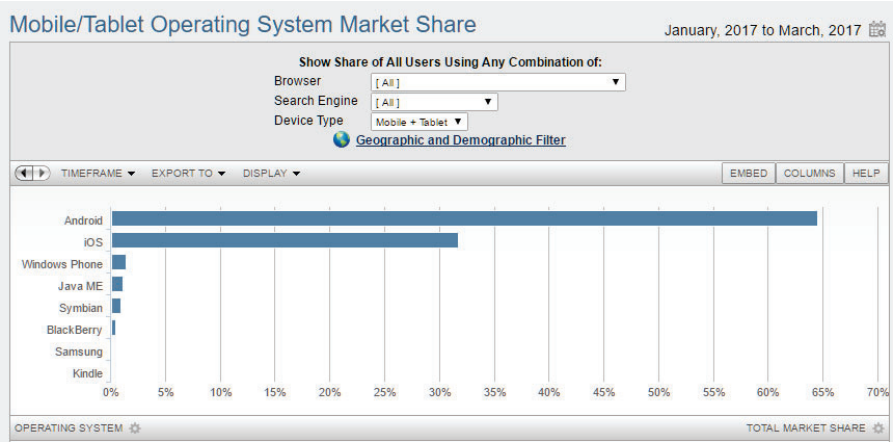
Mobile Programming and Multimedia

Cross-Platform Frameworks

Prof. Ombretta Gaggi
University of Padua

… but Android diffusion is higher than iOS!



https://www.netmarketshare.com/

Scenario

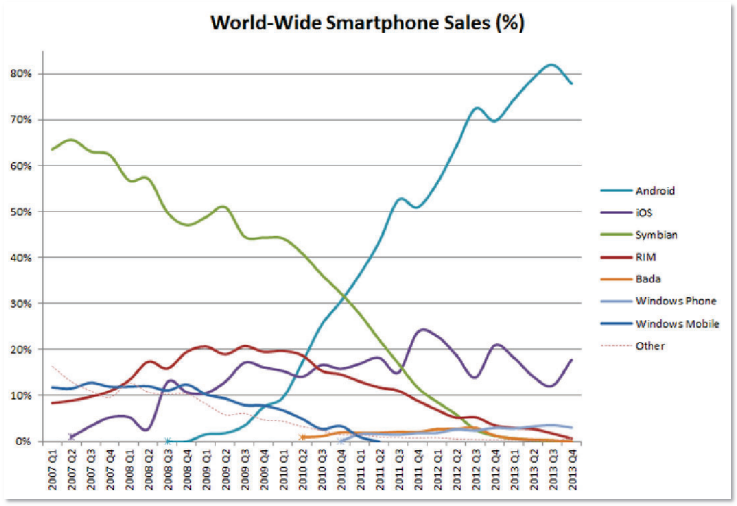
I have an exciting idea for a new app

I want to develop this fantastic app

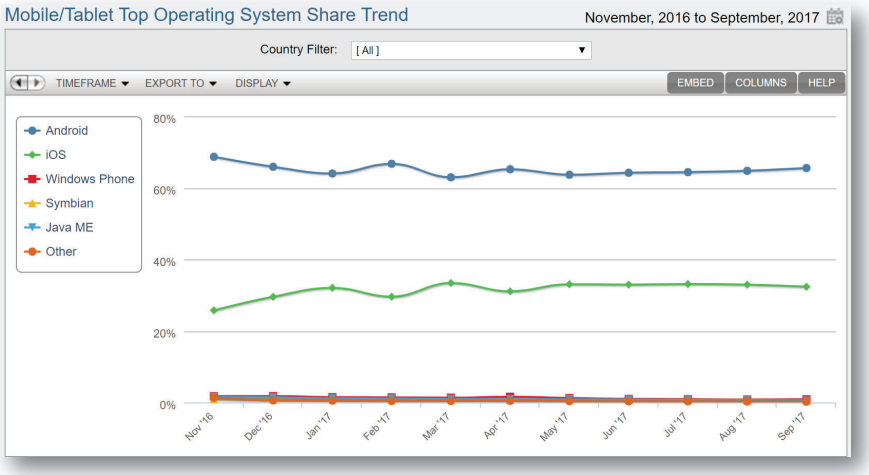
I have an iPhone so…

… I develop it in iOS

Solution: go back to 2010 or before ;-)

