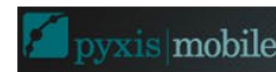
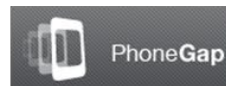


- Application developed on time, using only one programming language (or a set of languages)
- The chosen framework allows the distribution of the application in several applications stores (so, there are several applications deployed)
- The frameworks usually provide support for native API

Really the solution?



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



CROSS PLATFORM TOOLS MARKET OVERVIEW

GLOBAL OPPORTUNITY

Cross Platform Tools Market Size Forecast
(In Billion USD)

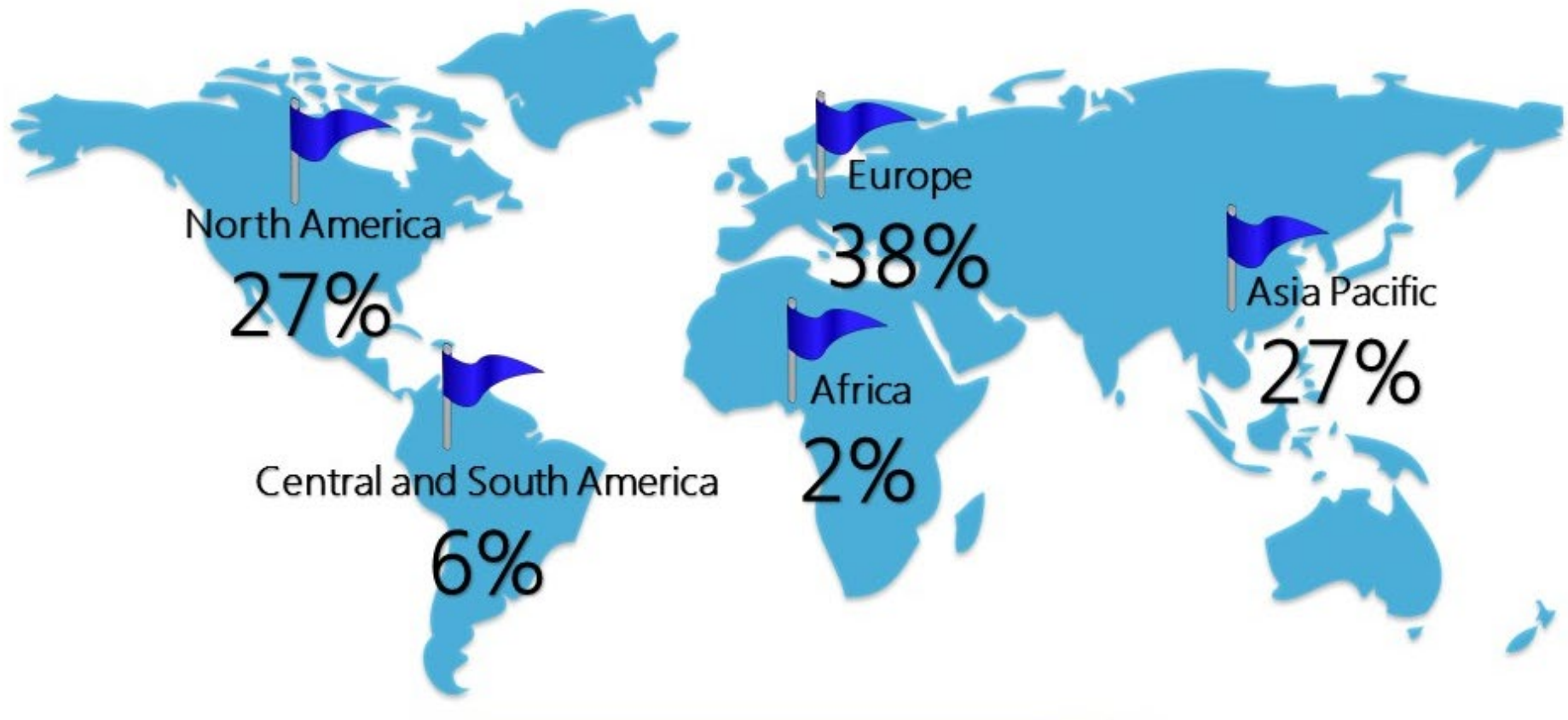


<https://www.rishabhsoft.com/blog/cross-platform-app-tools-infographic>

Geographical distribution



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Benefits



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

COST EFFECTIVE

■ YES ■ AVERAGE ■ NO



TIME SAVING

■ FASTER ■ SAME ■ SLOWER



POPULAR PLATFORMS

% Cross Platform Apps Published on



76%



69%



28%



19%

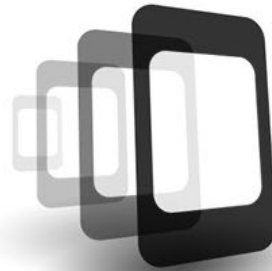
Most popular



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



titanium™



Phone**Gap**



ADOBE® AIR™



Sencha



rhomobile



Xamarin



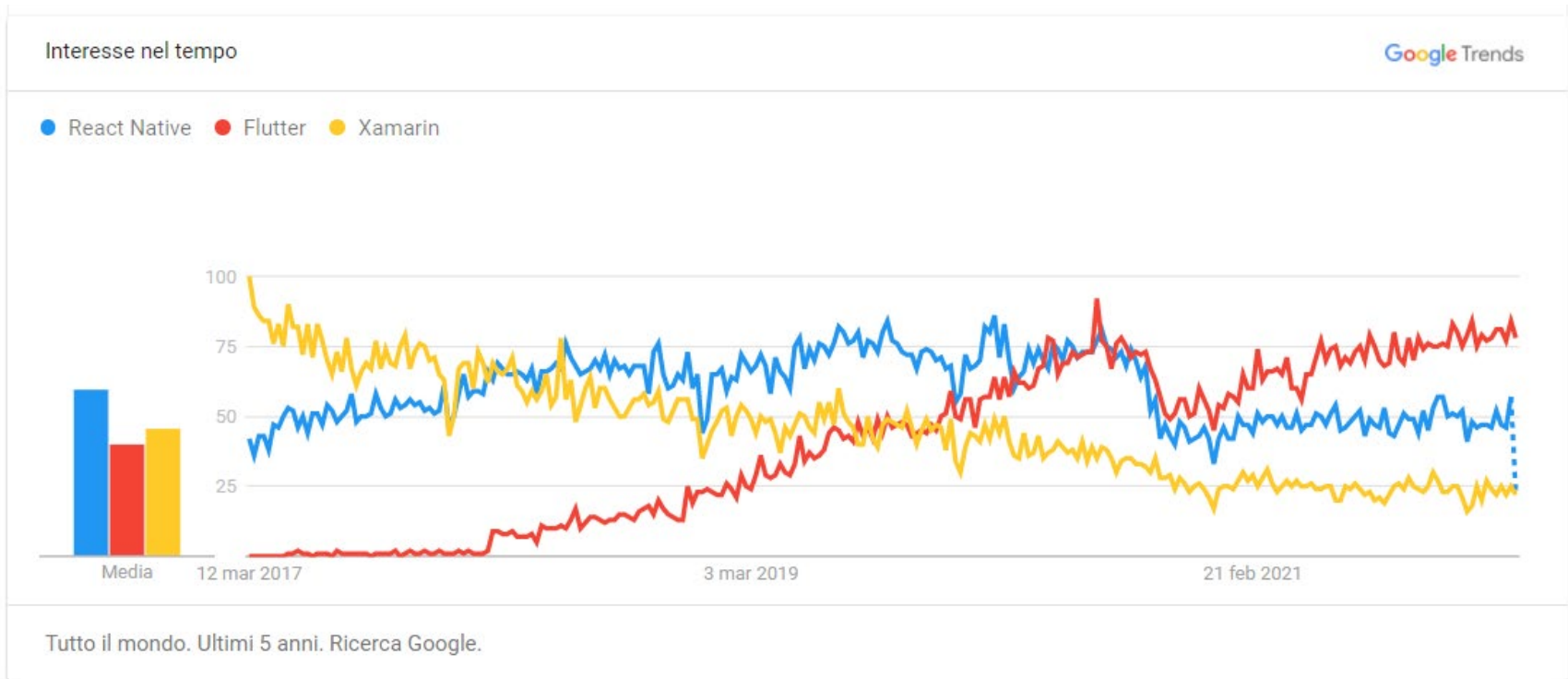
Corona

Pay attention to the stats!

Google statistics



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Pros and cons of cross-platform



Cross-platform mobile development	
pros	cons
wide market reach	possibly slower performance
single codebase	UX and UI discrepancies
faster and cheaper deployment	
reduced workload	
platform consistency	

- Pros of native applications:
 - Usually, a native application offers a better user experience, a faster and more high-performance interaction
 - Non-native applications are limited by the expressivity of the used framework (e.g., available APIs)
 - An Apple computer is always needed
- Cons
 - Fragmentation = higher development costs
 - Problems with test
 - *How to choose the best framework?*

Steps for app development



App development involves 4 steps:

1. Idea analysis
2. Interface design
3. App development
4. Store deployment

Store deployment is necessary every time there is an update and for each platform

Native development requires repeating steps 2-3-4 for each platform

Frameworks' classification is still an open problem

Raj and Tolety classification define 4 different classes:

- *Web Approach*
- *Hybrid Approach*
- *Interpreted Approach*
- *Cross-compiled Approach*

R. Raj, S. Tolety. *A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach*. In INDICON 2012, p. 625-629

Web Approach



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Web Approach: Pros & Cons



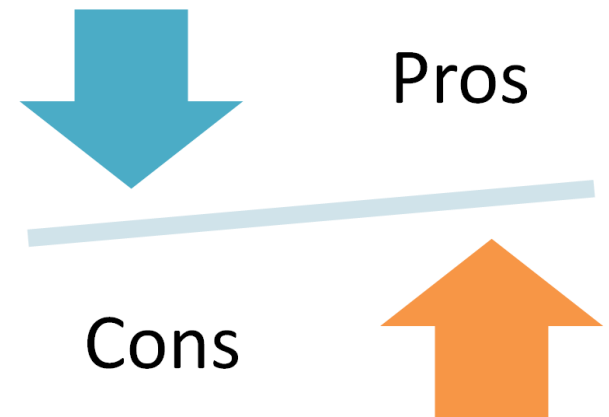
UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Pros:

- + Same interface (but not same experience) on all devices
- + No installation necessary
- + Easy update

Cons:

- No store publishing
- Network connection necessary (but..)
- Difficult test
- Strongly connected to HTML5 support of the device
- Non-native interfaces bring to low usability



