Laboratorio di programmazione di sistemi mobili e tablet

Prefazione

Marco Ronchetti
Università degli Studi di Trento

Intro to the course

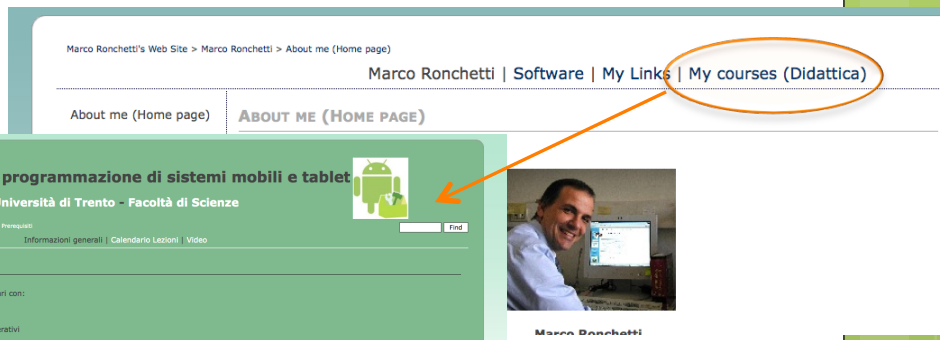
2 teachers (Marco Ronchetti - Giuseppe Riccardi)

2 teaching assistants (Carlo Menapace – Carmelo Ferrante)

Final project

web site: google for “marco ronchetti”, go to “My courses”

Videos might be available for the next theory Lectures.



Per poter frequentare il corso con profitto, gli studenti devono essere familiari con:

la programmazione in Java

Basi di dati

Nozioni base di sistemi operativi

Nozioni base di reti, di ingegneria del software e familiarità con Linux sono utili.

E' OBBLIGATORIO aver superato un esame di Programmazione ad oggetti (es. Progammazione 2 per Informatica) per potersi iscrivere all'esame.

Why is mobile programming different?

- Screen: from small phones to large TV sets
- OS version (multiple APK)
- Scarce resources (memory, disk)
- Unreliable and mutable connectivity (GSM, WiFi)
- Data transfer: costly, slow, high latency
- Battery
- Priorities (what if a phone call comes in?)
- User interaction (no kbd, gestures…)
- Devices (accelerometer, GPS, camera, audio, mic)
- Speech APIs
- Inter-app communication
- Security threats
- Development model (cross compilation)
- Distribution model (store)

Design philosophy

Applications should be:

- – Fast
  - In spite of the constraints: < 200 MB RAM, slow processor
- – Responsive
  - Apps must respond to user actions within 5 seconds
- – Secure
  - Apps declare permissions in manifest
- – Seamless
  - Usability is key, persist data, suspend services
  - The OS may kill processes in background as needed