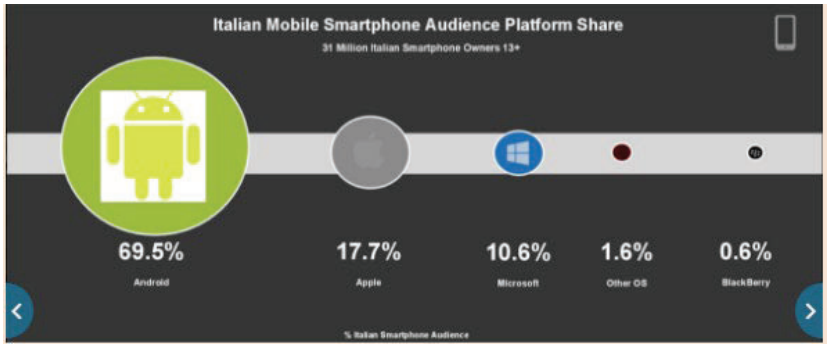


Market fragmentation



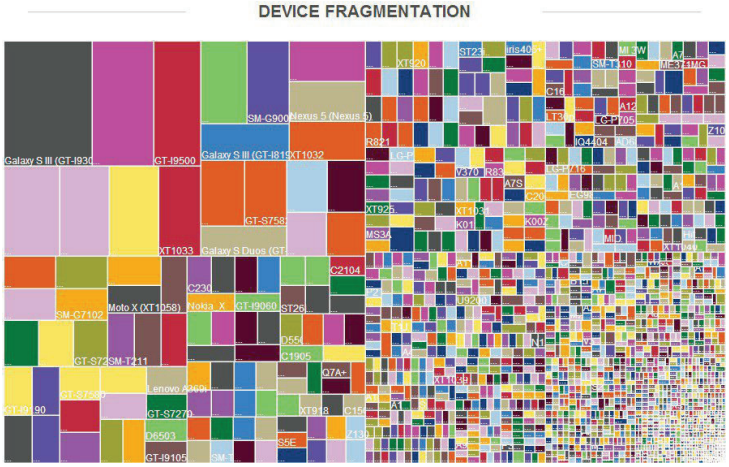
https://www.netmarketshare.com/

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

Università degli Studi di Padova

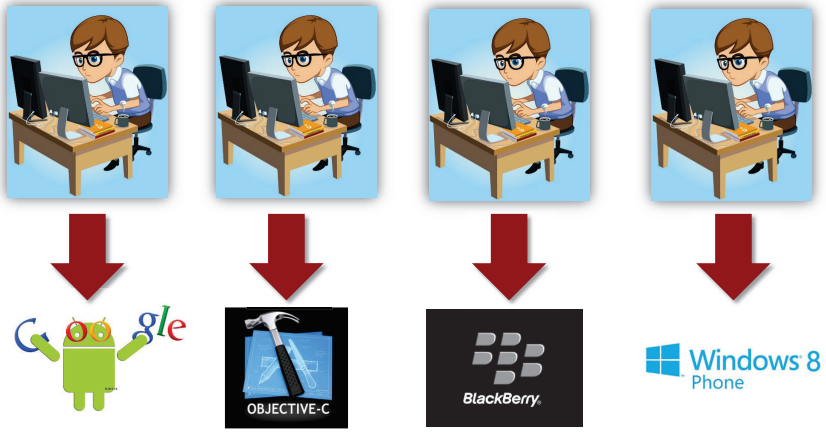
Different OS => different languages

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

How many?

Università degli Studi di Padova

One for each operating system?



Highly expensive!

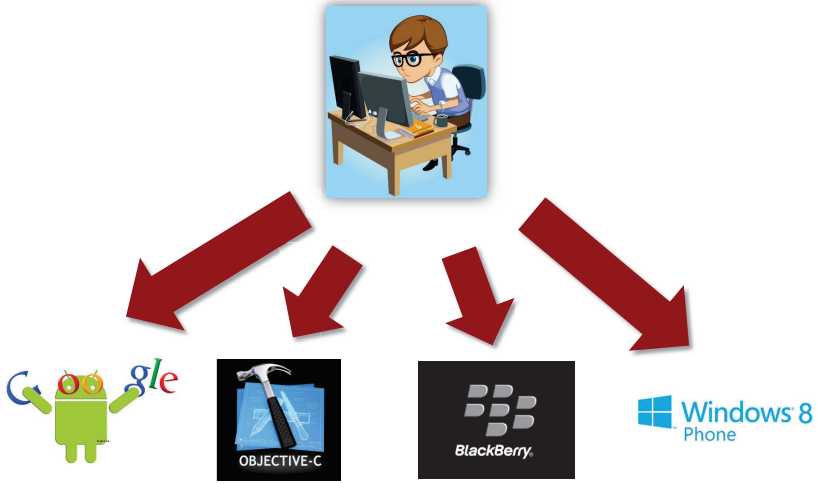
Università degli Studi di Padova

Which are the variables?

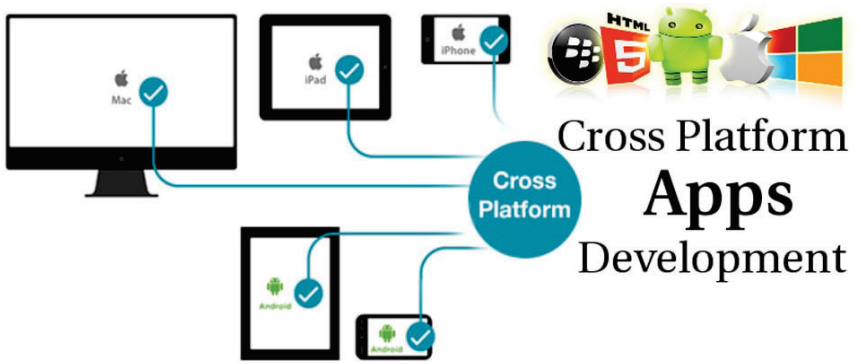
Università degli Studi di Padova

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

Goal



Solution



A new approach

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

«Write once, distribute everywhere»

Main features

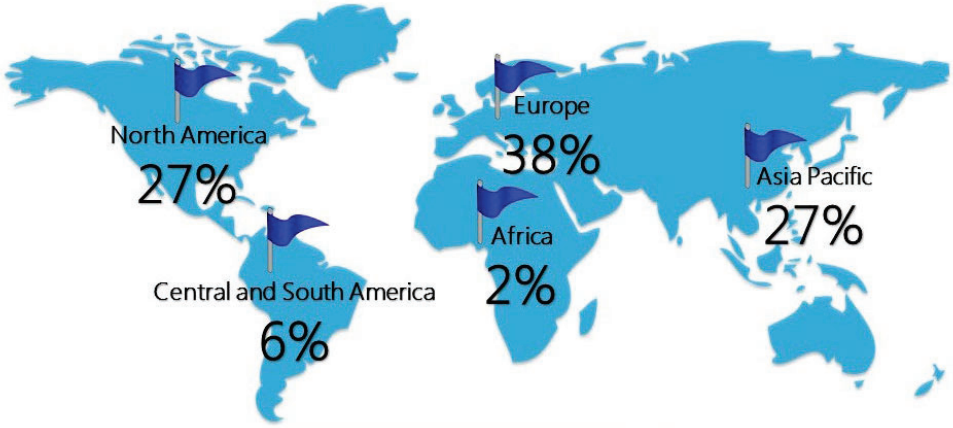
- Application developed on time, using only one programming language (or a set of languages)
- The chosen framework allows the distribution of the application in several applications stores (so, there are several applications deployed)
- The frameworks usually provide support for native API

Really the solution?