Examples



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Sencha  Touch

 React

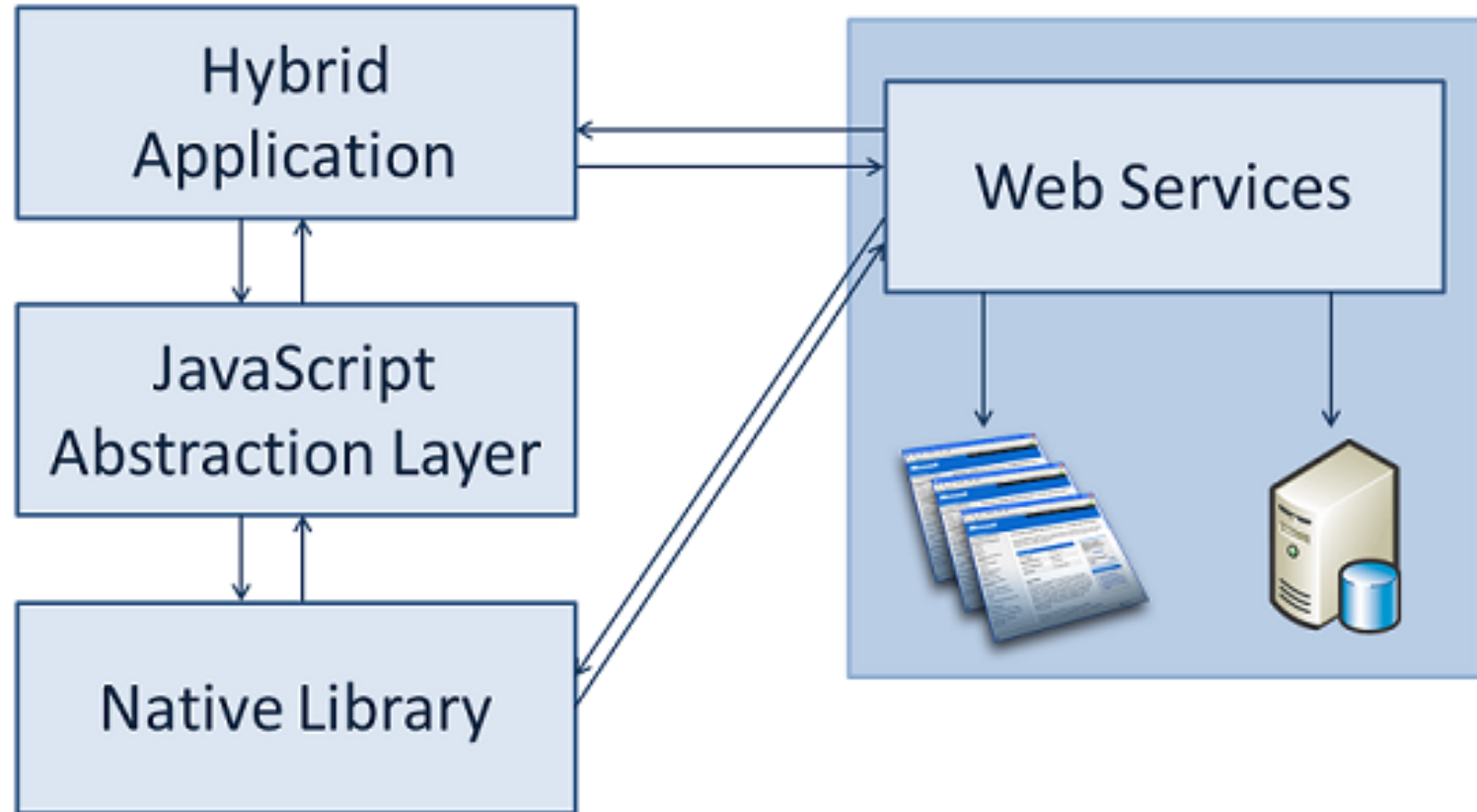


 **jQuery**
mobile

Hybrid Approach



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



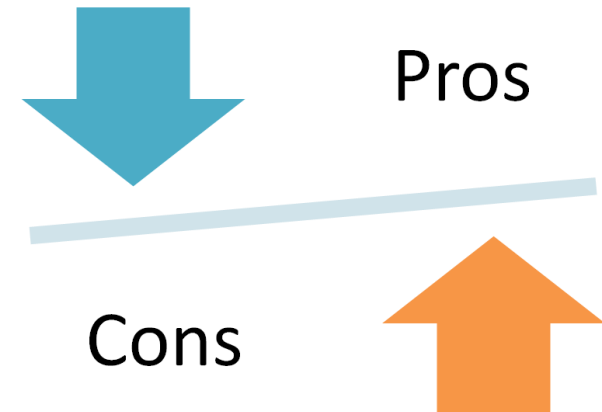
Hybrid Approach: Pros & Cons



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Pros:

- + Store publishing available
- + Reusable UI
- + Usage of device components



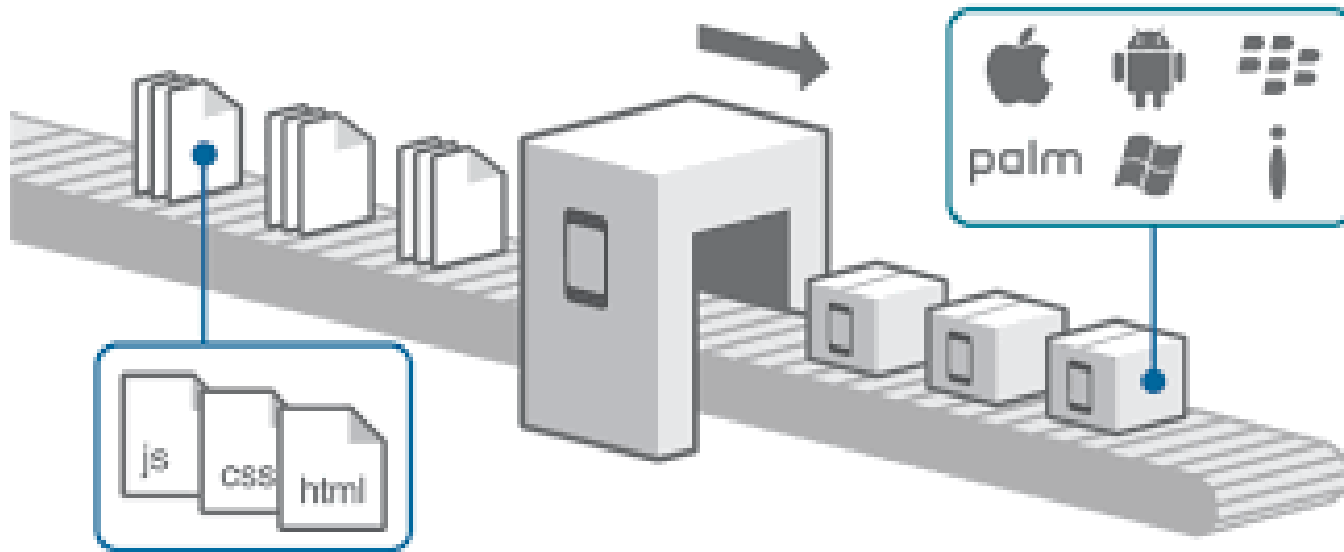
Cons:

- Lower performances
- UI do not follow native Look and Feel

Example: PhoneGap



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

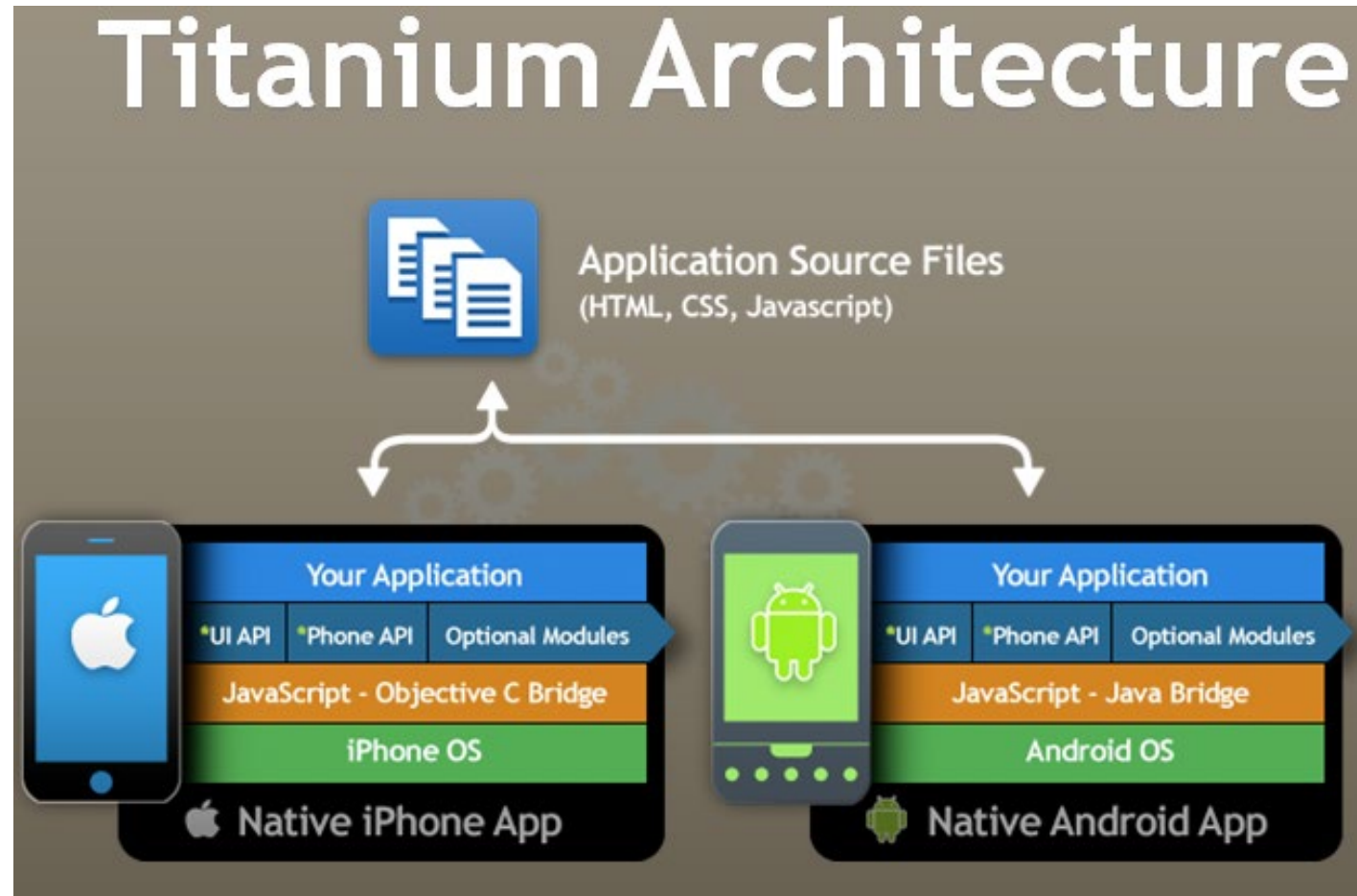


HelloWorld – body



```
<body>
  <div class="app">
    <h1>PhoneGap</h1>
    <div id="deviceready" class="blink">
      <p class="event listening">Connecting to Device</p>
      <p class="event received">Device is Ready</p>
    </div>
  </div>
  <script type="text/javascript" src="cordova.js"></script>
  <script type="text/javascript" src="js/index.js"></script>
  <script type="text/javascript">
    app.initialize();
  </script>
</body></html>
```

- index.js provides an event manager and the application behavior
- Most important events to manage:
 - load
 - *deviceready*: event provided by Cordova API. Fired when Cordova API loading is completed and are available for usage
 - offline
 - online



