Programming with Android: System Architecture

Luca Bedogni **Marco Di Felice**

Dipartimento di Scienze dell'Informazione
Università di Bologna

Outline

- Android Architecture: An **Overview**
- Android **Dalvik** Java **Virtual Machine**
- Android Components: **Activities**
- Android Components: **Intents**
- Android Components: **Services**
- Android Components: **Content Providers**
- Android Application **Distribution** and **Markets**

Android … What?

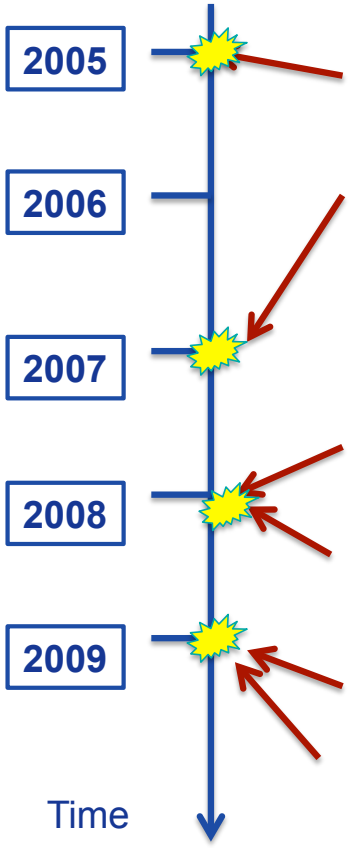
- **Android** is a *Linux-based* platform for *mobile devices* …
  - *Operating System*
  - *Middleware*
  - *Applications*
  - *Software Development Kit* (**SDK**)
- Which kind of **mobile devices** … (examples)

SMARTPHONES

TABLETS

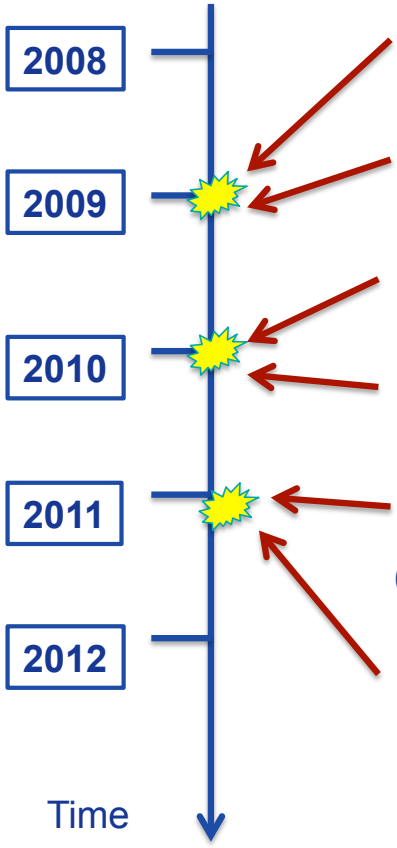
EREADERS

Android … When?



- **Google** buys Android from the **Android Inch**
- Open Handset Alliance (**OHA**) created for open standards for mobile devices. Partners of OHA: Google, Motorola, Samsung, Vodafone, T-Mobile, etc
- Android 1.0 Released
- The first Android smartphone: G1 HTC-Dream
- Android **1.1** Released
- Android **1.5 (CupCake)** Released

Android … When?



- Android **1.6** (**Donut**) Released
- Android **2.0** (**Eclair**) Released

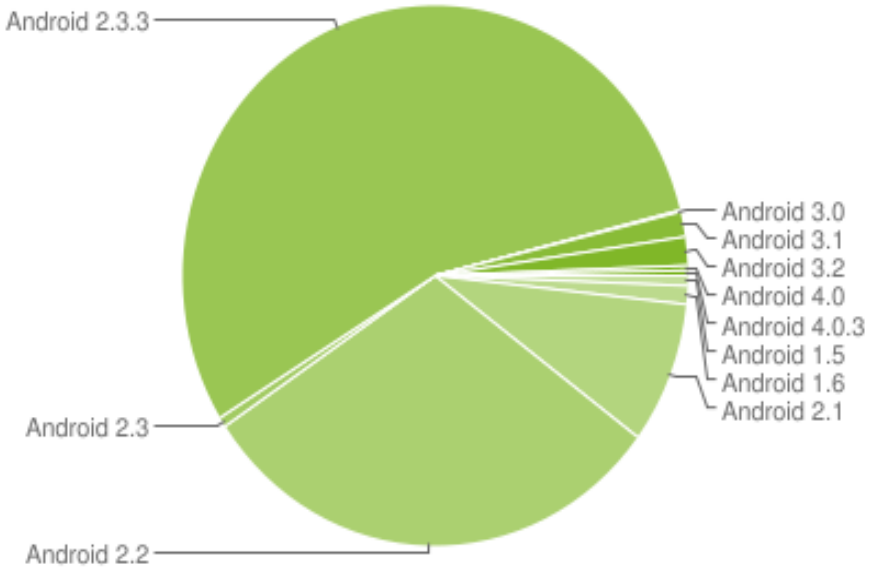
- Android **2.2** (**Froyo**) Released
- Android **2.3** (**Gingerbread**) Released
- Android **3.0** (**Honeycomb**) Released

(First version for devices with larger screens such as tablets)

- Android **4.0** (**Ice-Cream Sandwich**) Released. (It merges the 3.x tab centric design and the v2.x phone based design into a single version.)