History and context

Marco Ronchetti
Università degli Studi di Trento

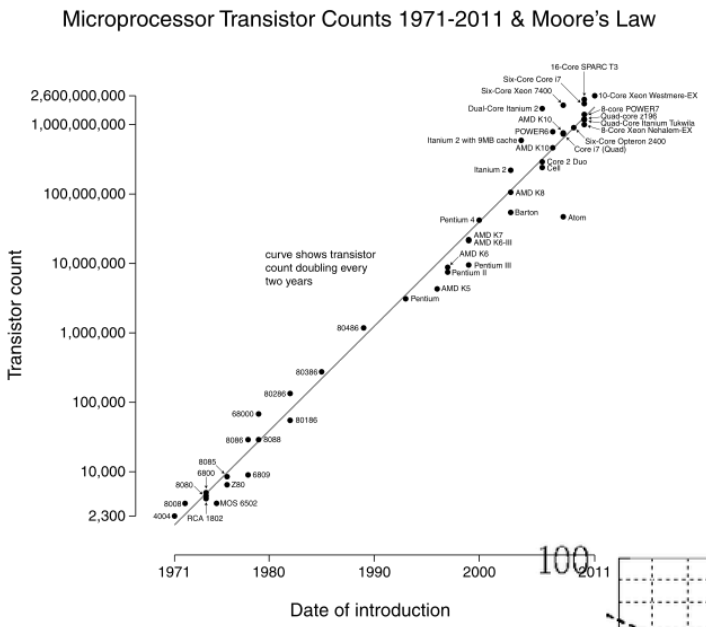
Moore's law

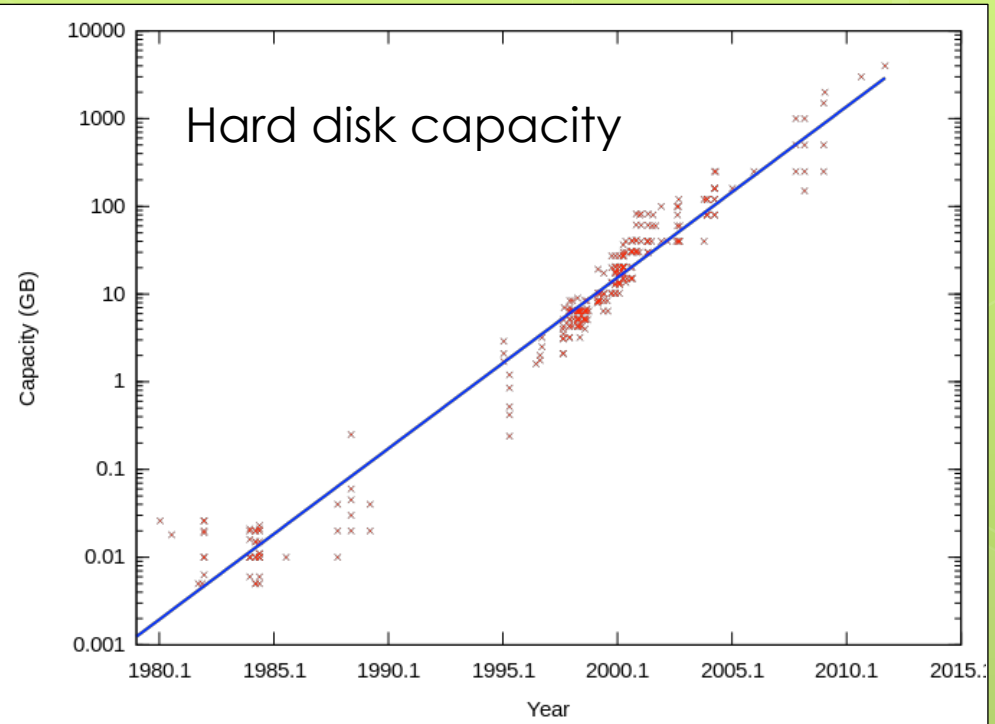
The number of transistors that can be placed inexpensively on an integrated circuit doubles approximately every two years.

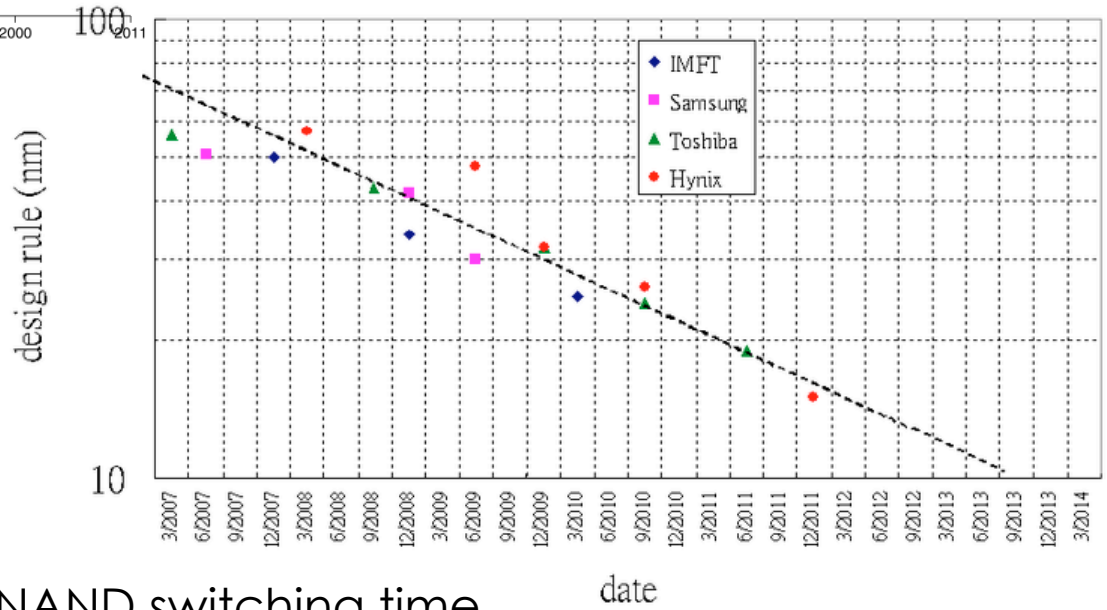
The period often quoted as "18 months" is due to David House, an Intel executive, who predicted that period for a doubling in chip performance (being a combination of the effect of more transistors and them being faster).