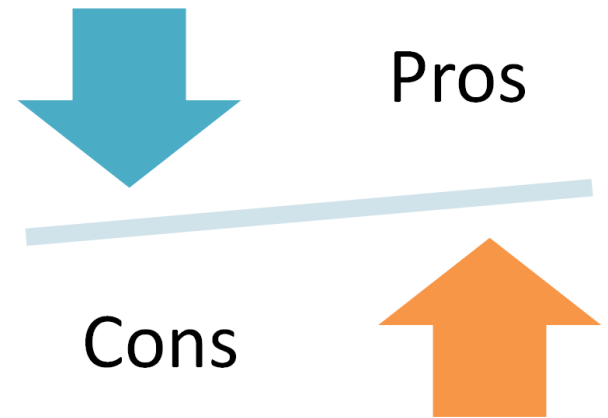
Interpreted Approach: Pros & Cons

Pros:

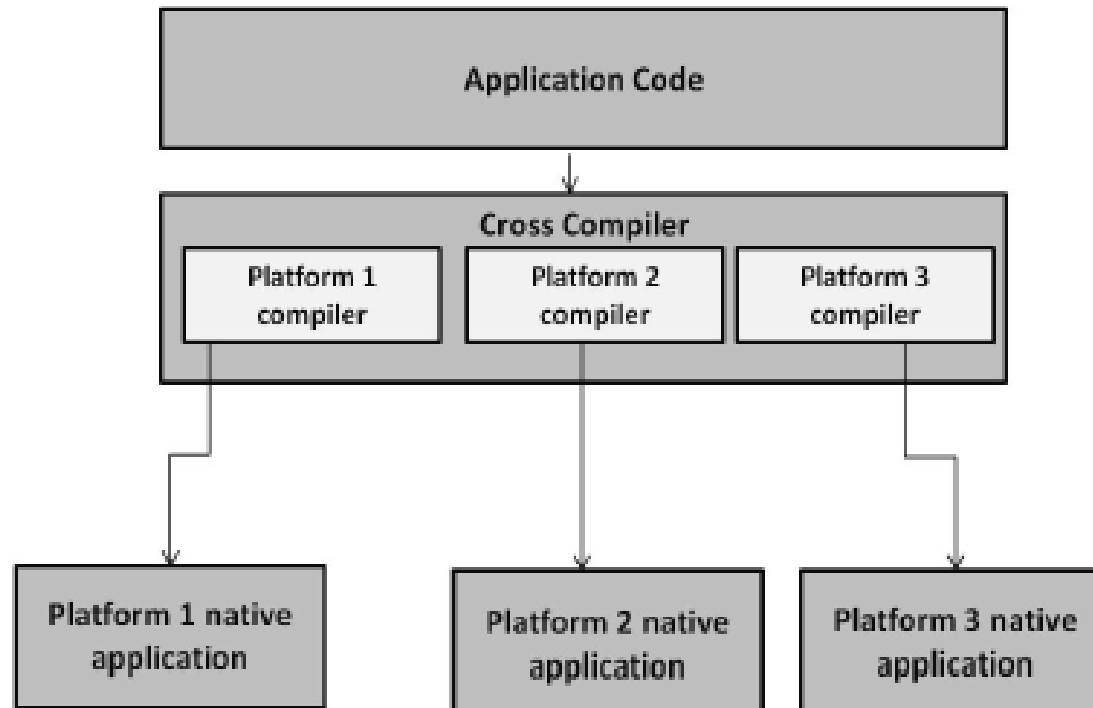
- + Native Look and Feel
- + Store publishing
- + APIs for device components

Cons:

- Really difficult to reuse the UI
- Available features depend on the framework
- The interpreter can have low performances



Cross-compiled Approach



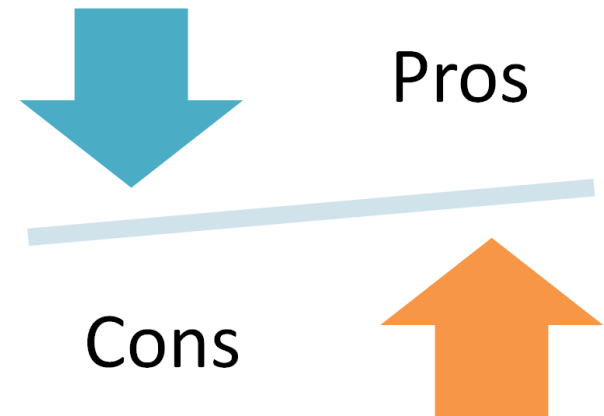
Cross-compiled Approach: Pros & Cons

Pros:

- + Allows to use all the components available
- + Native interface
- + Good performances
- + Store publishing

Cons:

- Not reusable UI
- Very complex apps can have problems during the building process



Cross-platform Frameworks



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Approach	Programming language	Supported platforms	Pros	Cons	Example
Web	HTML, CSS, Javascript	Android, iOS, Windows, BlackBerry ^a	<ul style="list-style-type: none">– Easy to update– No installation– Same UI over different devices	<ul style="list-style-type: none">– No access to application store– Network delays– Expensive testing– No native UI– No native look and feel	jQuery mobile, Sencha Touch
Hybrid	HTML, CSS, Javascript	Android, iOS, Windows, Blackberry,	<ul style="list-style-type: none">– Access to application store^b– Support to most smartphone hardware	<ul style="list-style-type: none">– Platform branching– Interpretation step	PhoneGap
Interpreted	Javascript	Android, iOS, Blackberry	<ul style="list-style-type: none">– Access to application store– Native look and feel	<ul style="list-style-type: none">– UI non reusable– High code conversion complexity for complex applications	Titanium
Cross-Compiled	C#, C++, Javascript	Android, iOS, Symbian	<ul style="list-style-type: none">– Native UI– Real native application		Mono, MoSync

^a Support depends on the browser chosen by the user.

^b Apple store usually tends to refuse applications developed with this approach [34].

M. Ciman, O. Gaggi. *An empirical analysis of energy consumption of cross-platform frameworks for mobile development*, PMC vo. 39, p. 214-230, 2017

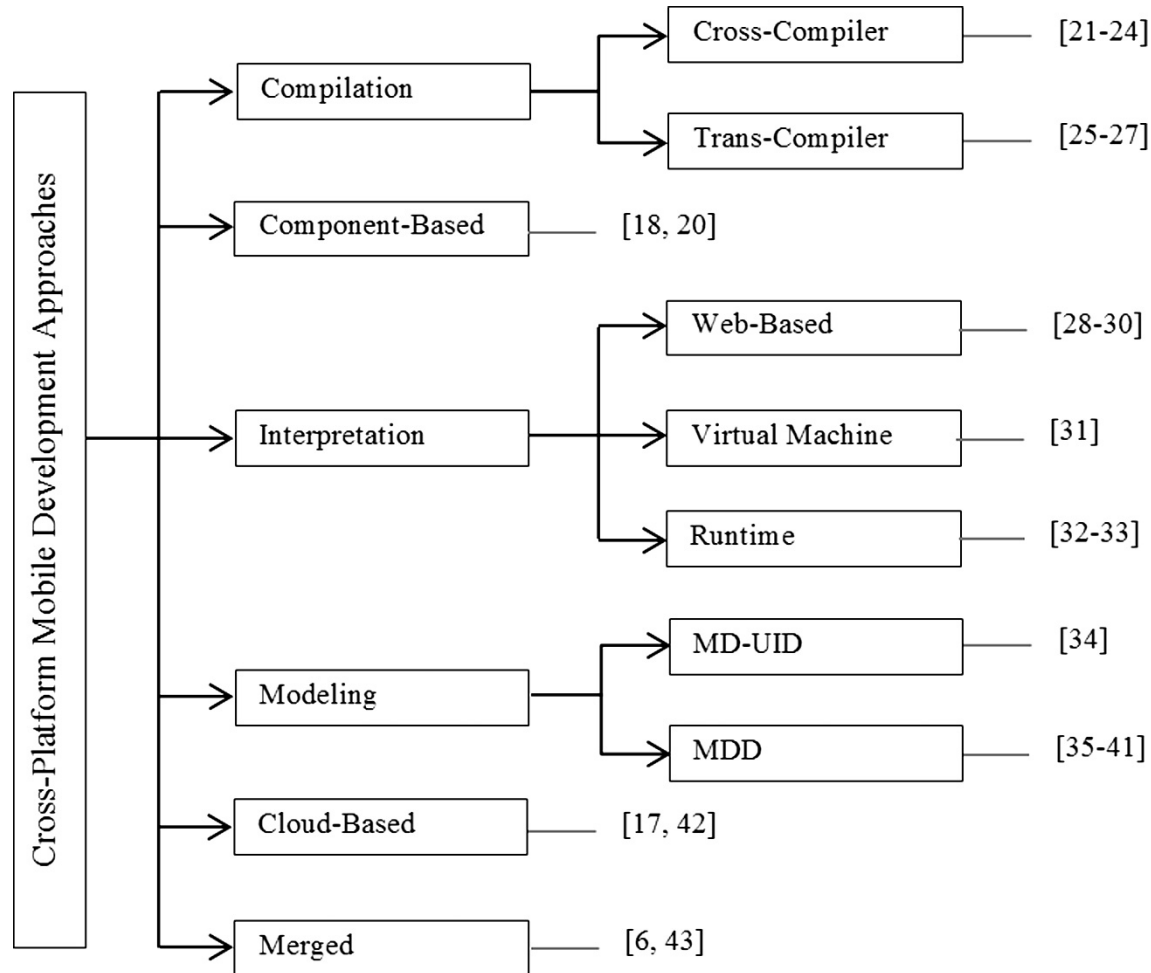
There are several methods for frameworks classification, some of them based on the development approach, others based on the result

El-Kassan et al. divide the apps into three categories

- Native app
- Web app
- Hybrid app

W. S. El-Kassas et al. *Taxonomy of Cross-Platform Mobile Applications Development Approaches*, Ain Shams Engineering J. 8(2), p. 163-190, 2017

Other classifications - 2



W. S. El-Kassas et al. *Taxonomy of Cross-Platform Mobile Applications Development Approaches*, Ain Shams Engineering J. 8(2), p. 163-190, 2017