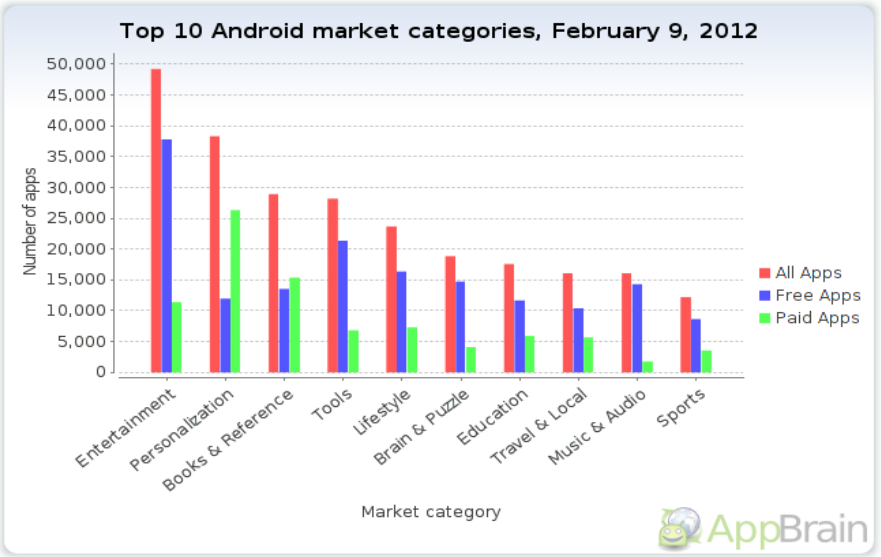
Android … When?

ANDROID DISTRIBUTIONS



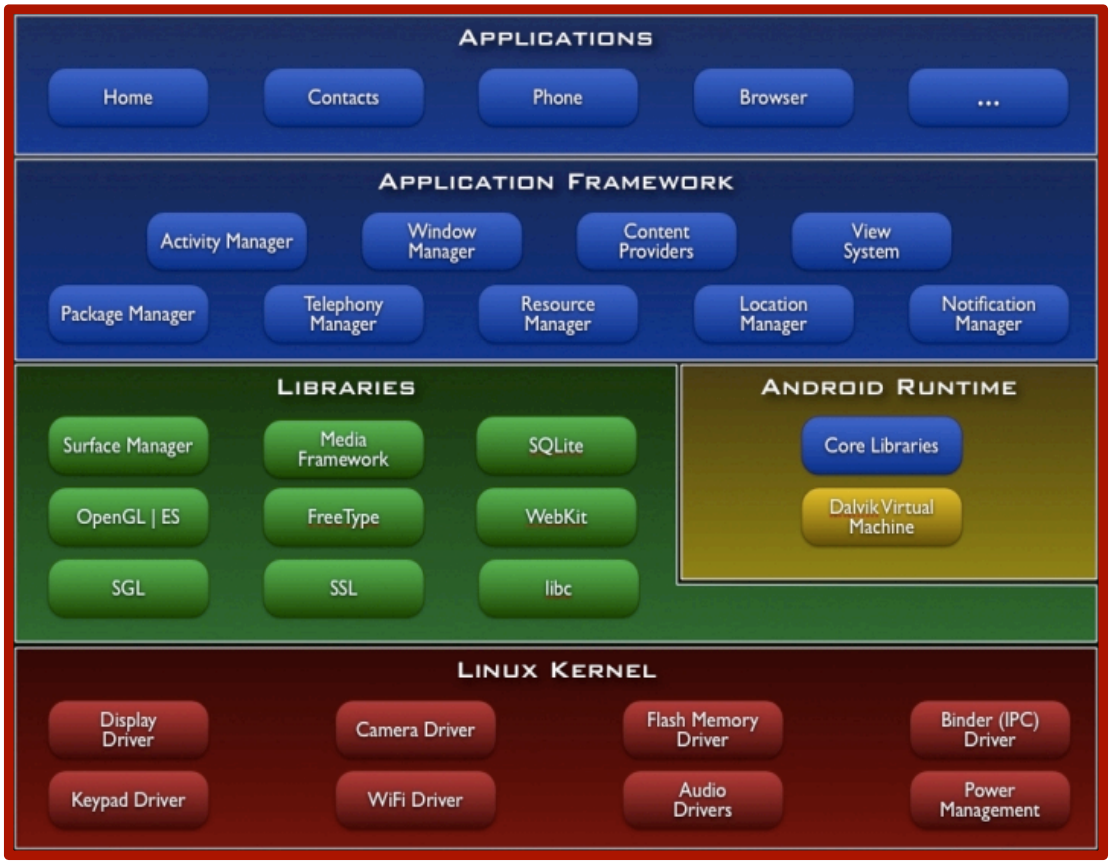
http://www.email-marketing-reports.com/wireless-mobile/smartphone-statistics.htm