Quoted from wikipedia

Moore's law



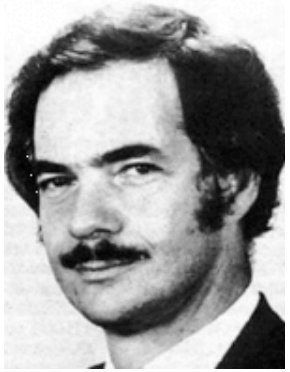




Batteries!

NAND switching time

The mobile computer



Adam Osborn

An Osborne Executive (early 80s) an an iPhone.

- 13,050 g / 135g = 100 times heavier [1]
- 4MHz / 412 Mhz = 100 times slower
- $2500 / $200-300 = 10 times more expensive
- (52cm x 23cm x 33cm)/(115mm x 61mm x 11.6mm)
- = 485 times as large (volume)

Images from wikimedia

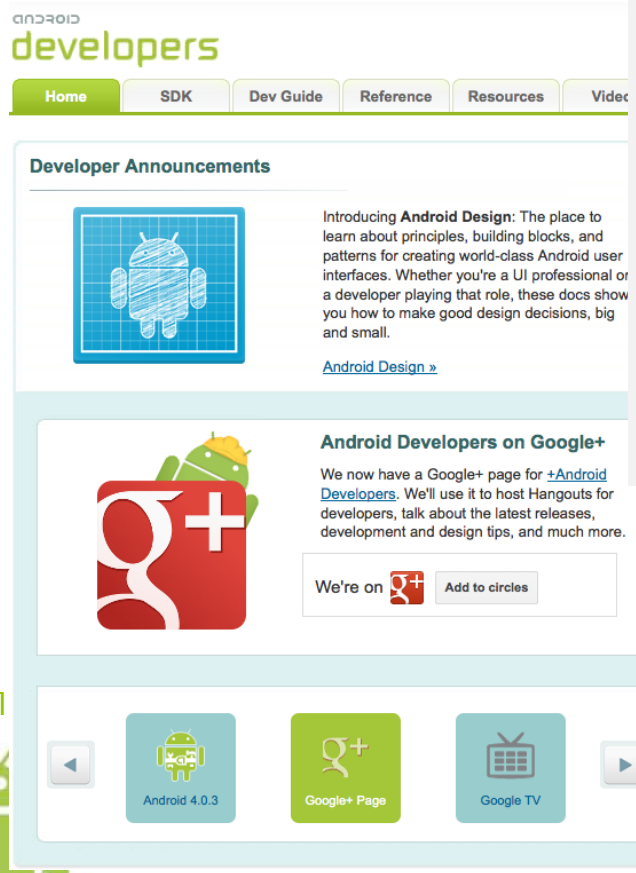
Smartphones

IBM Simon: concept product (1992), sold 1993

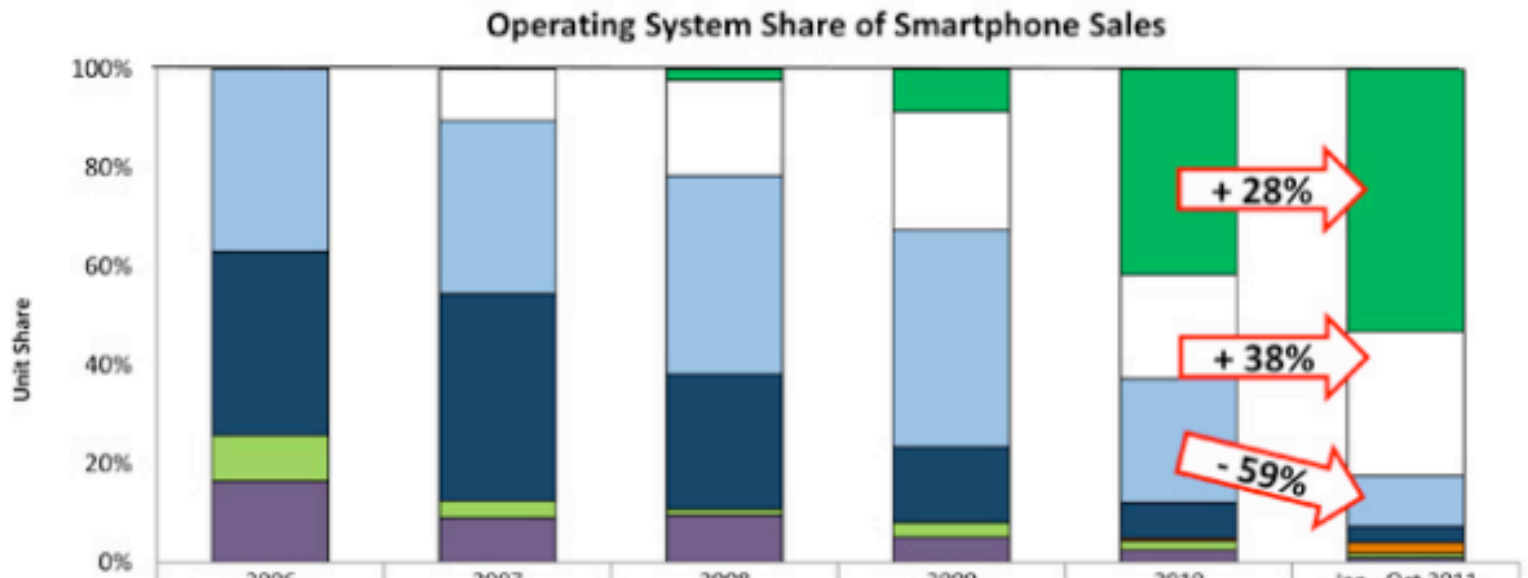
- mobile phone
- calendar,
- address book
- world clock
- calculator,
- note pad
- e-mail client
- send and receive faxes
- games.

touchscreen

Mobile: many worlds…



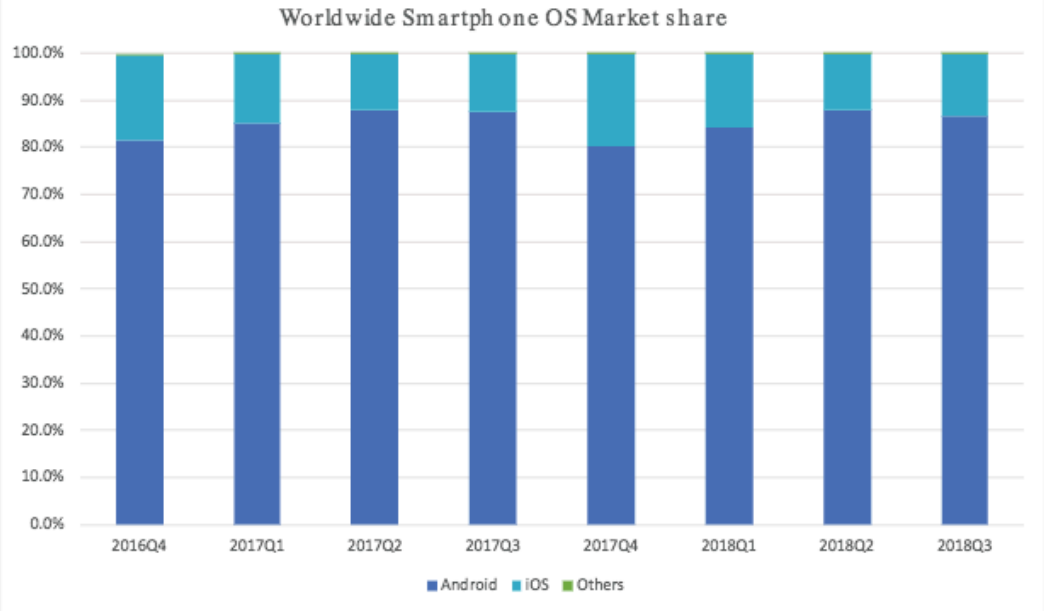
What the market says (2006-2011)