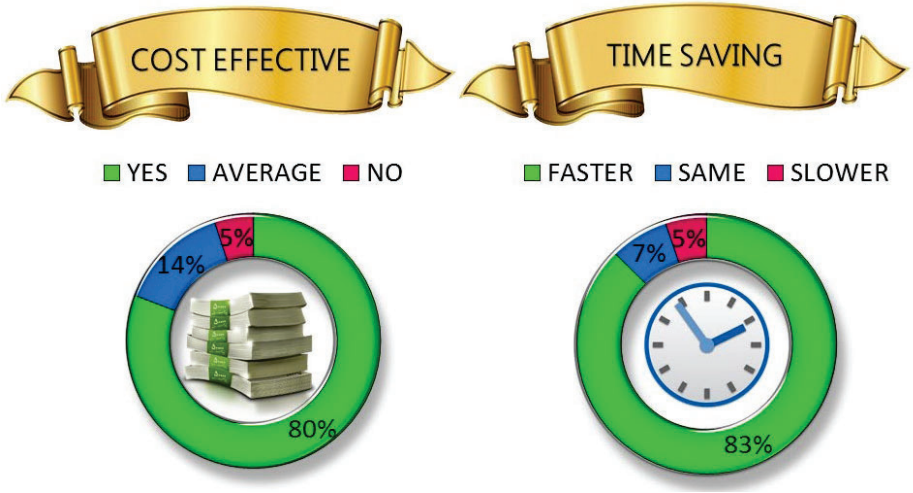
Market

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

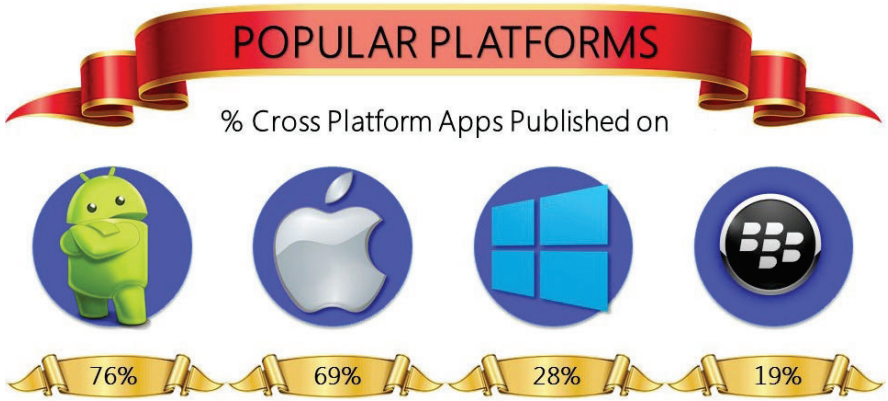
Geographical distribution



Benefits



Platforms

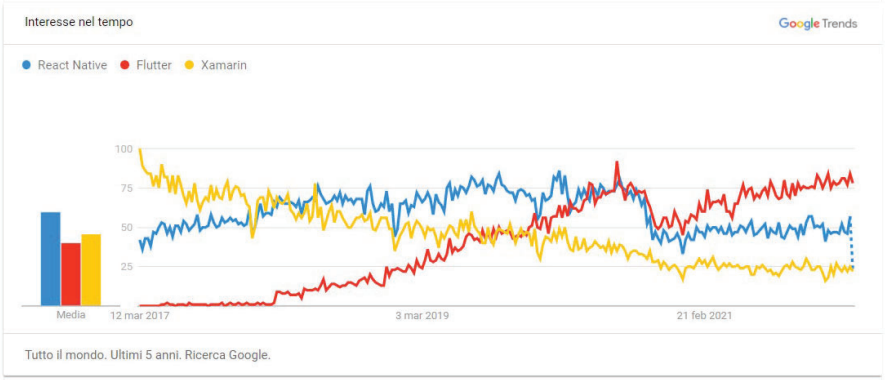


Most popular

titanium
PhoneGap
Qt
ADOBE AIR
Sencha
rhomobile
Xamarin
Corona

Pay attention to the stats!

Google statistics



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	

Native: Pros & cons

- 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

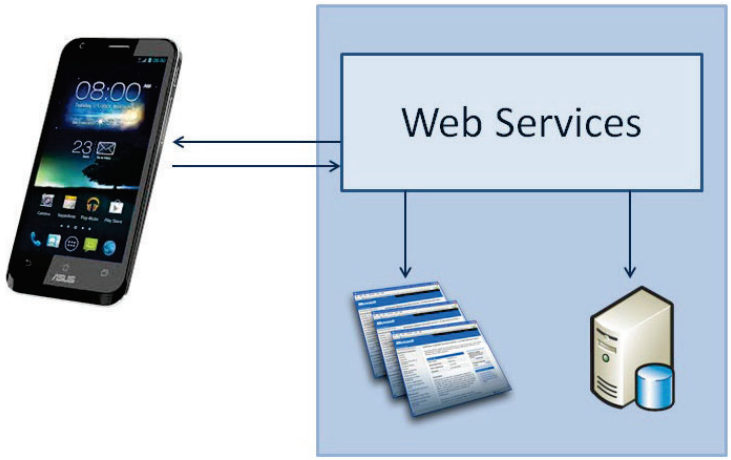
Frameworks' classification is still an open problem