ANDROID APPLICATIONS



http://www.appbrain.com/stats/android-market-app-categories

The Android Architecture

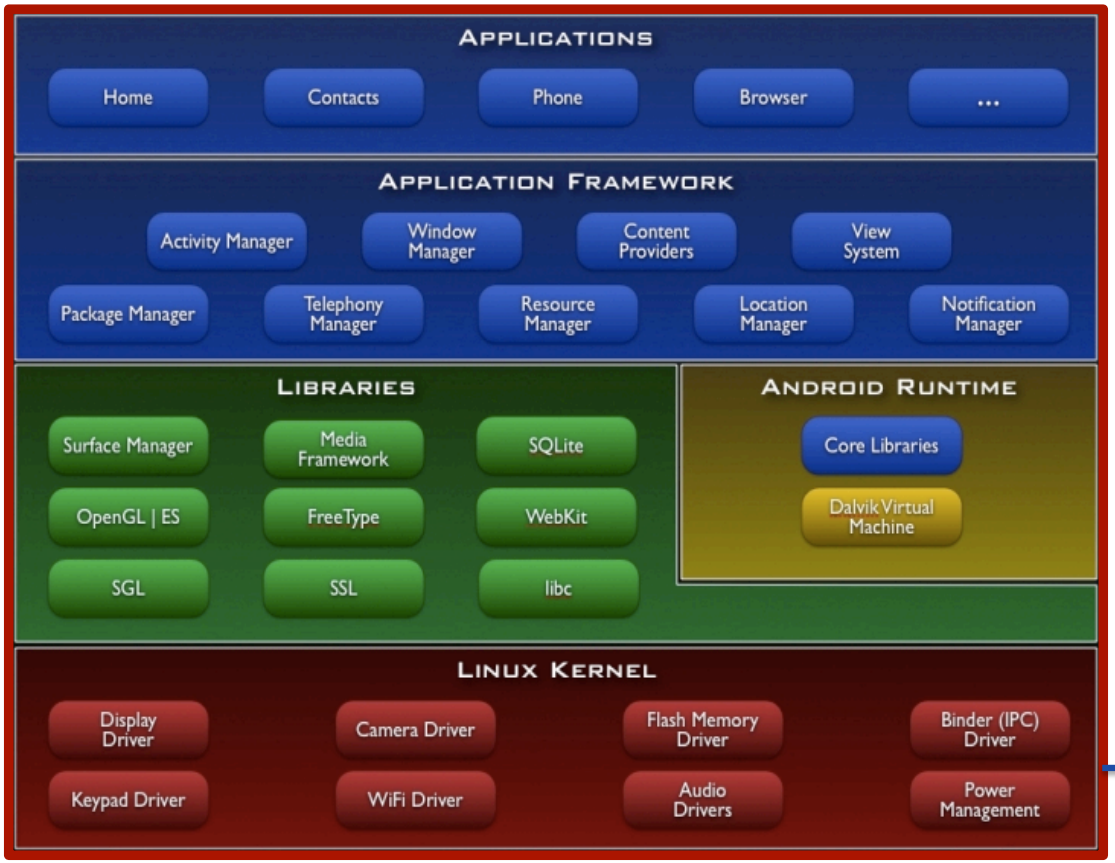


Stack
Architecture

Open Source Architecture
(Apache/MIT License v. 2.0)

Business-friendly License

The Android Architecture

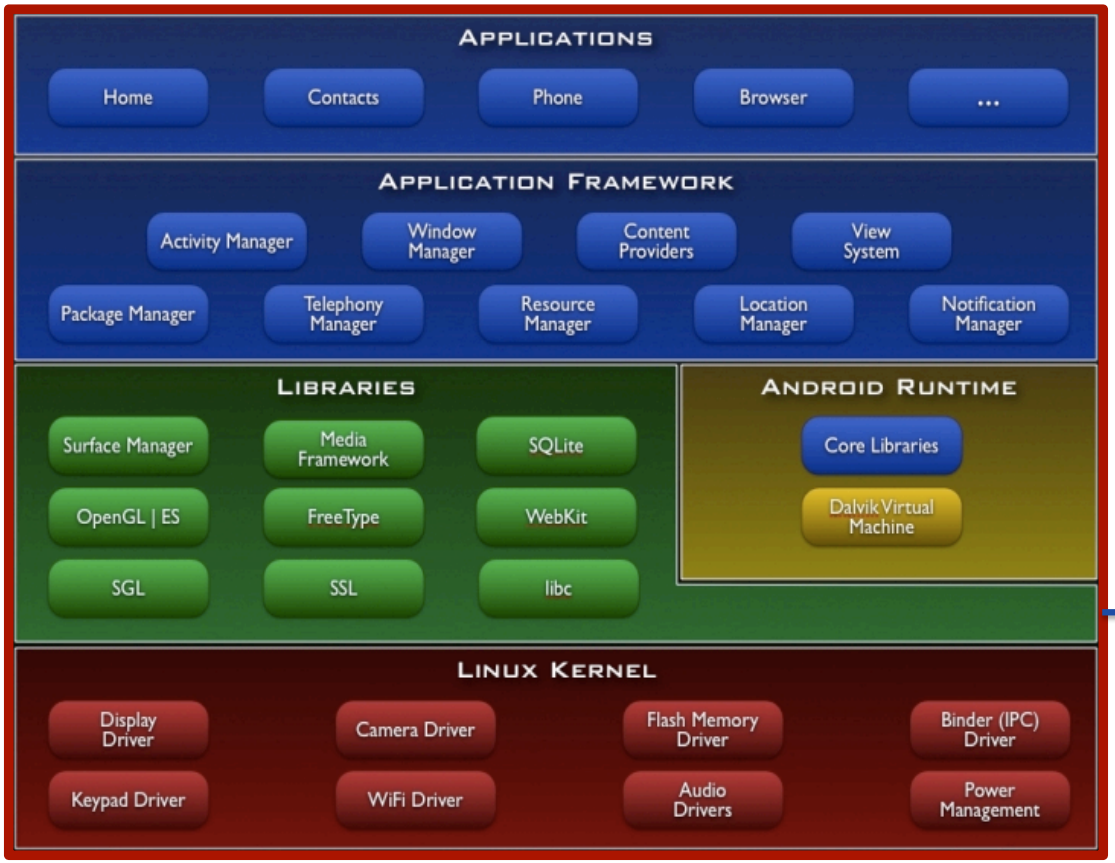


Built on top of **Linux kernel** (v. 2.6-3.0)

Advantages:

- **Portability** (i.e. easy to compile on different harwdare architectures)
- **Security** (e.g. secure multi-process environment)
- **Power** Management