	2006	2007	2008	2009	2010	Jan - Oct 2011	2013
Android	0%	0%	2%	9%	42%	53%	82%
iOS	0%	11%	19%	24%	21%	29%	12%
RIM	37%	35%	40%	44%	25%	10%	1,8%
Windows Mobile	37%	42%	27%	15%	7%	3%	3,6%
Windows Phone 7	0%	0%	0%	0%	1%	2%	
Symbian OS	9%	3%	1%	3%	2%	1%	0,2%
Palm / webOS	17%	9%	10%	5%	3%	1%	

Source: The NPD Group, Consumer Tracking Service, Mobile Phone Track

What the market says (2016-2018)



Quarter	2016Q4	2017Q1	2017Q2	2017Q3	2017Q4	2018Q1	2018Q2	2018Q3
Android	81,4%	85,0%	88,0%	87,6%	80,3%	84,3%	87,8%	86,8%
iOS	18,2%	14,7%	11,8%	12,4%	19,6%	15,7%	12,1%	13,2%
Others	0,4%	0,2%	0,2%	0,1%	0,1%	0,0%	0,1%	0,0%
TOTAL	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%	100,0%

http://www.idc.com/prodserv/smartphone-os-market-share.jsp

Android or iOS ?

iOS:

- Develop in Objective-C
- Develop (only) on Macs (with emulator)

iPhone Open Development

- Jailbreak your iPhone or Touch
- Develop on any computer
  - Apps will not work with App Store
  - Device may not work with upgrades
- Need device! (iPhone or Touch)

Android:

- Develop in Java
- Develop on any platform (with emulator)

Introduction to Android

Laboratorio di programmazione di sistemi mobili e tablet

Marco Ronchetti, Università di Trento

What Android is not

- A Java ME implementation
- Part of the Linux Phone Standard Forum
- “only” an application layer
- A mobile phone

What Android is

From: http://source.android.com/

Android is:

an open-source software stack for mobile devices,

and a corresponding open-source project led by Google.

"We created Android in response to our own experiences launching mobile apps. We wanted to make sure that there was no central point of failure, so that no industry player can restrict or control the innovations of any other."

"That's why we created Android, and made its source code open." (under Apache Software Licence, 2.0)

The Android system tries to avoid incorporating GPL components

Why Android is not LGPL

LGPL requires either:

- shipping of source to the application;
- a written offer for source;
- linking the LGPL-ed library dynamically and allowing users to manually upgrade or replace the library.

Since Android software is typically shipped in the form of a static system image, complying with these requirements ends up restricting OEMs' designs. (For instance, it's difficult for a user to replace a library on read-only flash storage.)

For more details: http://source.android.com/source/licenses.html

Development and governance

- At any given moment, there is a current latest release of the Android platform
- Device builders and Contributors work with the current latest release, fixing bugs, launching new devices, experimenting with new features, and so on.
- In parallel, Google works internally on the next version of the Android platform and framework, working according to the product's needs and goals. We develop the next version of Android by working with a device partner on a flagship device whose specifications are chosen to push Android in the direction we believe it should go.
- When the "n+1"th version is ready, it will be published to the public source tree, and become the new latest release.

Android and Linux

Android relies on Linux version 2.6 for core system services such as security, memory management, process management, network stack, and driver model.

The kernel also acts as an abstraction layer between the hardware and the rest of the software stack.

Early History

- Oct 2003 Android, Inc. founded in Palo Alto
- 2005 Google buys Android, Inc..
- 2007 Open Handset Alliance is announced. Android is officially open-sourced.
- 2008 Android SDK 1.0 is released. The G1 phone, manufactured by HTC, is sold by T-Mobile USA.
- 2009 sees a proliferation of Android-based devices (20+ devices run Android).
- 2010 Android is 2nd only to RIM as best-selling smart phone platform. 60+ devices run Android

Three components

The Android Compatibility Program defines the technical details of Android platform and provides tools used by OEMs to ensure that developers' apps run on a variety of devices.

The Android SDK provides built-in tools that Developers use to clearly state the device features their apps require.

The Android Market shows apps only to those devices that can properly run them.

The main building blocks

Device Hardware: Android runs on a wide range of hardware configurations including smart phones, tablets, and set-top-boxes. Android is processor-agnostic, but it does take advantage of some hardware-specific security capabilities (e.g. on ARM).