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

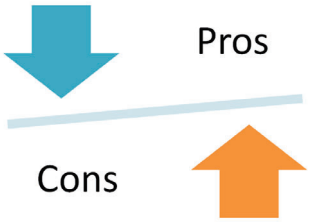
Web Approach



Web Approach: Pros & Cons

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

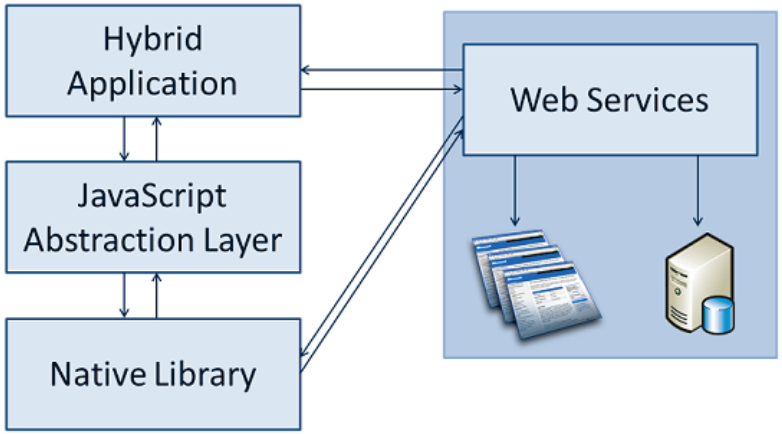
Examples

Sencha Touch

React

jQuery mobile

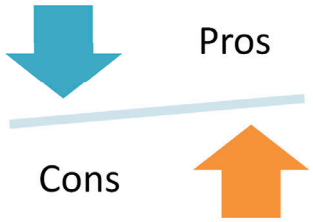
Hybrid Approach



Hybrid Approach: Pros & Cons

Pros:

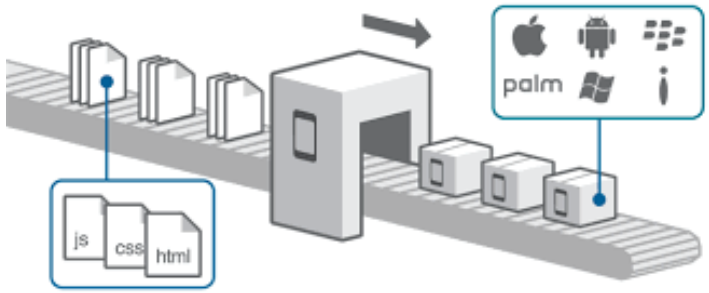
+ Store publishing available
+ Reusable UI
+ Usage of device components



Cons:

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

Example: PhoneGap



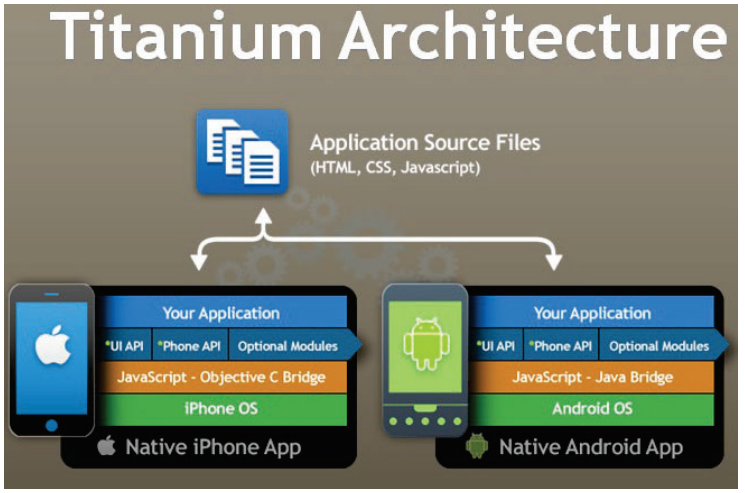
HelloWorld – body

```
<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

- 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

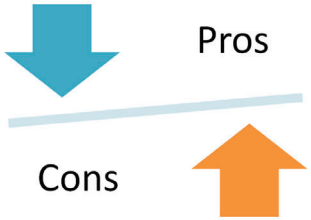
Interpreted Approach



Interpreted Approach: Pros & Cons

Pros:

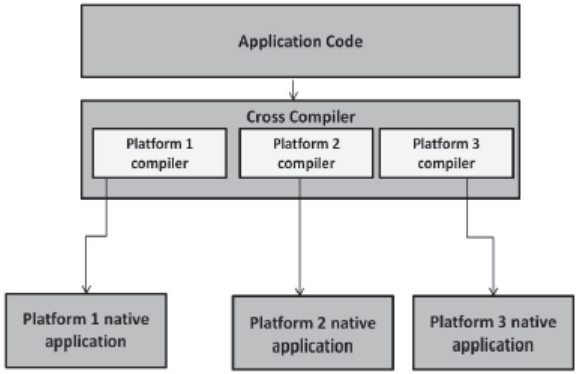
+ Native Look and Feel
+ Store publishing
+ APIs for device components



Cons:

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

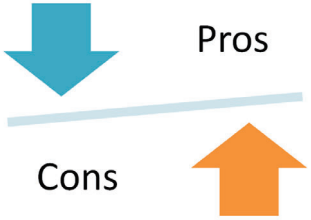
Cross-compiled Approach



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

Approach	Programming language	Supported platforms	Pros	Cons	Example
Web	HTML, CSS, Javascript	Android, iOS, Windows, BlackBerry[a]	– Easy to update – No installation – Same UI over different devices	– No access to application store – Network delays – Expensive testing – No native UI	jQuery mobile, Sencha Touch
Hybrid	HTML, CSS, Javascript	Android, iOS, Windows, Blackberry,	– Access to application store[b] – Support to most smartphone hardware	– No native look and feel	PhoneGap
Interpreted	Javascript	Android, iOS, Blackberry	– Access to application store – Native look and feel	– Platform branching – Interpretation step	Titanium
Cross-Compiled	C#, C++, Javascript	Android, iOS, Symbian	– Native UI – Real native application	– UI non reusable – High code conversion complexity for complex applications	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