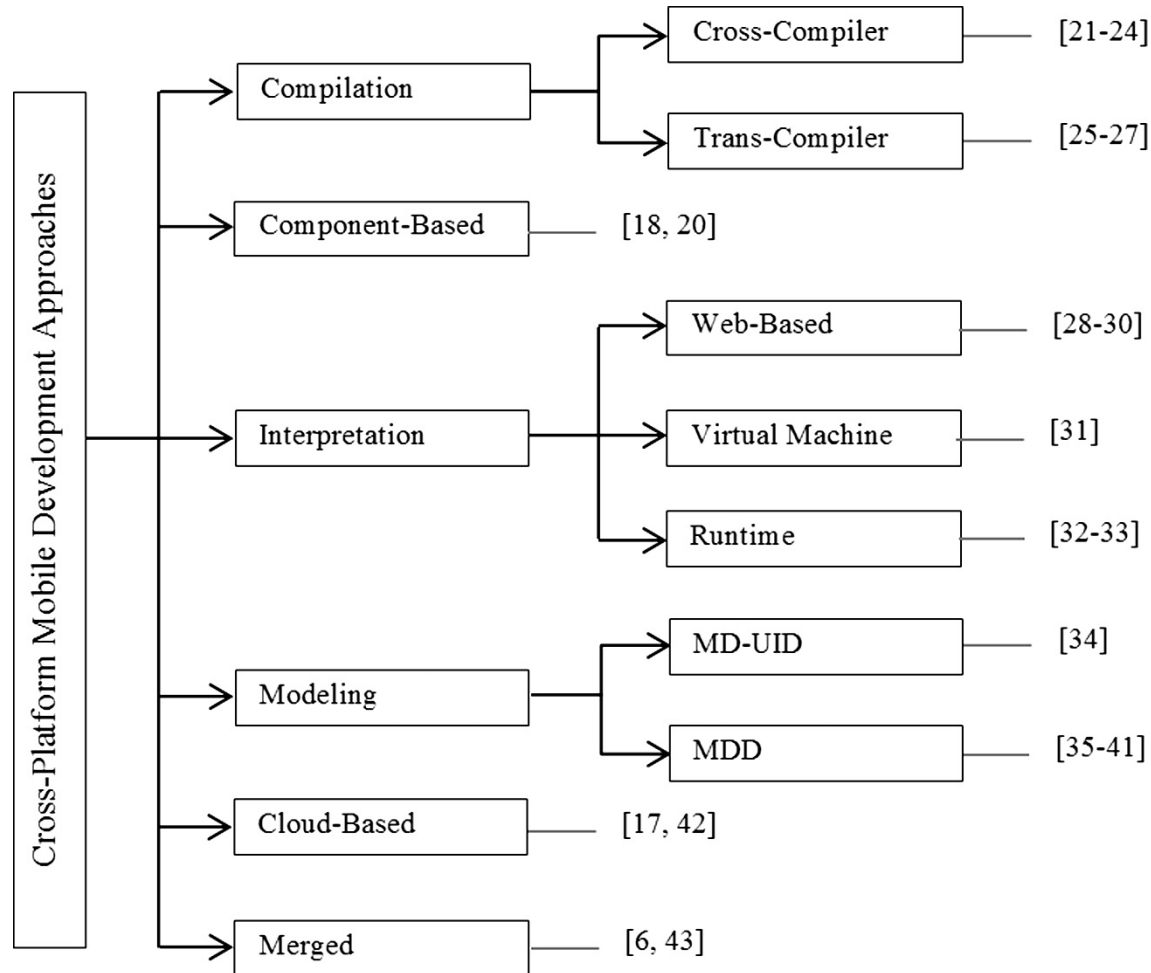
App development starts from different components that communicate together using interfaces

Each component has the same interface in every platform, but different implementations

Advantages:

- It simplifies the development, assuming that there are several *off-the-shelf* components available

Other classifications - 3



W. S. El-Kassas et al. *Taxonomy of Cross-Platform Mobile Applications Development Approaches*, Ain Shams Engineering J. 8(2), p. 163-190, 2017

With this approach the developer uses several abstract models to describe user interface and functions of the application

Models are then translated into native source code

With this approach, all the app computations are done in the *cloud*, and the application receives user interaction, sends them to the cloud, and shows the result of the elaboration.

Pros & Cons:

- Continuous network usage
- No need for specific hardware

How to choose the right framework?



*Still an open
research
problem...
But some steps
have been
taken!*



Cross-platform frameworks comparison

Idea: write one application using different frameworks, deploy on 2 target platforms, and compare results

4 different approaches were considered: *Web*, *Hybrid*, *Interpreted*, and *Cross Compiled*



Case study



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



jQuery & jQuery Mobile



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Desktop and mobile applications were developed as a single one

Hardware access depends on HTML5 support of the browser (therefore not controllable by the developer)

Low initial knowledge (Javascript, HTML, CSS)

Low development complexity

Animation support, but...

Complex animation performances rapidly decrease



An application developed with PhoneGap is a web application plus the WebKit rendering engine

Allows access to several device sensors (accelerometer, compass, etc.)

Good performances with simple apps, but poor performances with complex apps

Development languages: HTML, CSS, and Javascript



Phone**Gap**

PhoneGap suite



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Titanium allows maximizing reuse of pieces of code

Several APIs are available, especially for iOS and Android

Provides support for iOS (starting from version 5.0), Android (from 2.3.3), Window Phone, BlackBerry and Tizen

Apps with native *look and feel*

Development language: JavaScript



Allows using different development languages:

C, C++ and HTML+Javascript

Several APIs are available, but some of them are for obsolete operating systems versions

C++ development is a bit tricky

Native UI



Results



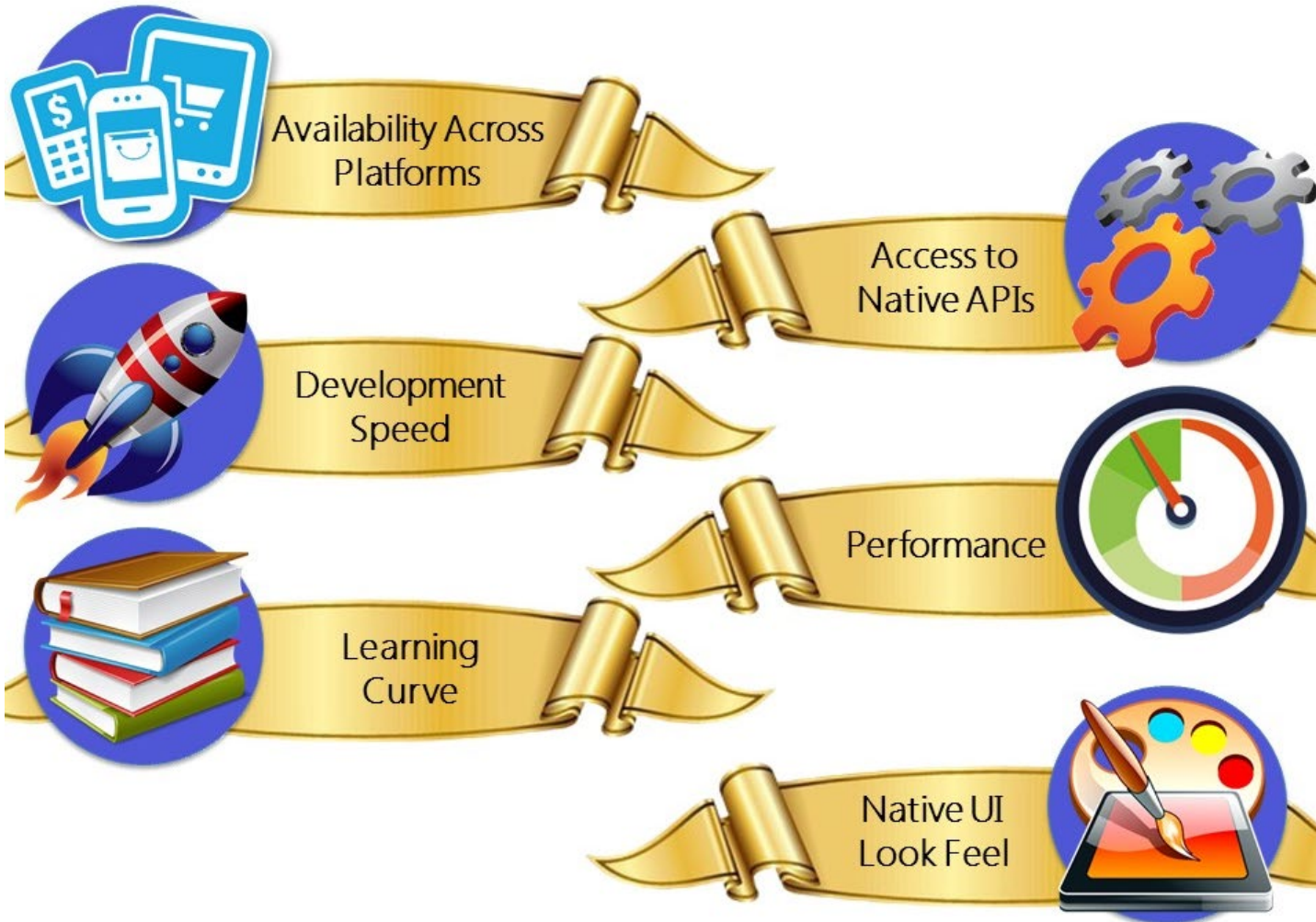
UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Licenses
APIs
Community
Tutorials
Complexity
IDE
Device support
Native UI
Learning curve

Selection criteria



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



<https://www.rishabhsoft.com/blog/cross-platform-app-tools-infographic>

Energy consumption analysis



UNIVERSITÀ
DEGLI STUDI
DI PADOVA

Energy consumption is a crucial element for application success: apps that drain the battery are rapidly uninstalled by users

We considered the energy consumption of apps that acquire data from different sensors:

- Accelerometer
- Compass
- Microphone
- GPS
- Camera



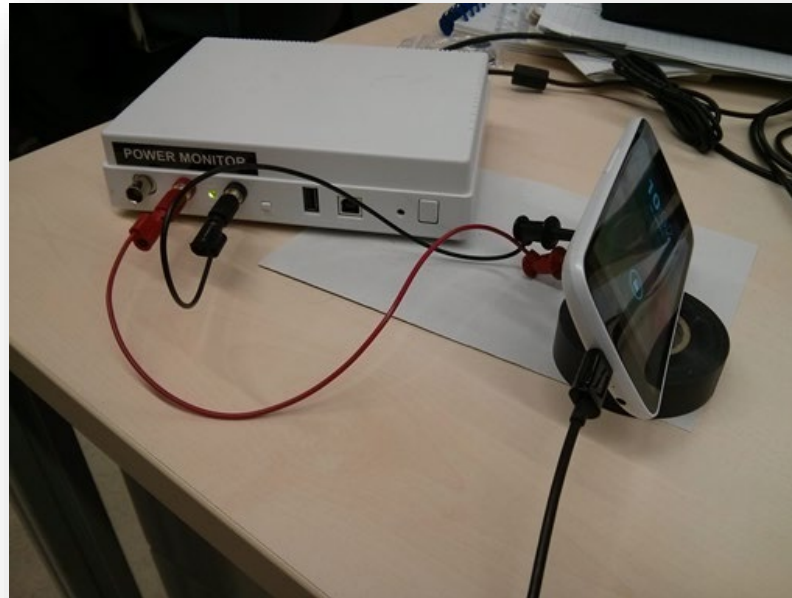
And compared these results between native apps and the ones developed using cross-platform frameworks

How can we avoid interferences from internal (OS events) or external (user interaction) factors?