Android Operating System: The core operating system is built on top of the Linux kernel. All device resources are accessed through the operating system.

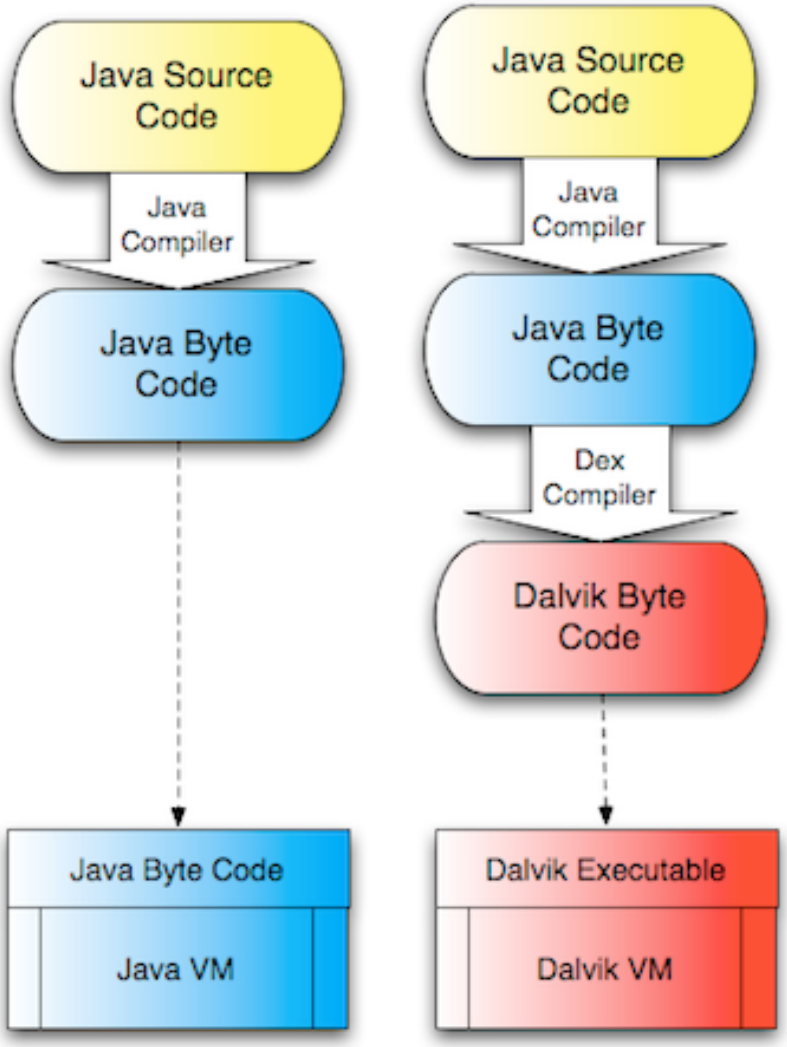
Android Application Runtime: Android applications are most often written in Java and run in the Dalvik V.M.

However, many applications, including core Android services and applications are native applications or include native libraries.

Both Dalvik and native applications run within the same security environment, in the Application Sandbox.

Applications get a dedicated part of the filesystem in which they can write private data, including databases and raw files.

Java vs. Dalvik



Dalvik is the managed runtime used by applications and some system services on Android. Dalvik was originally created specifically for the Android project.

Specification of the bytecode format, .dex (dalvik executable) and Dalvik VM Instruction Formats are available at

http://source.android.com/tech/dalvik/index.html

Dalvik

Every Android application runs in its own process, with its own instance of the Dalvik virtual machine.

Dalvik has been written so that a device can run multiple VMs efficiently. The Dalvik VM executes files in the Dalvik Executable (.dex) format which is optimized for minimal memory footprint.

The Dalvik VM relies on the Linux kernel for underlying functionality such as threading and low-level memory management.