Other classifications - 1

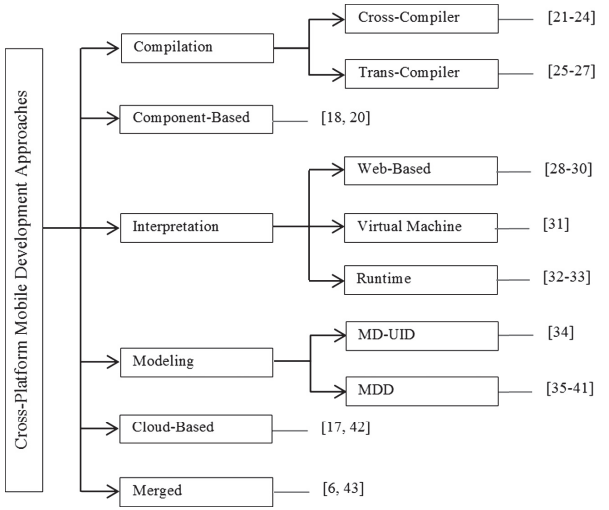
There are several methods for frameworks classification, some of them based on the development approach, others based on the result

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

Component based approach

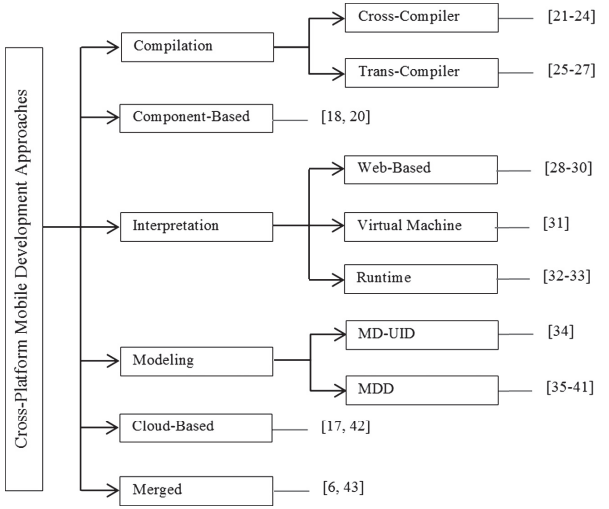
App development starts from different components that communicate together using interfaces

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

Modeling approach

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

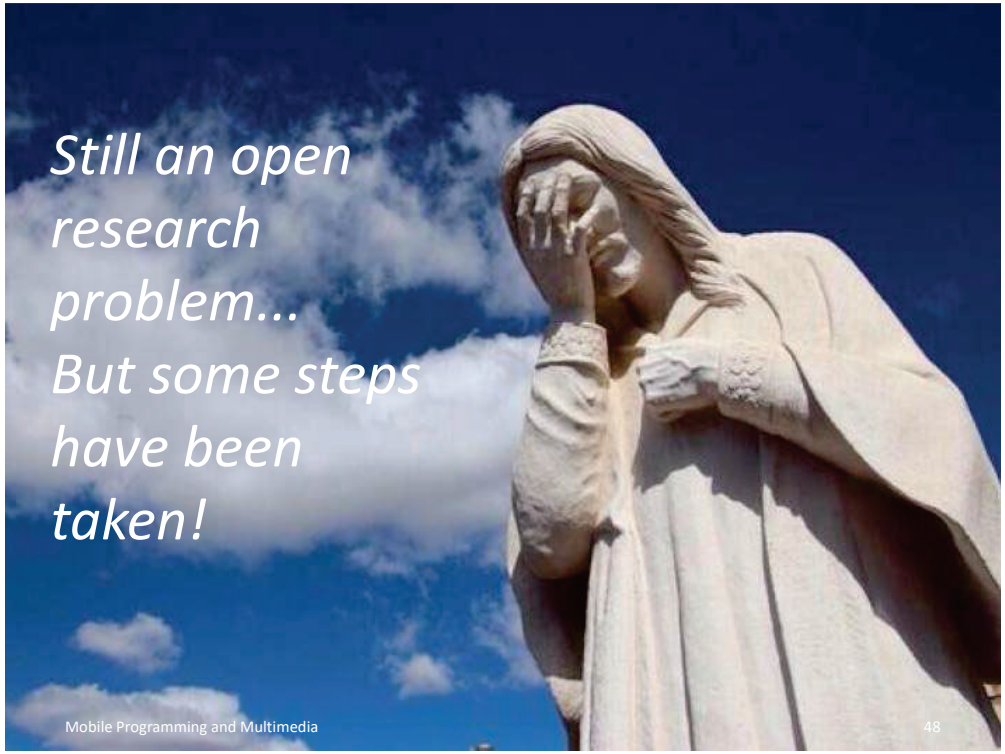
Cloud-based approach

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?

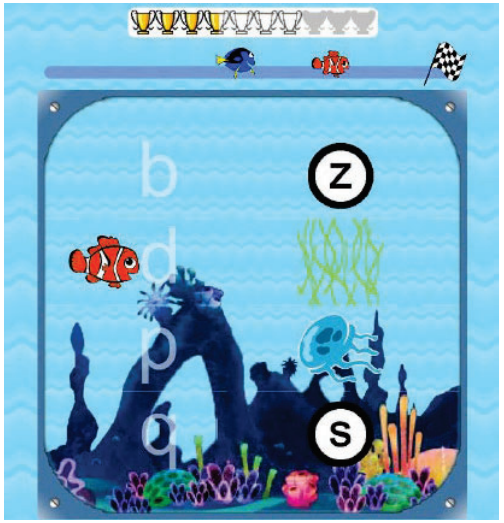


Cross-platform frameworks comparison

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



jQuery & jQuery Mobile

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

PhoneGap

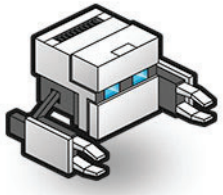
An application developed with PhoneGap is a web application plus the WebKit rendering engine

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

PhoneGap suite



Titanium

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

titanium™

MoSync

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

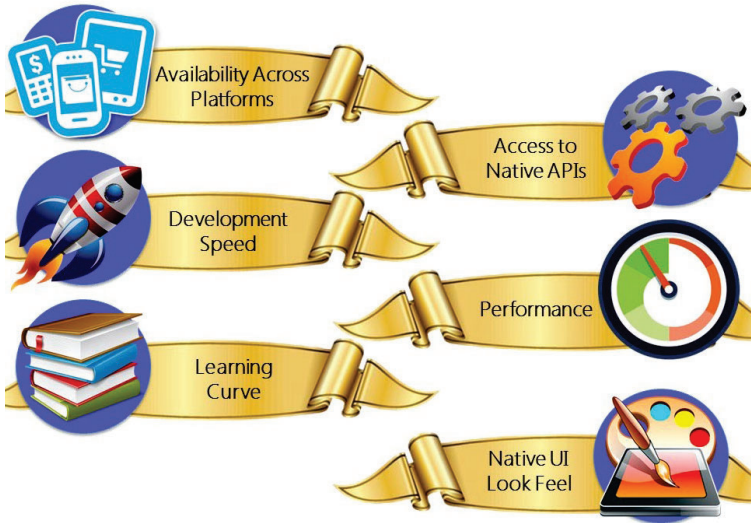
Native UI

MoSync

Results