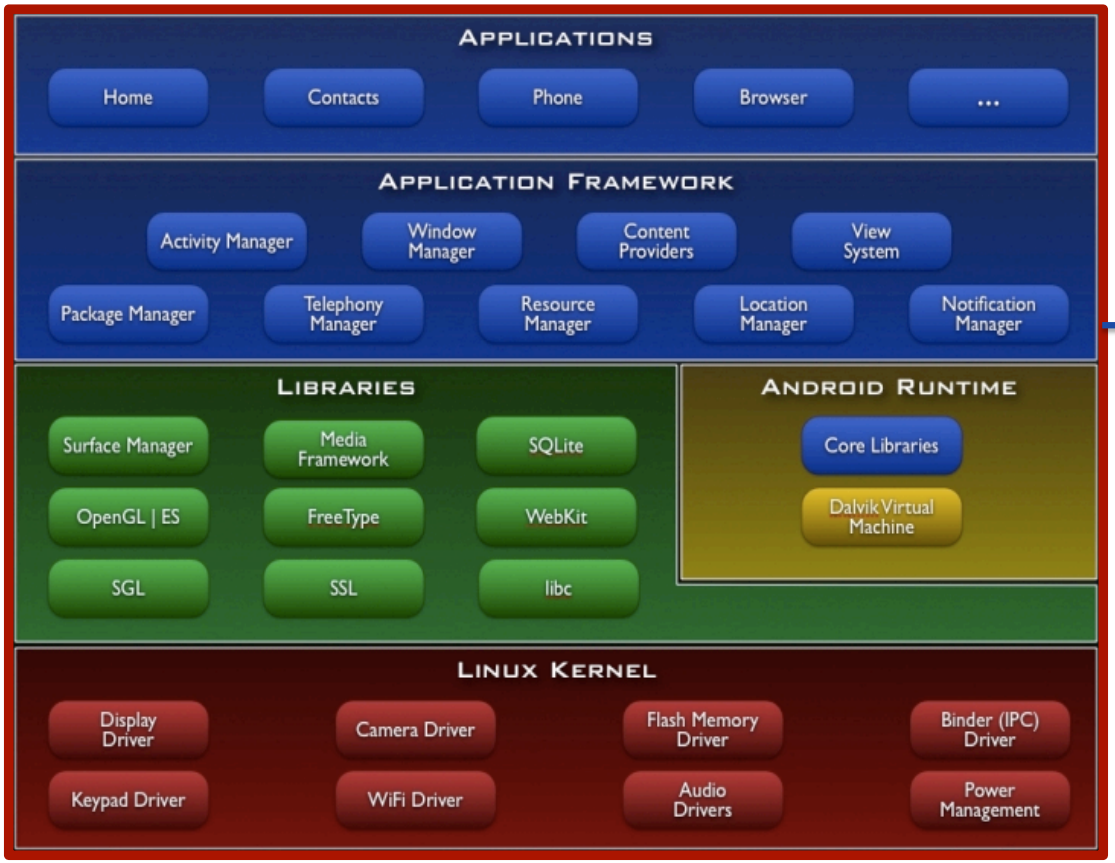
The Android Architecture



Native **Libraries**
(C/C++ code)

- **Graphics** (Surface Manager)
- **Multimedia** (Media Framework)
- **Database DBMS** (SQLite)
- **Font Management** (FreeType)
- **WebKit**
- **C libraries** (Bionic)
- ....

The Android Architecture

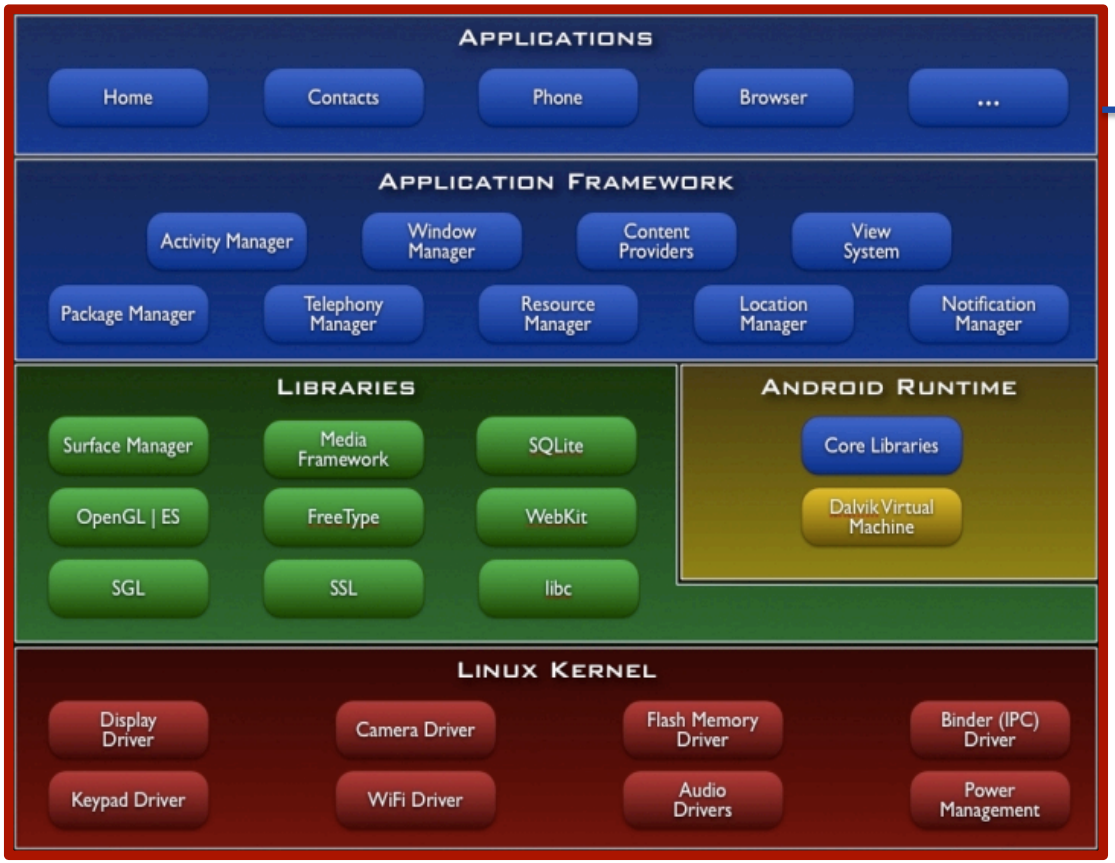


Application Libraries

(Core Components of Android)

- **Activity Manager**
- **Packet Manager**
- **Telephony Manager**
- **Location Manager**
- **Contents Provide**
- **Notification Manager**
- **….**