Several authors measured energy consumption of mobile applications

- Thompson et al. proposed a model-driven approach (SPOT, *System Power Optimization Tool*) for energy consumption estimation before application development
- *AppScope* (Yoon et al.) is an Android application that estimates energy consumption of each hardware component

Monsoon Power Monitor

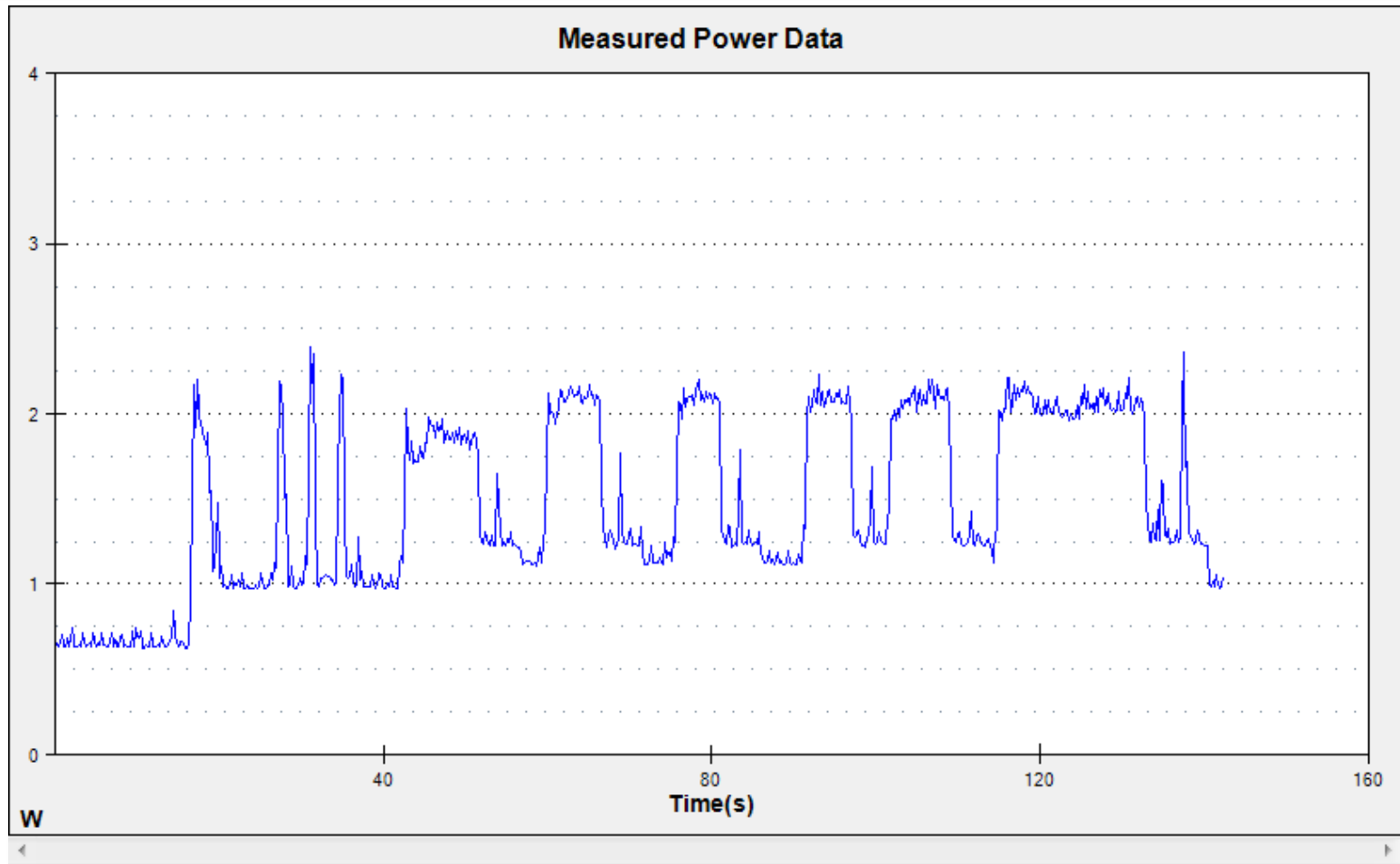


Provided data: energy consumption, average current and consumption, estimated battery duration, etc.

Monsoon PowerTool



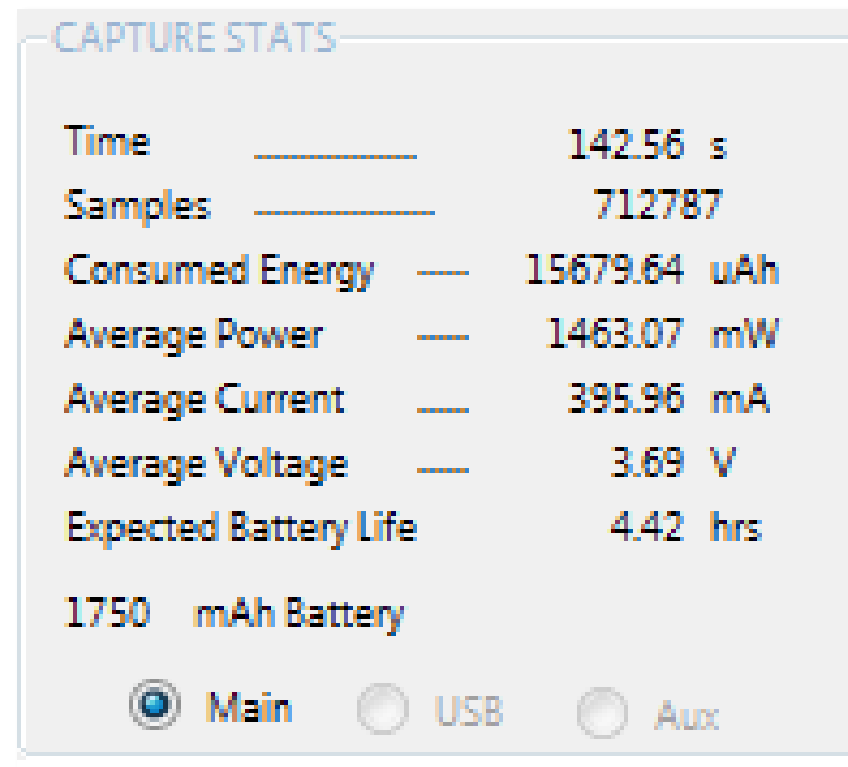
UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Monsoon PowerTool



UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Goal: energy consumption comparison of different hardware components during data collection, considering different platforms and different frameworks

Applications considered:

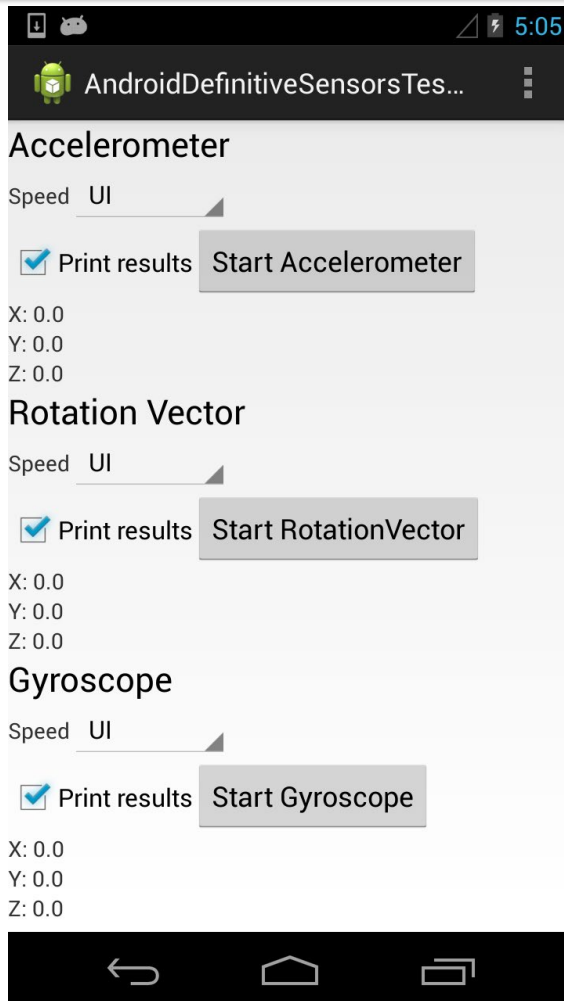
- Native Android application
- Web application
- Hybrid application developed with PhoneGap
- Application developed with Titanium
- Application developed with MoSync using C++
- Application developed with MoSync using Javascript