Android is non standard Java

Standard Java distributions:

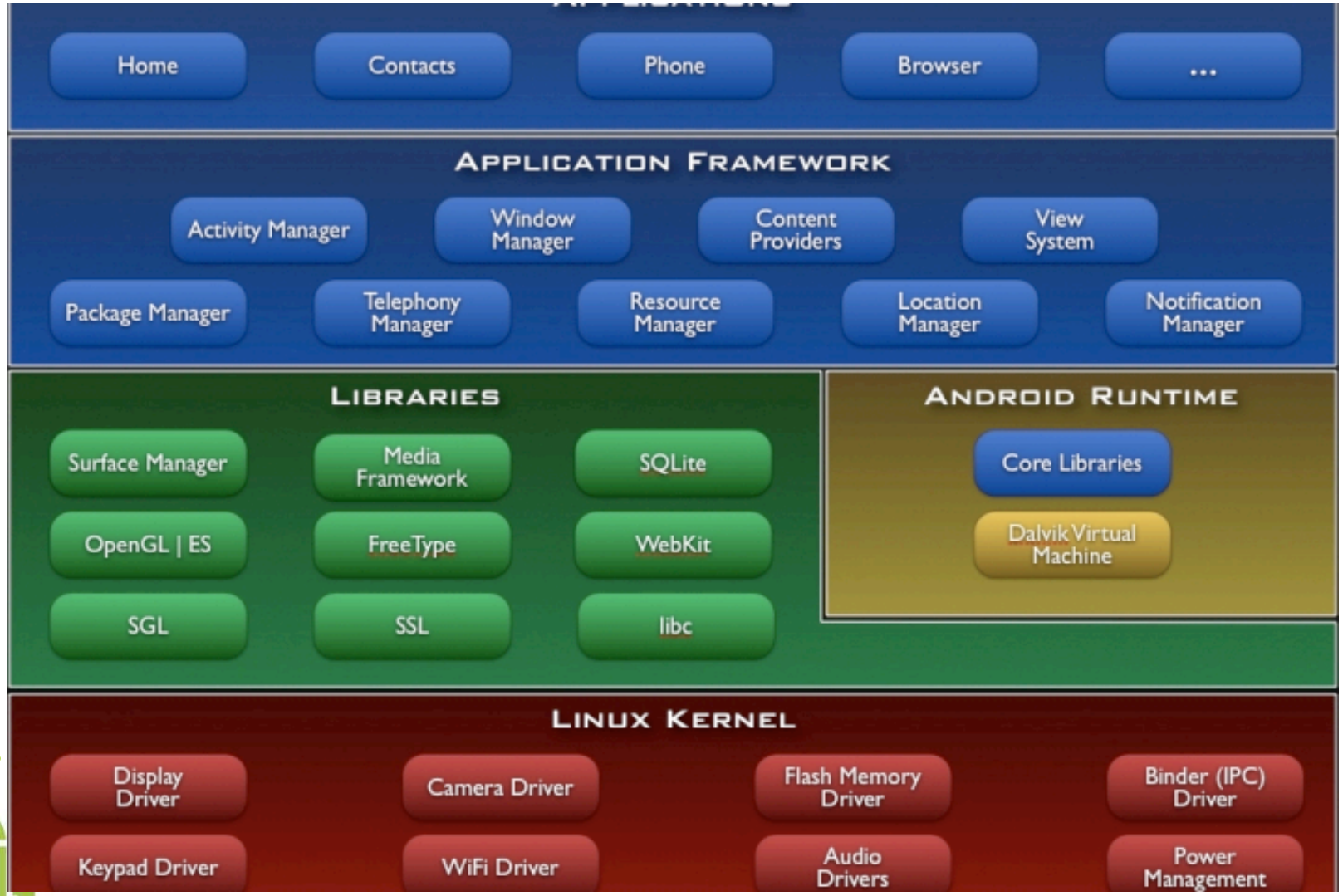
1. Java Standard Edition: used for development on basic desktop-type applications.
2. Java Enterprise Edition (aka J2EE or JavaEE): used for development of enterprise applications.
3. Java Micro Edition (aka J2ME or JavaME): Java for mobile applications.

Android's Java set of libraries is closest to Java Standard Edition. The major difference is that Java libraries for user interface (AWT and Swing) have been taken out and replaced with Android-specific user interface libraries. Android also adds quite a few new features to standard Java while supporting most of Java's standard features.