The Android Architecture

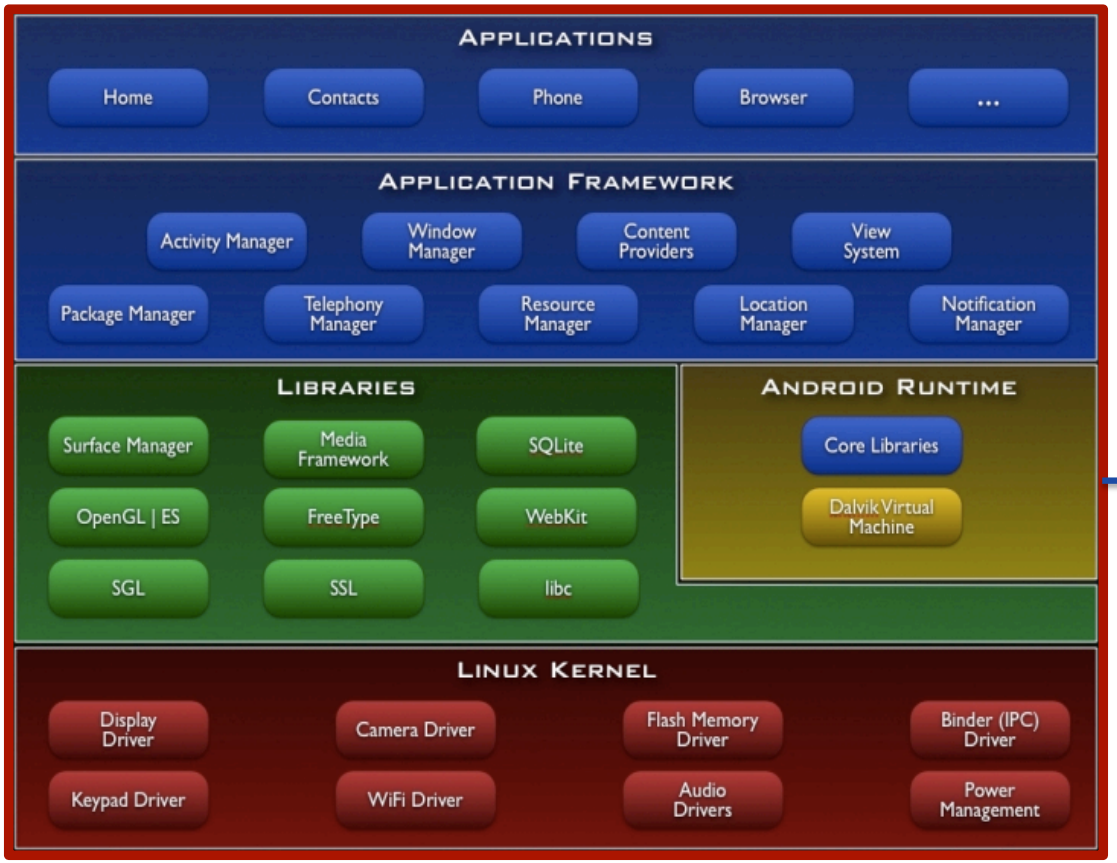


Applications

(Written in **Java** code)

- **Android Market**
- **Entertainment**
- **Productivity**
- **Personalization**
- **Education**
- **Geo-communication**
- **….**