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

Selection criteria



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

Energy consumption analysis

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?

Some solutions

Several authors measured energy consumption of mobile applications

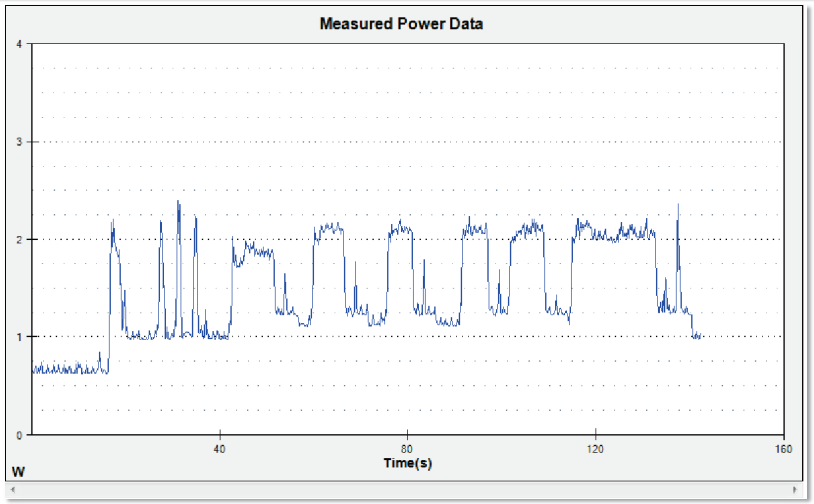
- Thompson et al. proposed a model-driven approach (SPOT, *System Power Optimization Tool*) for energy consumption estimation before application development
- *AppScope* (Yoon et al.) is an Android application that estimates energy consumption of each hardware component

Measurement system

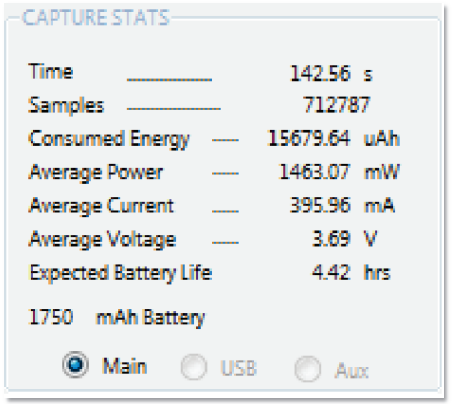
Monsoon Power Monitor

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

Monsoon PowerTool



Monsoon PowerTool



Analysis

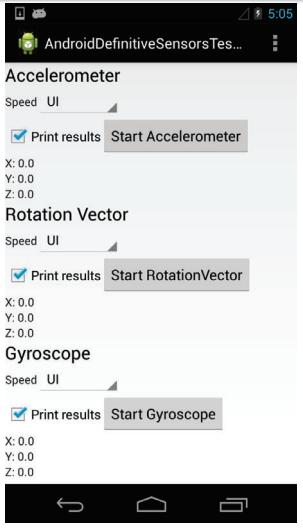
Goal: energy consumption comparison of different hardware components during data collection, considering different platforms and different frameworks

Applications considered:

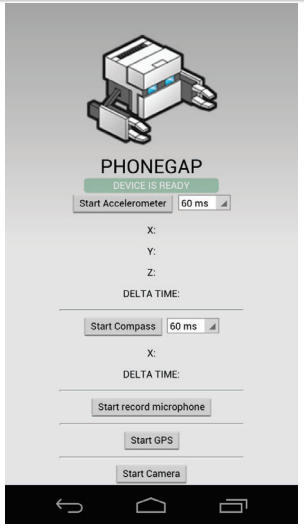
- Native Android application
- Web application
- Hybrid application developed with PhoneGap
- Application developed with Titanium
- Application developed with MoSync using C++
- Application developed with MoSync using Javascript

Sensors depend on the API available with each framework

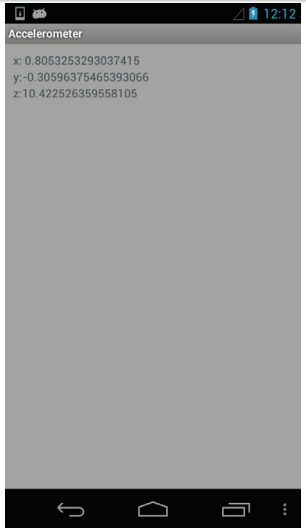
Applications



Native Application



Phonegap Application