Sensors depend on the API available with each framework

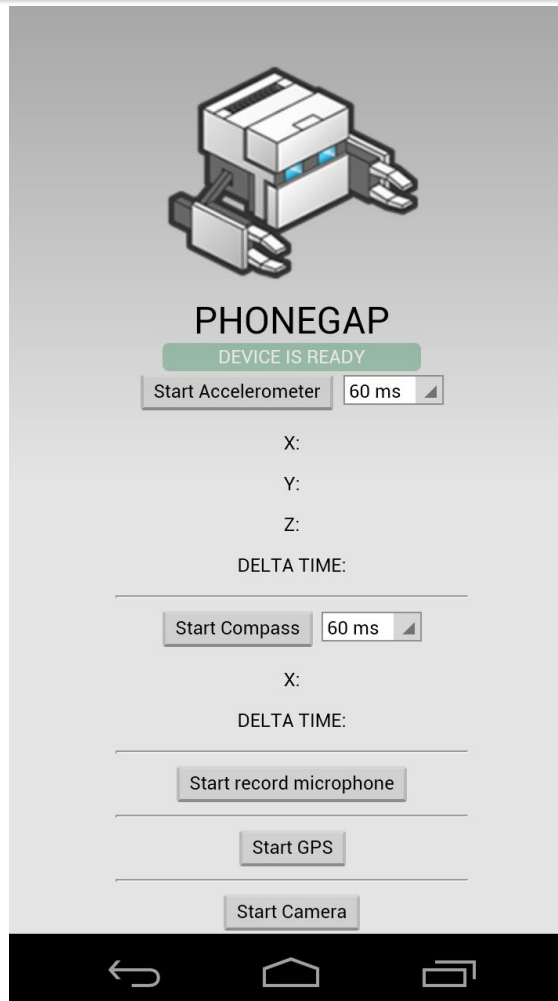
Applications



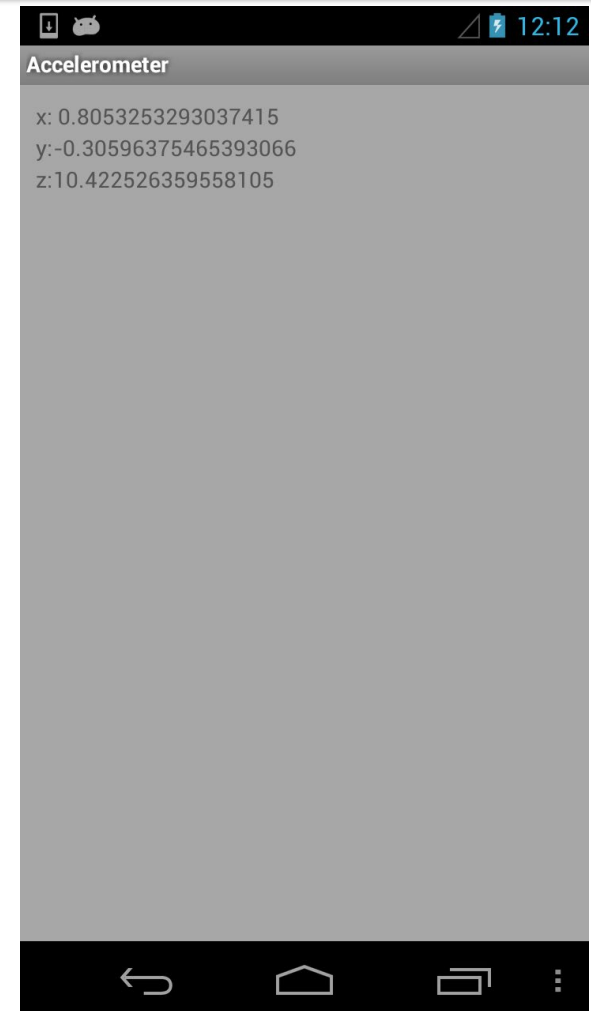
UNIVERSITÀ
DEGLI STUDI
DI PADOVA



Native Application



Phonegap Application



Titanium Application

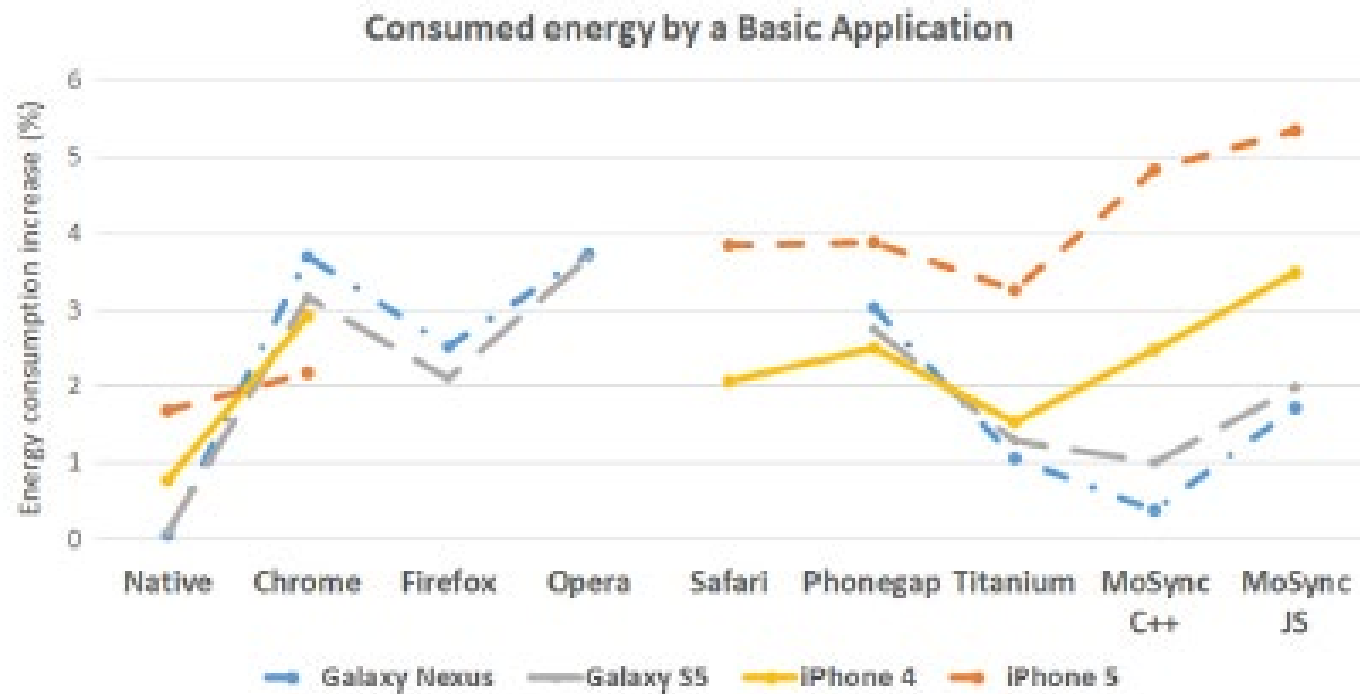
Elementary applications that collect data from different sensors at a given frequency, showing data with a simple interface

It is necessary to define a *base level (or 0 level)* of energy consumption: device in stand by, airplane mode, black screen with minimum white elements

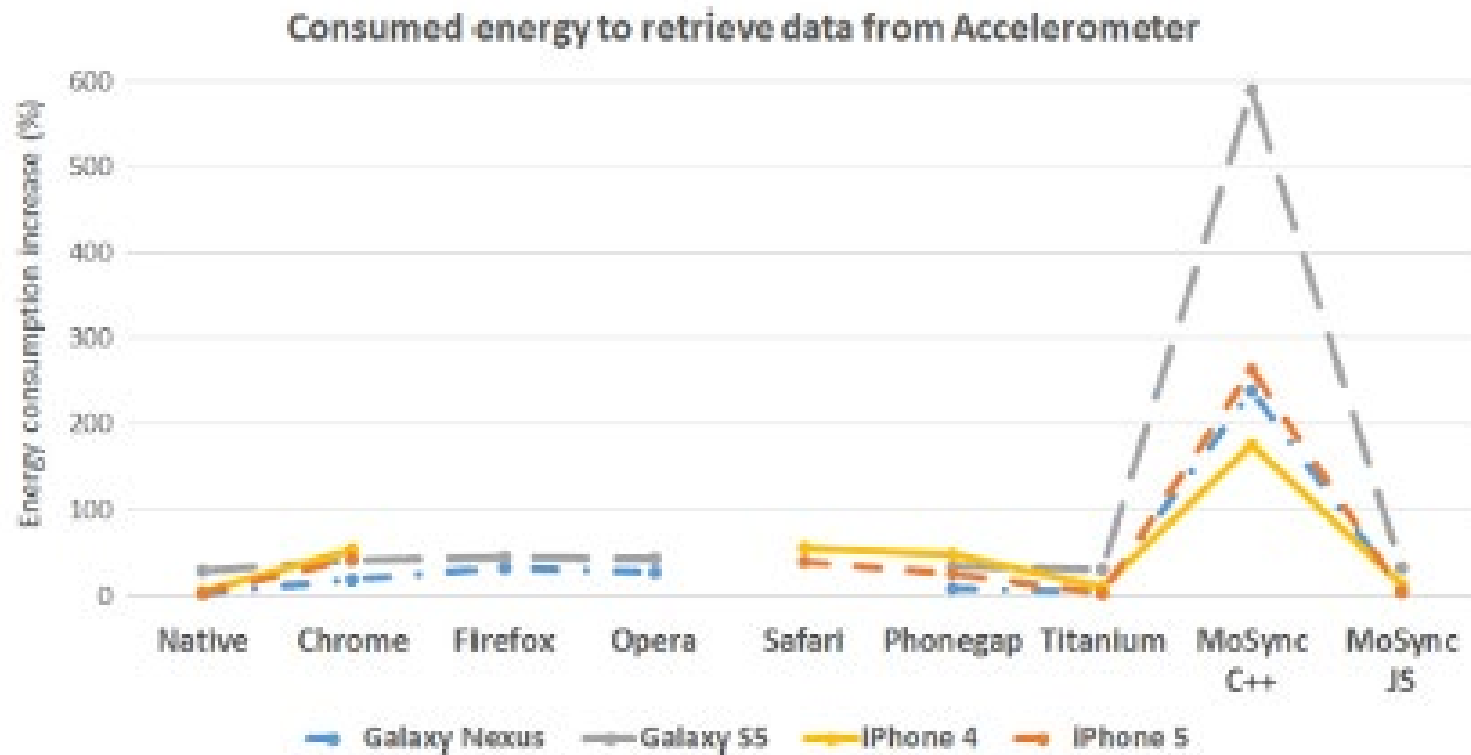
The base level depends on the device and its battery. Considered smartphones:

- An iPhone 4 and an iPhone 5
- A Samsung Galaxy Nexus and a Samsung Galaxy S5

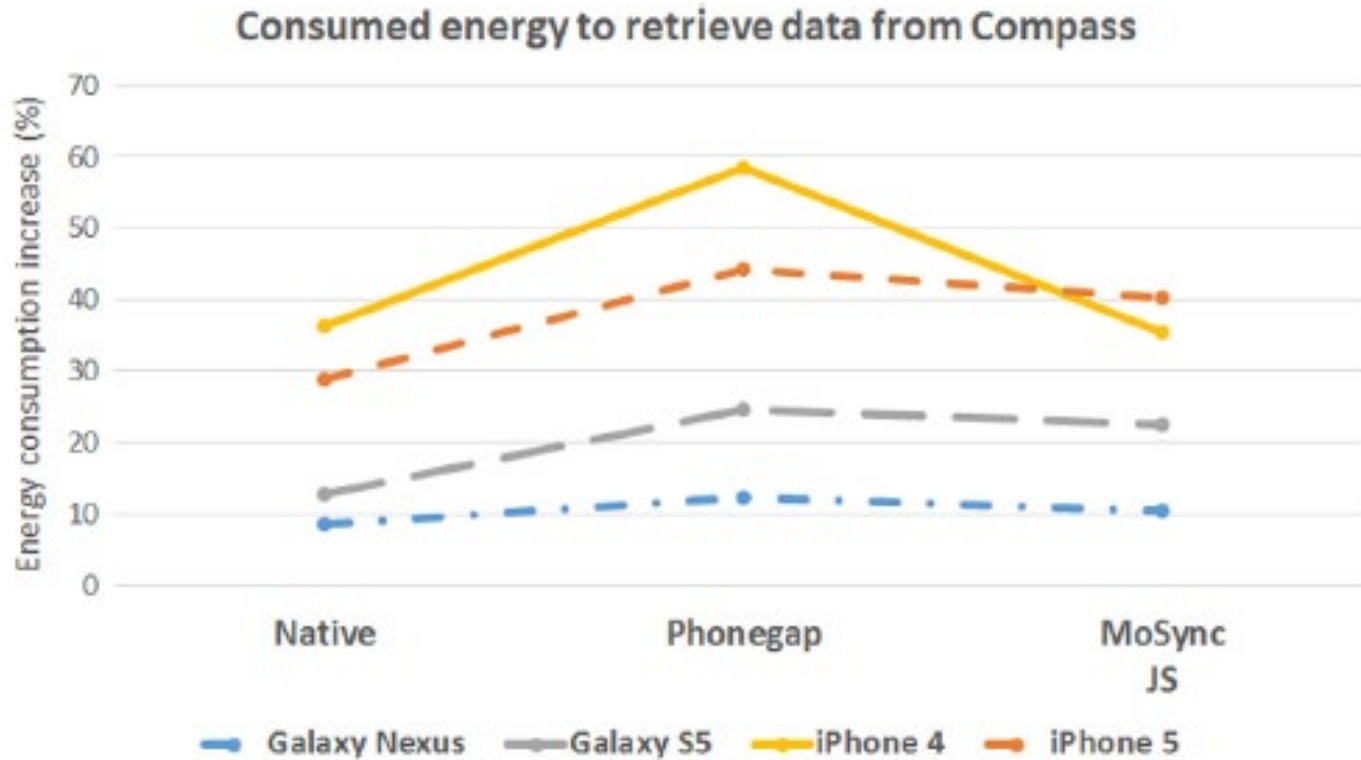
Base energy consumption



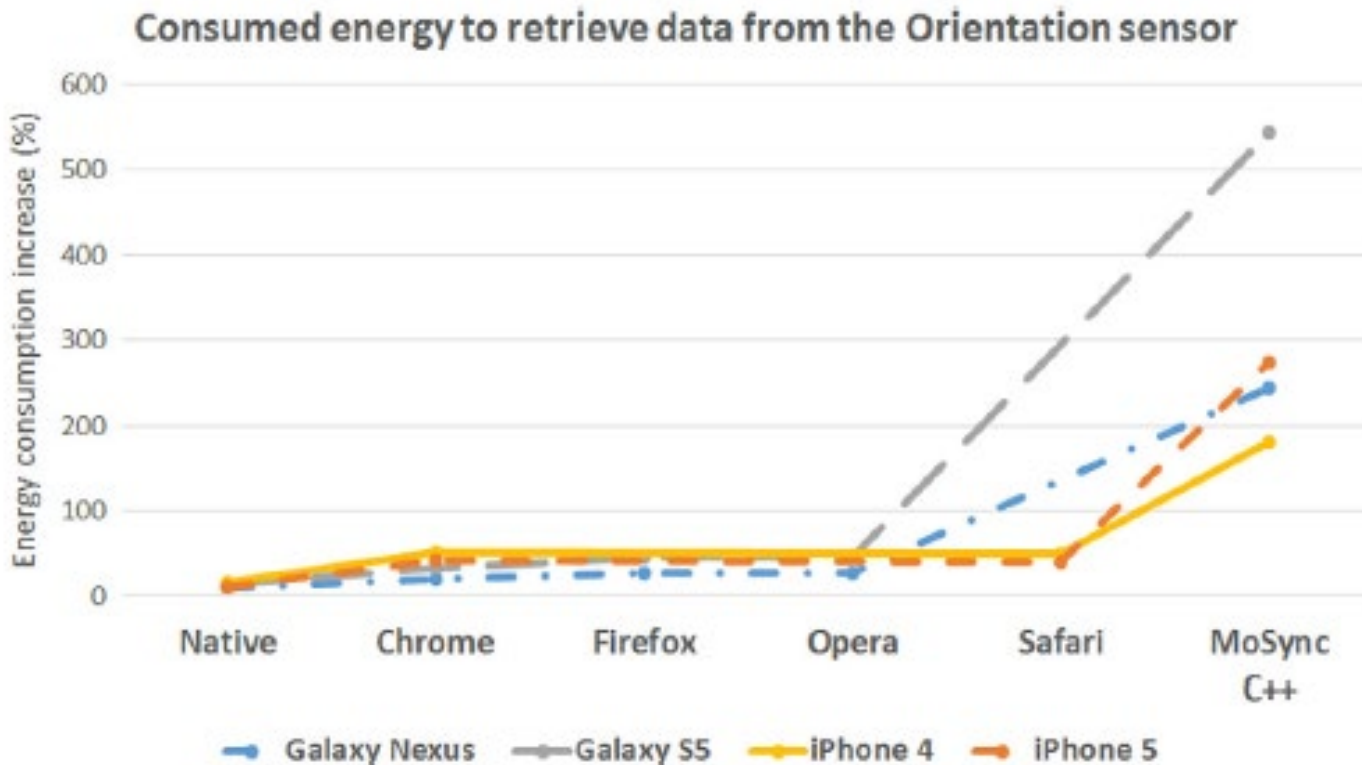
Results: accelerometer



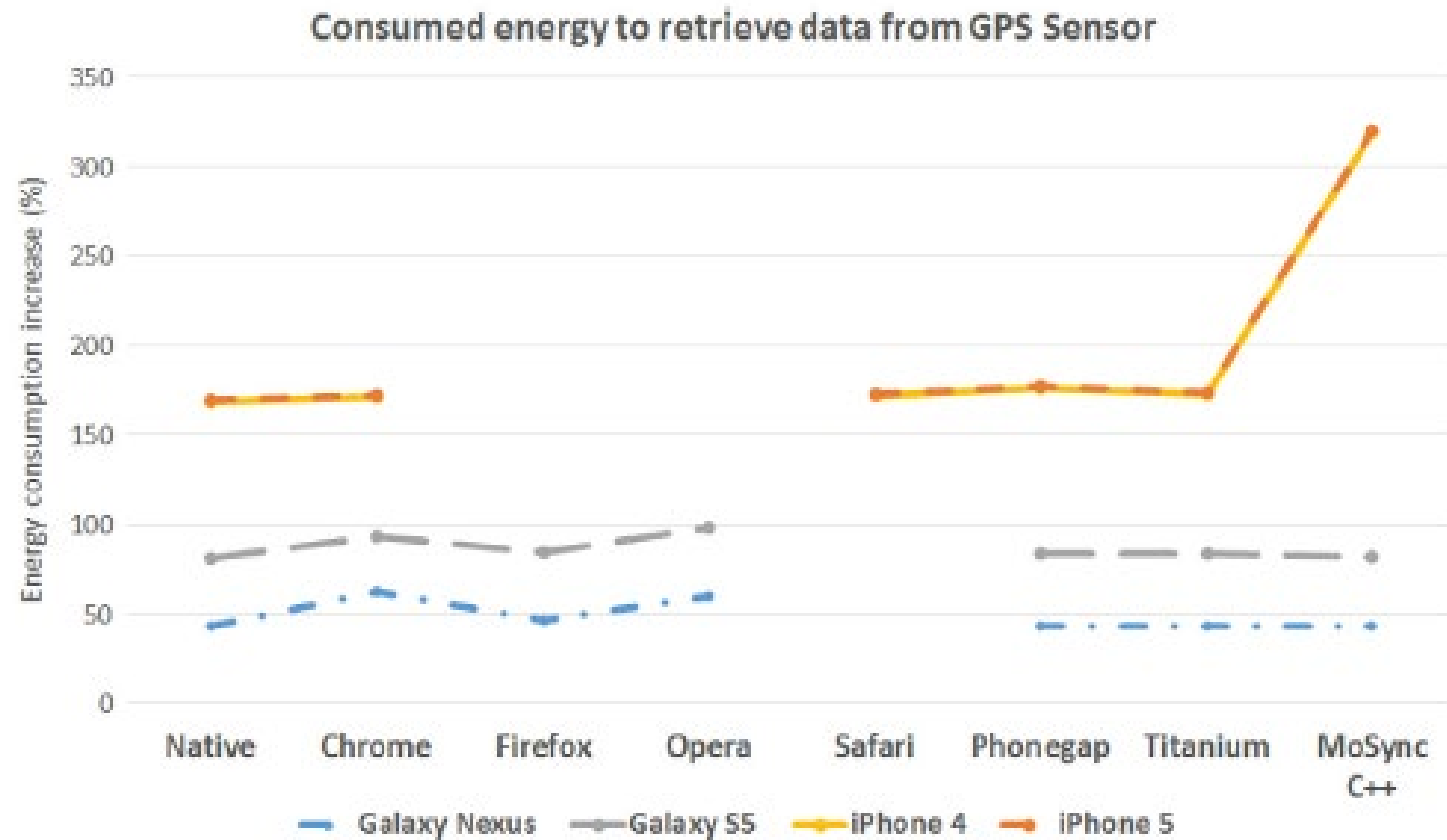
Results: compass



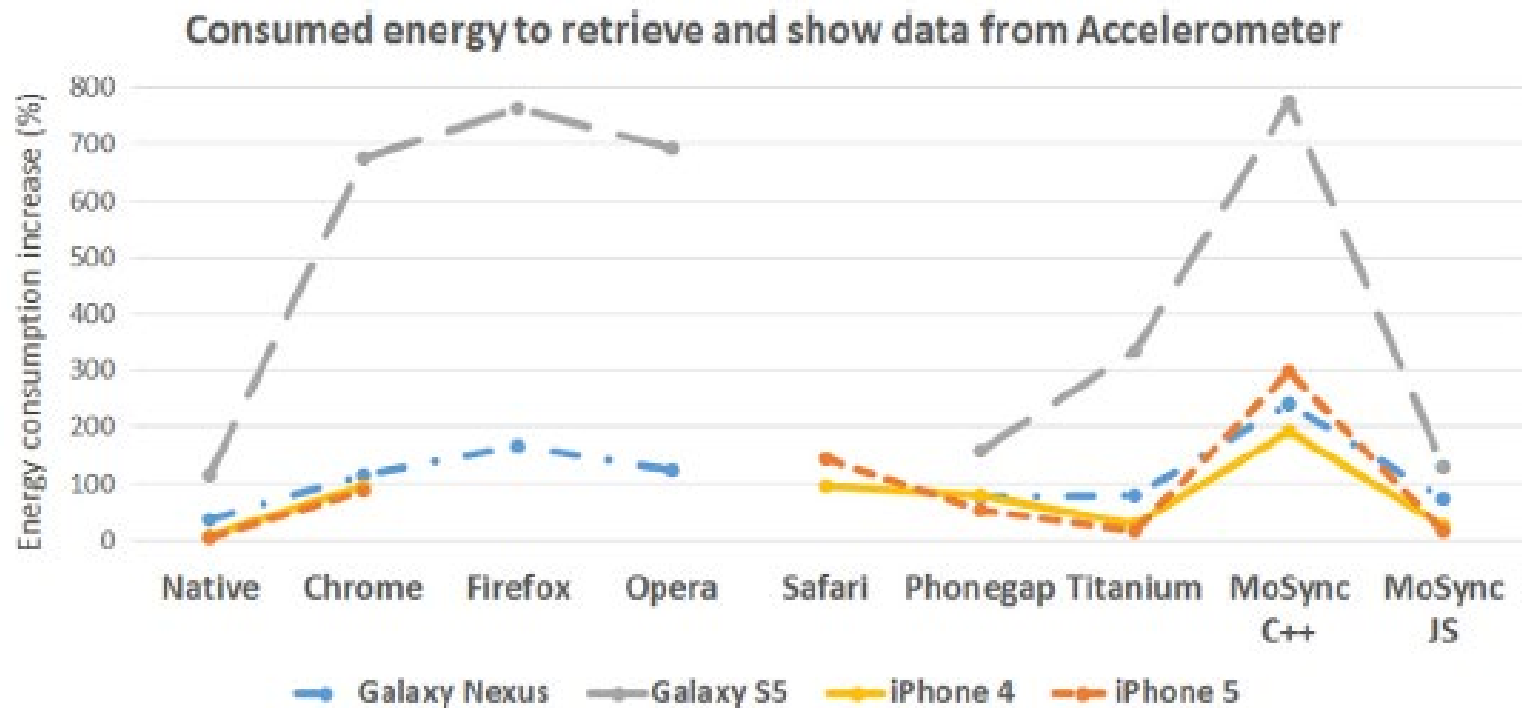
Results: orientation sensor



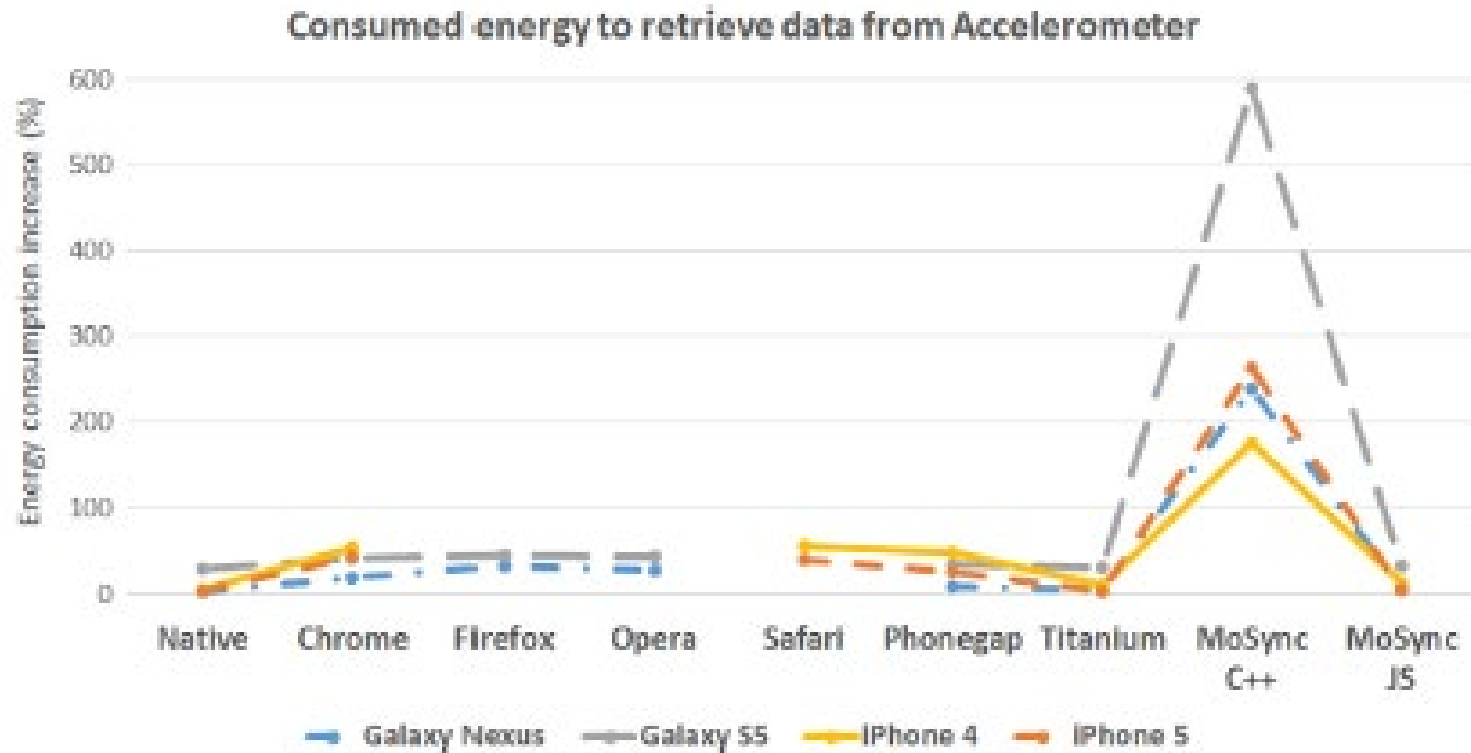
Results: GPS



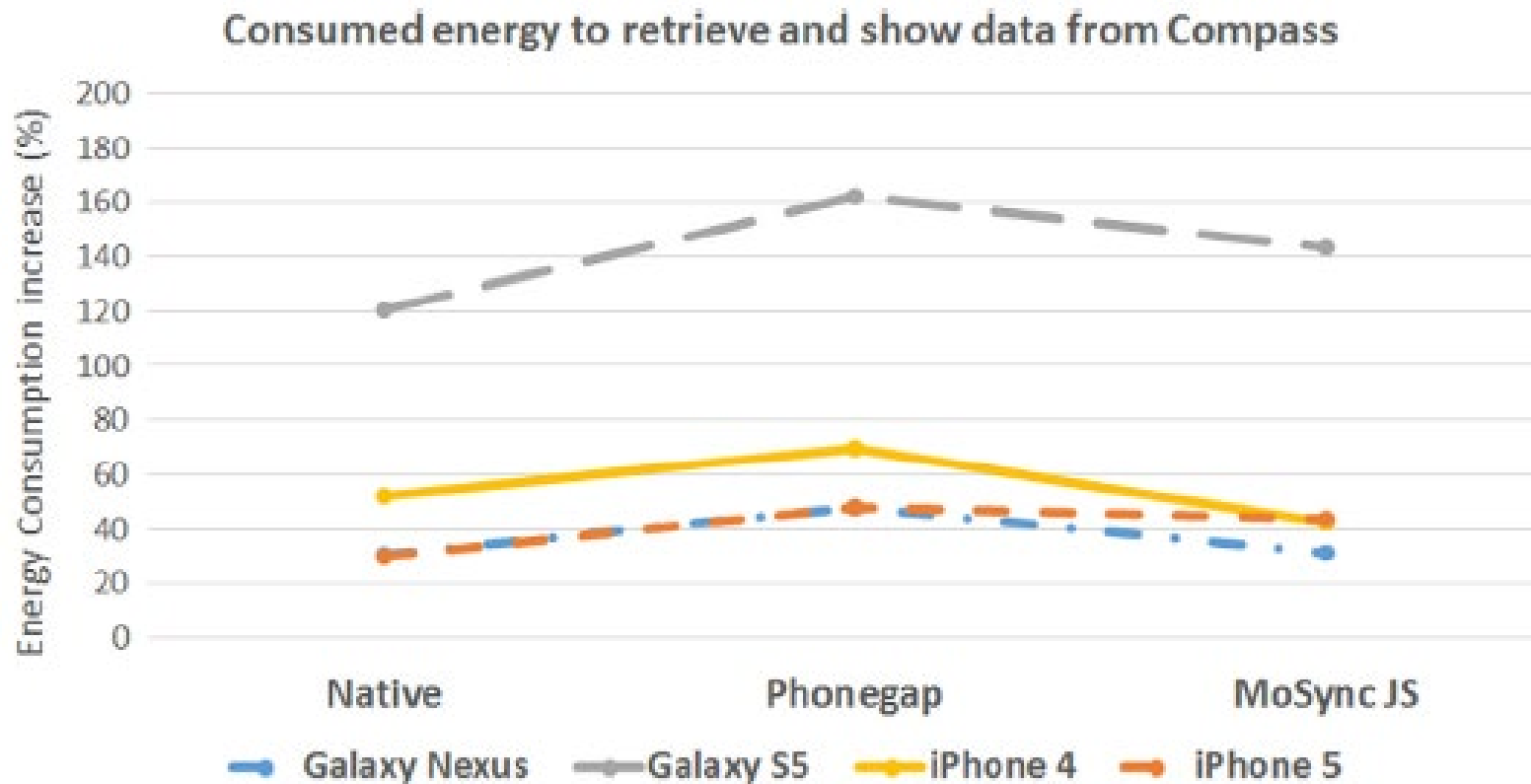
Data visualization: accelerometer



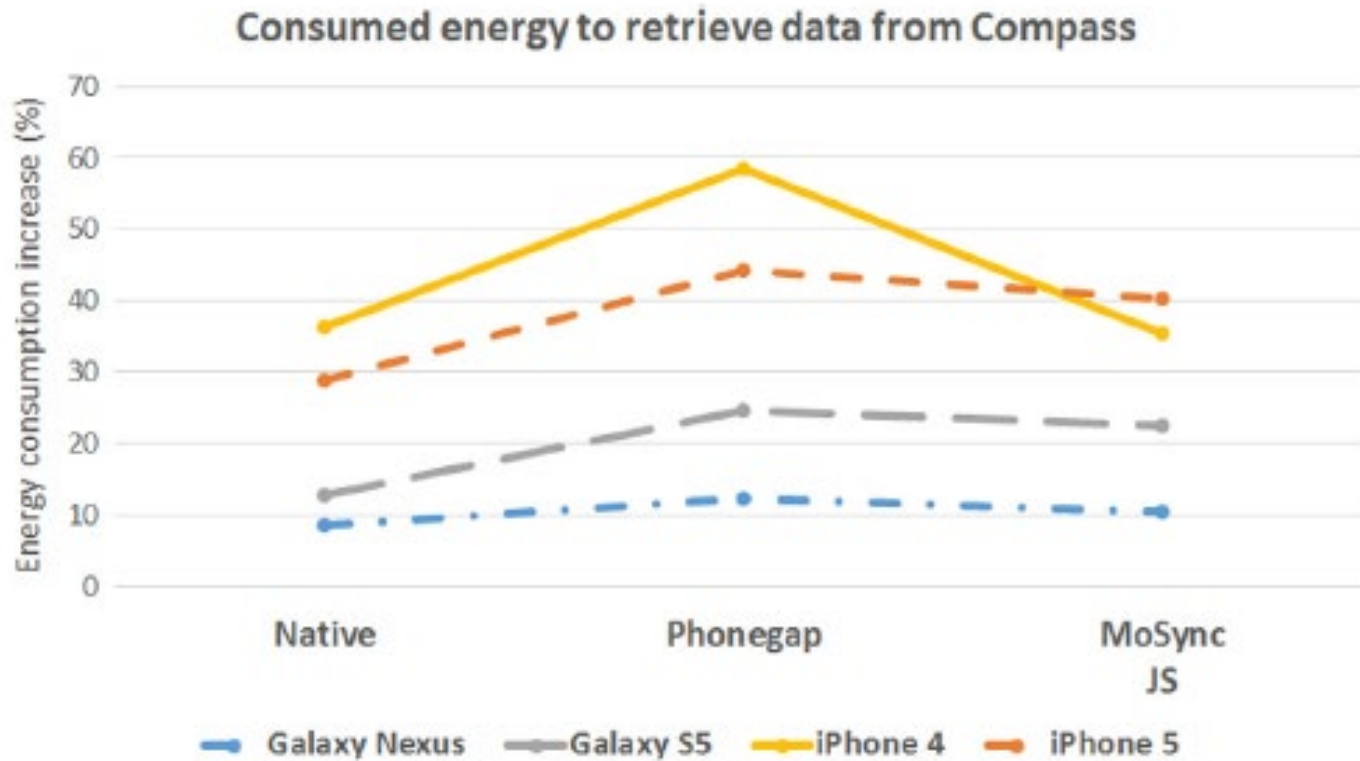
Results: accelerometer



Data visualization: compass



Results: Compass



- Cross-platform frameworks determine higher energy consumption, even if the framework generates native code
- The most expensive task is the interface update
- Data acquisition frequency strongly influence energy consumption
- *Not really cross-platform!*
 - Frameworks have different energy consumption depending on the operating system where they are running

- Results show that cross-platform frameworks consume more energy, hence determining lower performances and user acceptance
- Depending on the type of application, native development should be preferred:
 - Lower energy consumption
 - More APIs are available
- The results suggest that
 - Framework choice is critical
 - For a complex application, efficient frameworks are still missing

- Framework choice is crucial because it can influence user experience
 - Providing a lousy application is worse than not providing an application at all