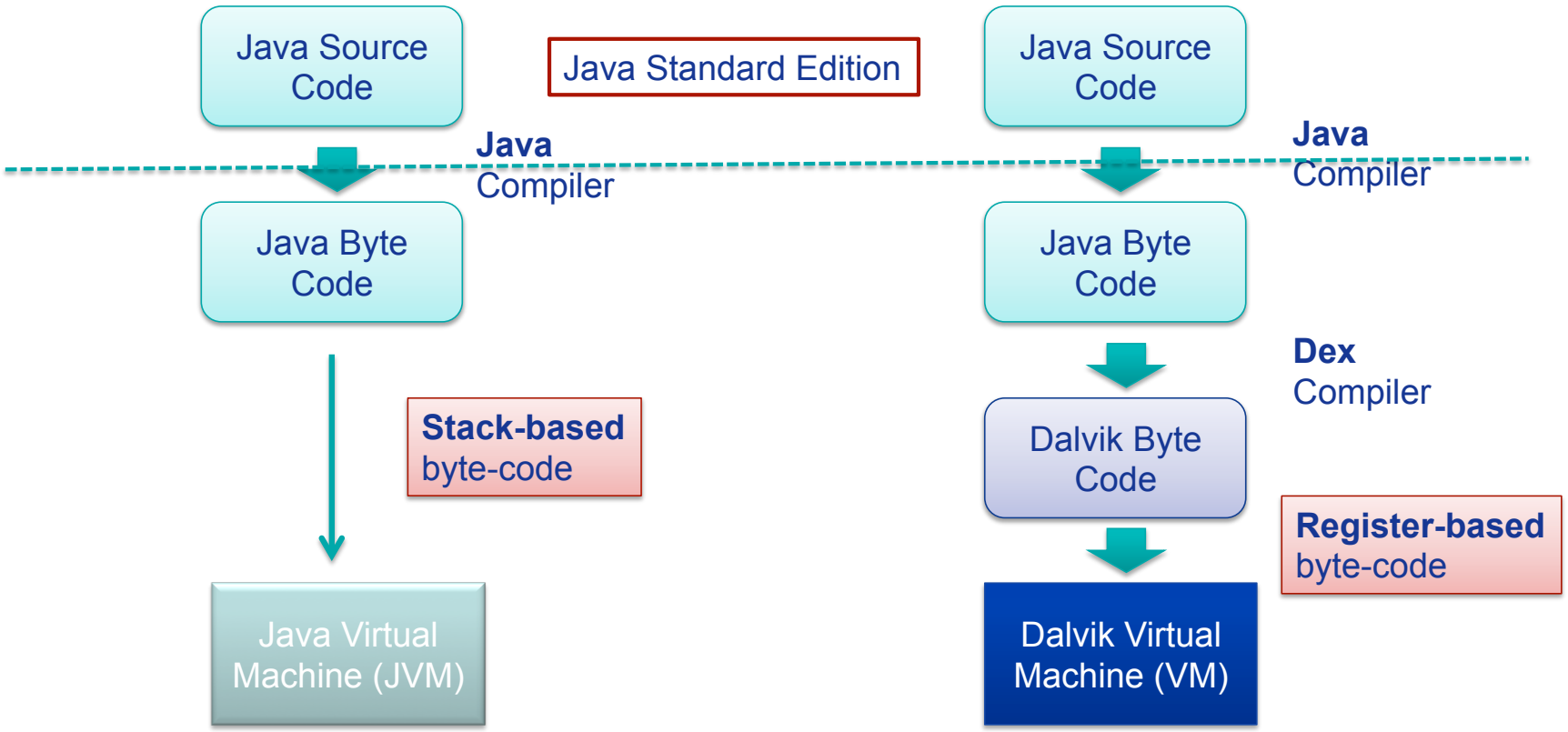
The Android Architecture



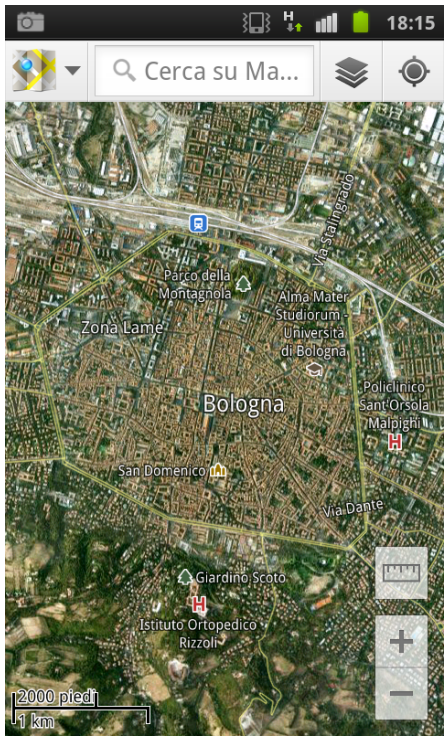
Dalvik Virtual Machine (VM)

- **Novel** Java Virtual Machine implementation (not using the Sun JVM)
- Open **License** (Sun JVM is not open!)
- **Optimized** for memory-constrained devices
- **Faster** than Sun JVM
- ….

Dalvik Java Virtual Machine (JVM)



Android Applications Design



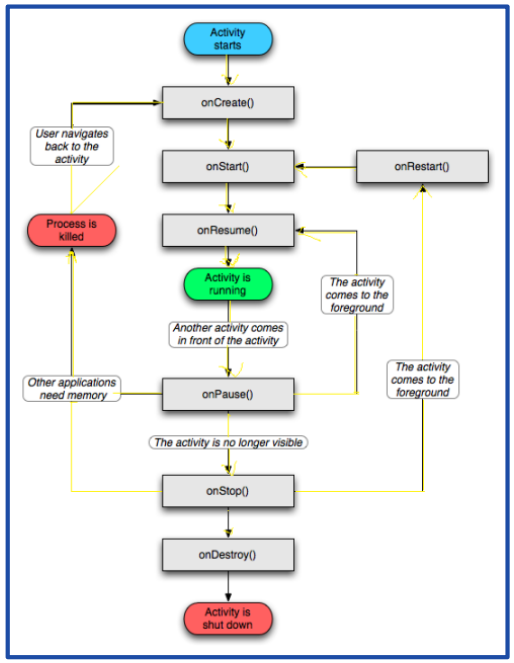
APPLICATION **DESIGN**:

- **GUI** Definition
- **Events** Management
- Application **Data** Management
- **Background** Operations
- **User** Notifications

Android Components: Activities

- An **Activity** corresponds to a **single screen** of the **Application**.
- An Application can be composed of *multiples screens* (Activities).
- The **Home Activity** is shown when the user launches an application.
- Different activities can exhange information one with each other.

Android Components: Activities



- The **Activity Manager** is responsible for creating, destroying, managing activities.
- Activities can be on different **states**: *starting*, *running*, *stopped*, *destroyed*, *paused*.
- Only one activity can be on the **running** state at a time.
- Activities are organized on a **stack**, and have an event-driven life cycle (details later …)

Android Components: Activities

- Each activity is composed by a list of *graphics components*.
- Some of these components (also called **Views**) can interact with the user by handling **events** (e.g. Buttons).
- Two ways to build the graphic interface:

PROGRAMMATIC APPROACH

Example:

```
Button button=new Button (this);
TextView text= new TextView();
text.setText("Hello world");
```

Android Components: Activities

- Each activity is composed by a list of *graphics components*.
- Some of these components (also called **Views**) can interact with the user by handling **events** (e.g. Buttons).
- Two ways to build the graphic interface:

DECLARATIVE APPROACH

Example:

```
< TextView android.text=@string/hello" android:textcolor=@color/blue
android:layout_width="fill_parent" android:layout_height="wrap_content" />
< Button android.id="@+id/Button01" android:textcolor="@color/blue"
android:layout_width="fill_parent" android:layout_height="wrap_content" />
```

Android Components: Activities

- *Android applications typically use both the approaches*!

DECLARATIVE APPROACH

XML Code

Define the Application **layouts** and **resources** used by the Application (e.g. labels).

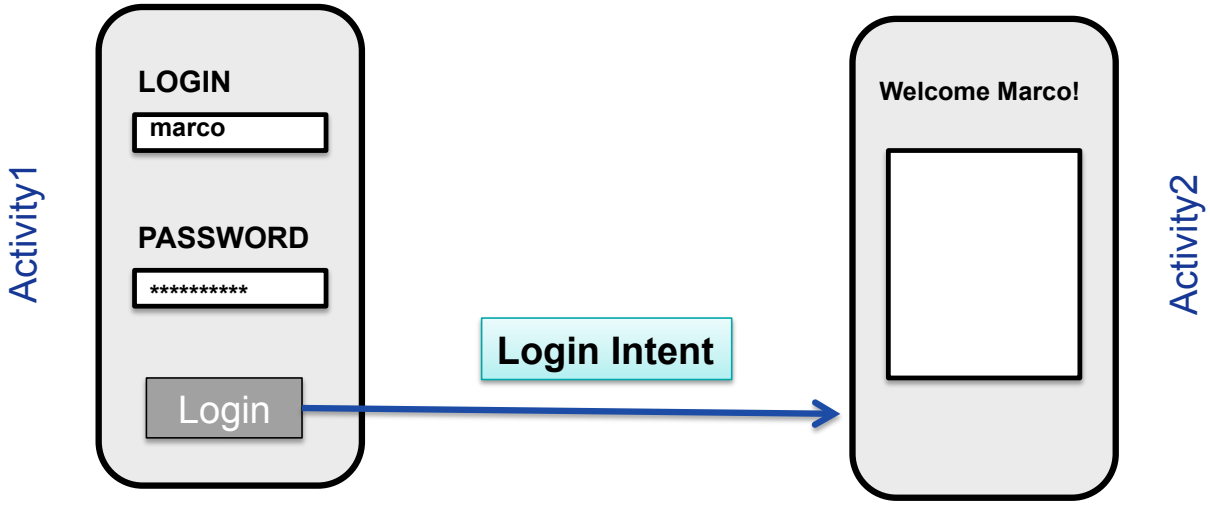
PROGRAMMATIC APPROACH

Java Code

Manages the **events**, and handles the **interaction** with the user.