Titanium Application

Developed applications

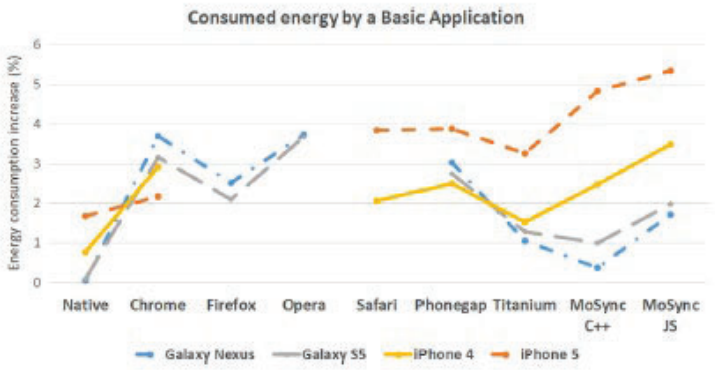
Elementary applications that collect data from different sensors at a given frequency, showing data with a simple interface

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

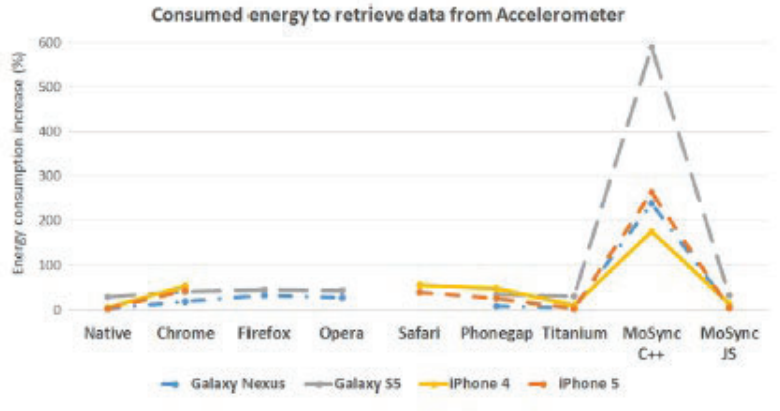
The base level depends on the device and its battery. Considered smartphones:

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

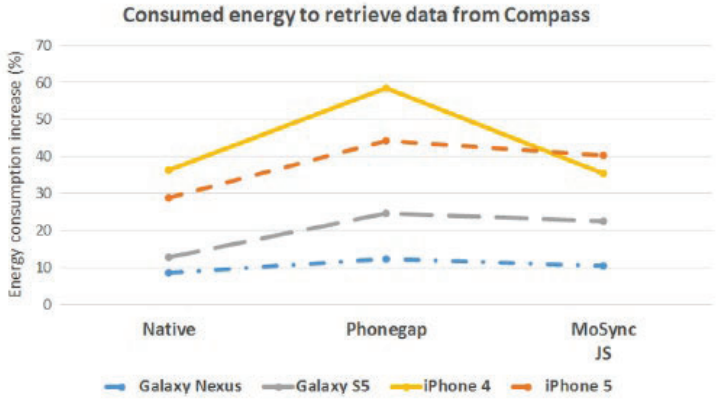
Base energy consumption



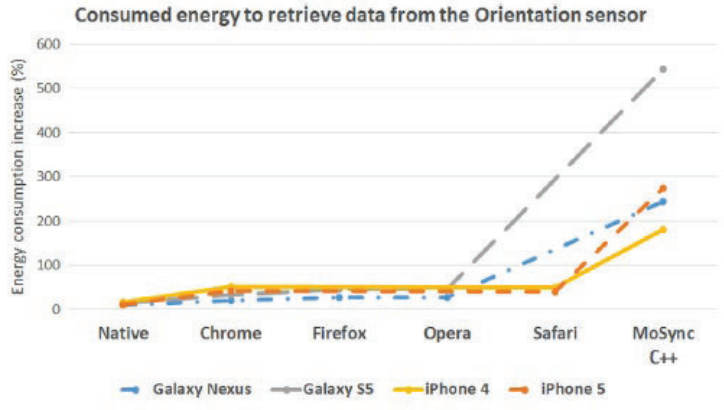
Results: accelerometer



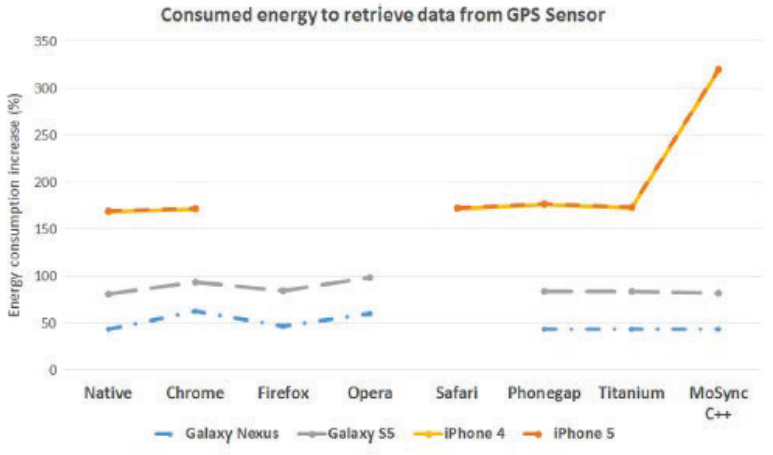
Results: compass



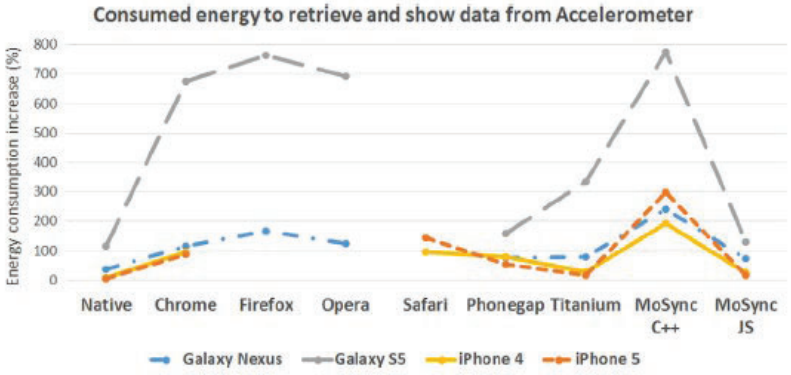
Results: orientation sensor



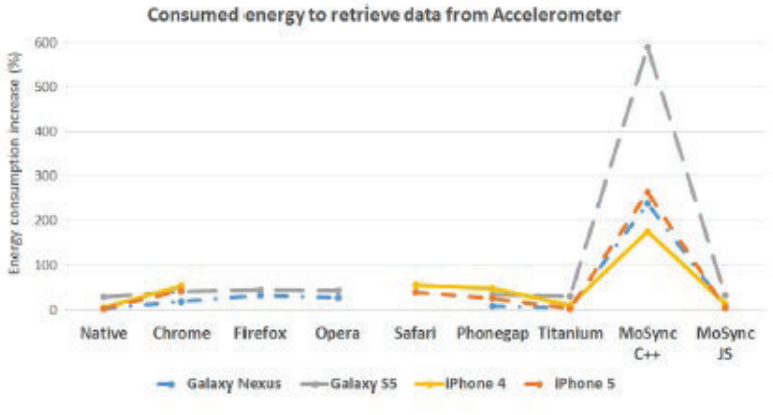
Results: GPS



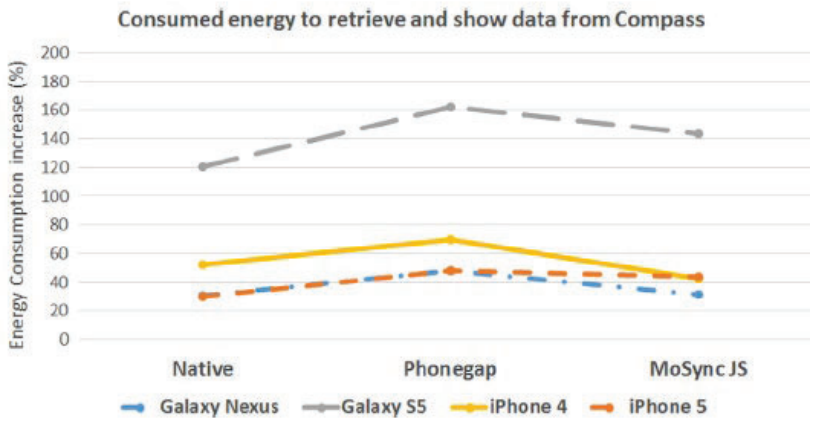
Data visualization: accelerometer



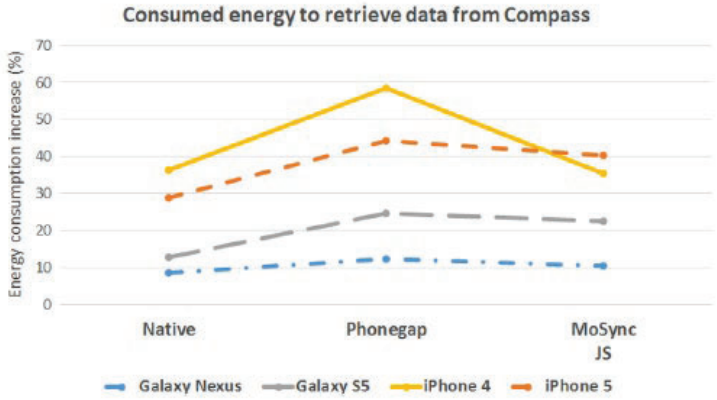
Results: accelerometer



Data visualization: compass



Results: Compass



Results

- 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

Conclusions - 1

- 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

Conclusions - 2

- 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

Future development

- An efficient cross-platform framework must provide an extremely efficient user interface rendering
- Improves 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

References

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.