 - Results show that, *at the moment*, Titanium seems to be the framework with better consumption

- An efficient cross-platform framework must provide an extremely efficient user interface rendering
- Improvements of efficiency of rendering engines will provide improvements even for those frameworks that work with the web approach
- The cross-compiled approach seems to be the most promising, but it is not easy to implement
 - The development of these frameworks strongly depends on the development of a compiler able to produce an efficient code for the application
 - Development should be focused on the optimization of events handling and management

Cross-platform classification

1. R. Raj, S. B. Tolety. *A study on approaches to build cross-platform mobile applications and criteria to select appropriate approach*. Annual IEEE India Conference, INDICON '12, 2012, p. 625- 629.
2. W. S. El-Kassas, B. A. Abdullah, A. H. Yousef, A. M. Wahba. *Taxonomy of Cross-Platform Mobile Applications Development Approaches*. Ain Shams Engineering Journal, Volume 8, Issue 2, 2017, p. 163-190.

Results about cross-platform frameworks analysis were published in two different articles:

1. M. Ciman, O. Gaggi. *An empirical analysis of energy consumption of cross-platform frameworks for mobile development*, PMC vo. 39, p. 214-230, 2017
2. M. Ciman, O. Gaggi, N. Gonzo. *Cross-Platform Mobile Development: A Study on Apps with Animations*. Proceedings of the ACM Symposium on Applied Computing (SAC2014), pages 757-759, Gyeongju, South Korea, May 2014.