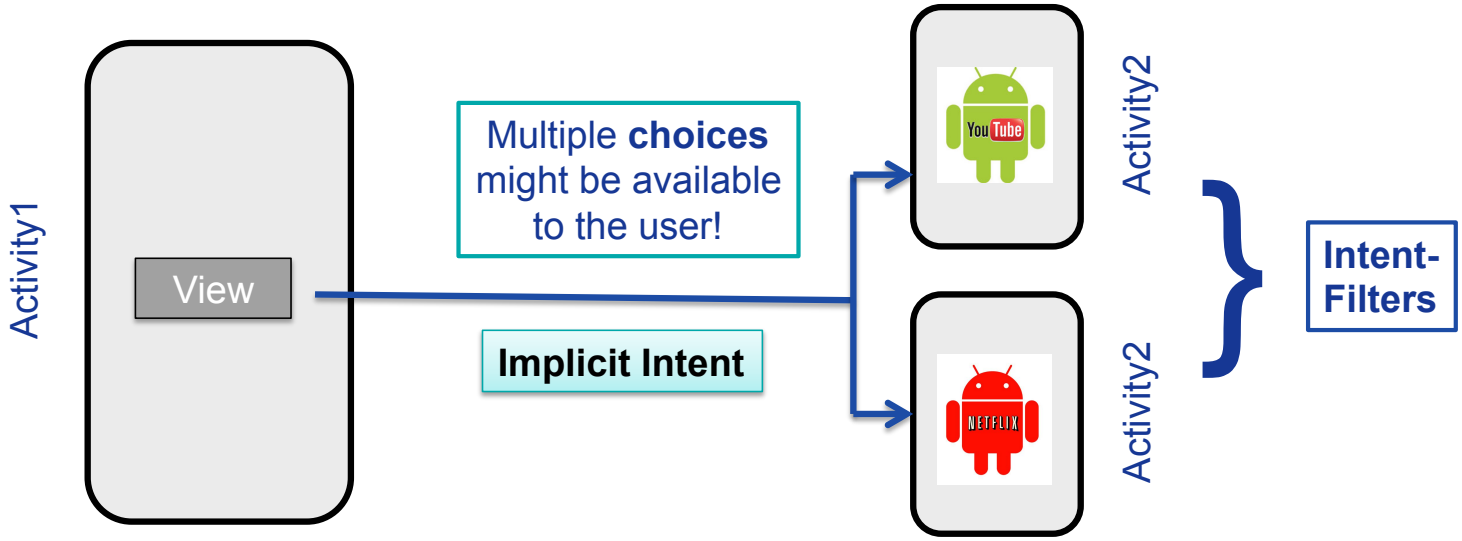
Android Components: Intents

- **Intents**: asynchronous **messages** to activate core Android components (e.g. Activities).
- **Explicit** Intent → The component *(e.g. Activity1)* specifies the destination of the intent *(e.g. Activity 2)*.



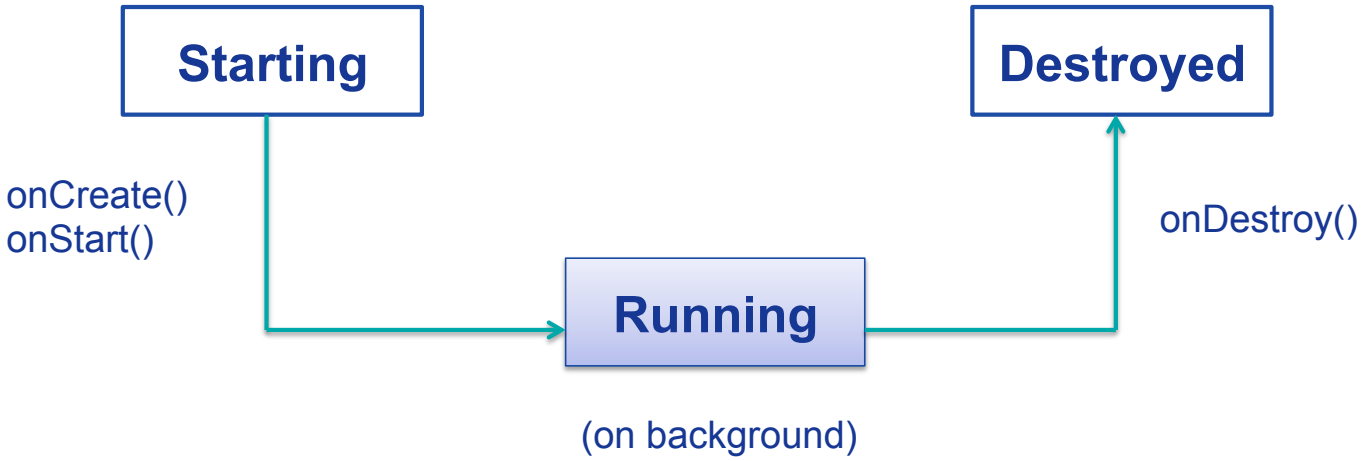
Android Components: Intents

- **Intents**: asynchronous **messages** to activate core Android components (e.g. Activities).
- **Implicit** Intent → The component *(e.g. Activity1)* specifies the type of the intent *(e.g. "View a video")*.



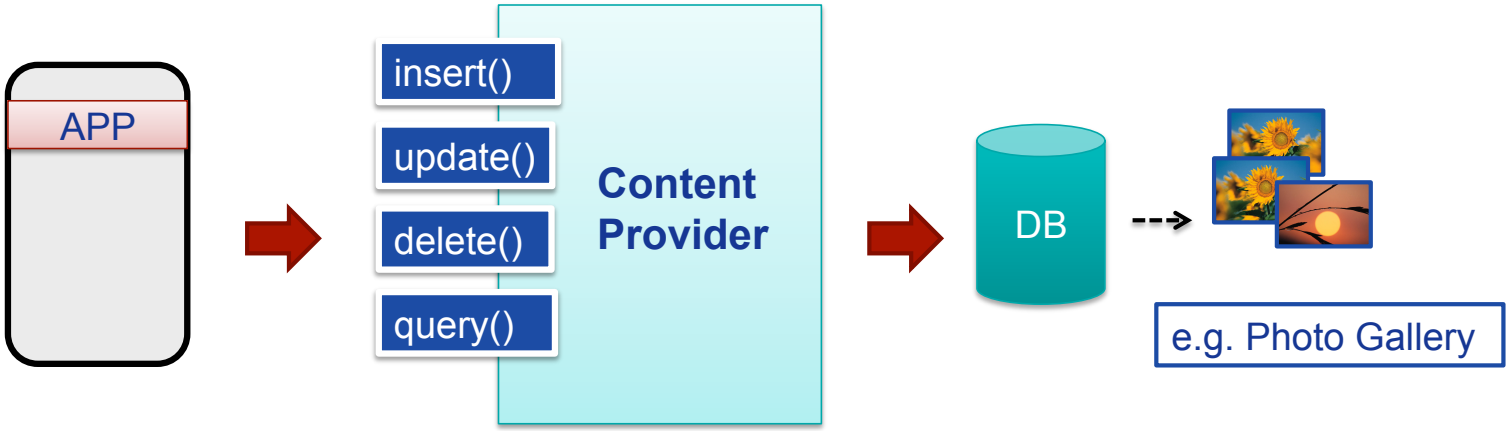
Android Components: Services

- **Services**: like Activities, but run in **background** and do not provide an user interface.
- Used for **non-interactive** tasks (e.g. networking).
- Service life-time composed of 3 states:

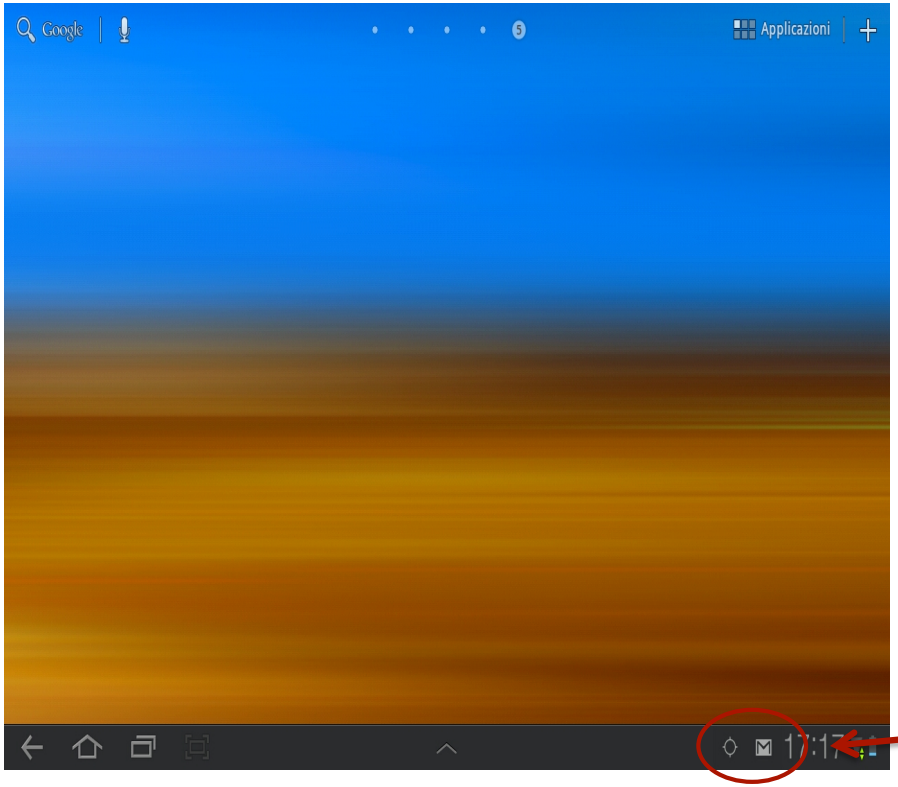


Android Components: Content Providers

- Each Android **application** has its own **private** set of data (managed through *files* or through *SQLite* database).
- **Content Providers**: Standard **interface** to *access and share data among different applications*.

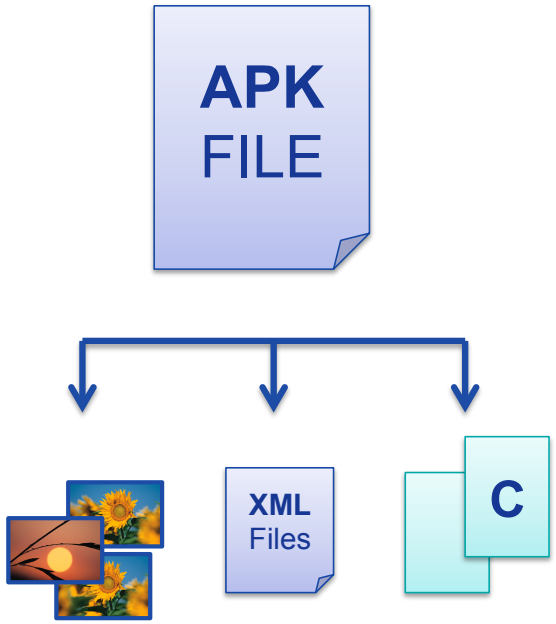


Android Components: Broadcast Receivers



- *Publish/Subscribe* paradigm
- **Broadcast Receivers**: An application can be signaled of **external events**.
- **Notification** types: Call incoming, SMS delivery, Wifi network detected, etc

Android Application Distribution



- Each Android **application** is contained on a single **APK** file.
  - Java **Byte-code** (*compiled for Dalvik JVM*)
  - **Resources** (e.g. images. videos, XML layout files)
  - **Libraries** (optimal native C/C++ code)

Android Application Distribution

- Each application must be signed through a **key** before being distributed.
- Applications can be **distributed** via *Web* or via *Markets*.
- **Android Market**: application store run by Google … but several other application stores are available (they are just normal applications).