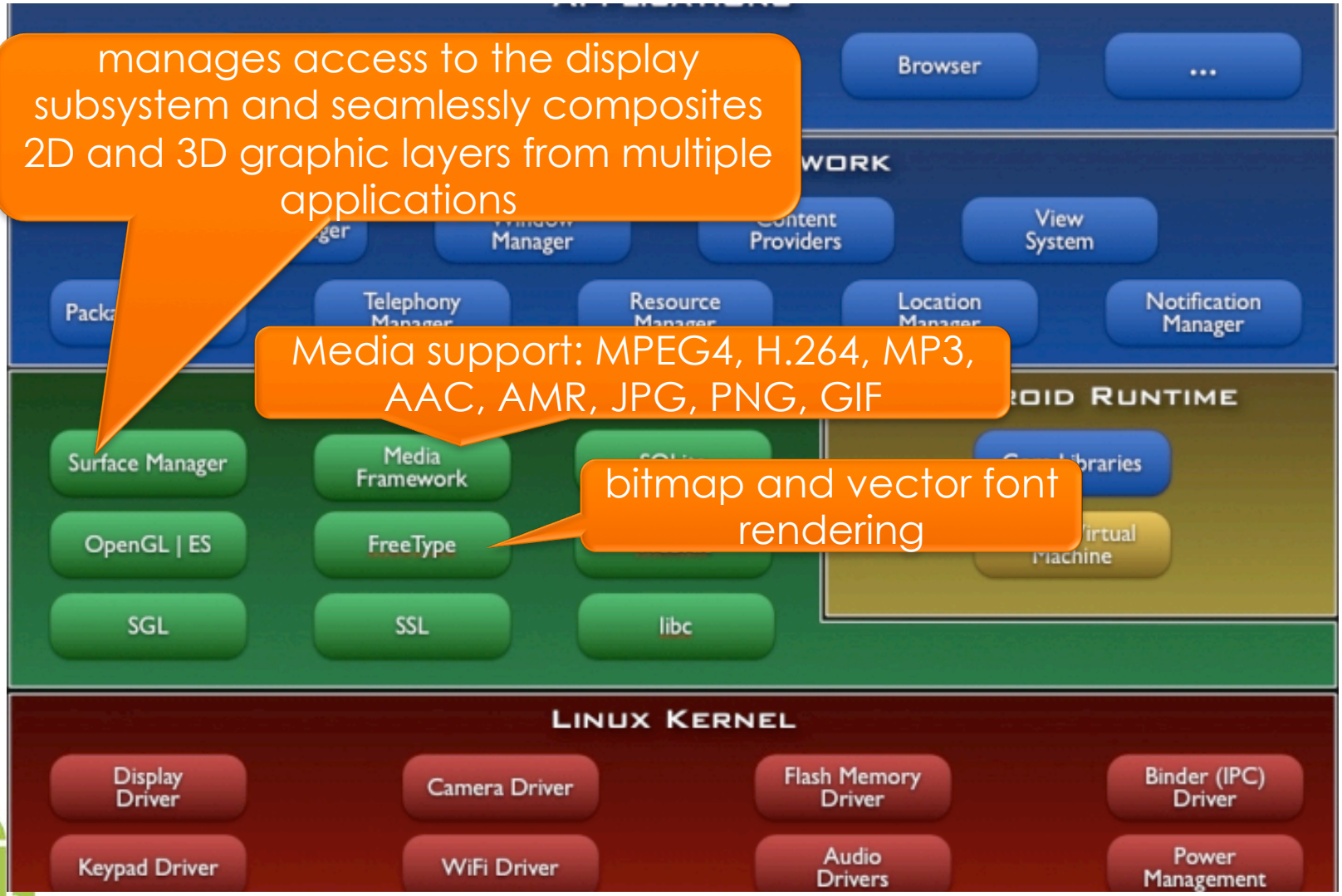
What Android supports

- User Interface
  - IO widgets (buttons, textboxes, lists)
  - Images
  - 2D/3D drawing
- Database
- Integrated browser
- Media support (audio, video, images; camera)
- Application framework lifecycle
- Connectivity (bluetooth, wi-fi, EDGE, 3G)
- Sensors (GPS/Geo-location, accelerometer, compass)
- GSM Telephony (call – sms)
- Google Maps
- Multiple processes
  - Managed by Android Dalvik VM
  - Background Services
  - Interprocess communications (e.g. Intents)
- Rich development environment including a device emulator, debugging tools, memory and performance profiling, Eclipse plug in

The Android stack



The Android stack

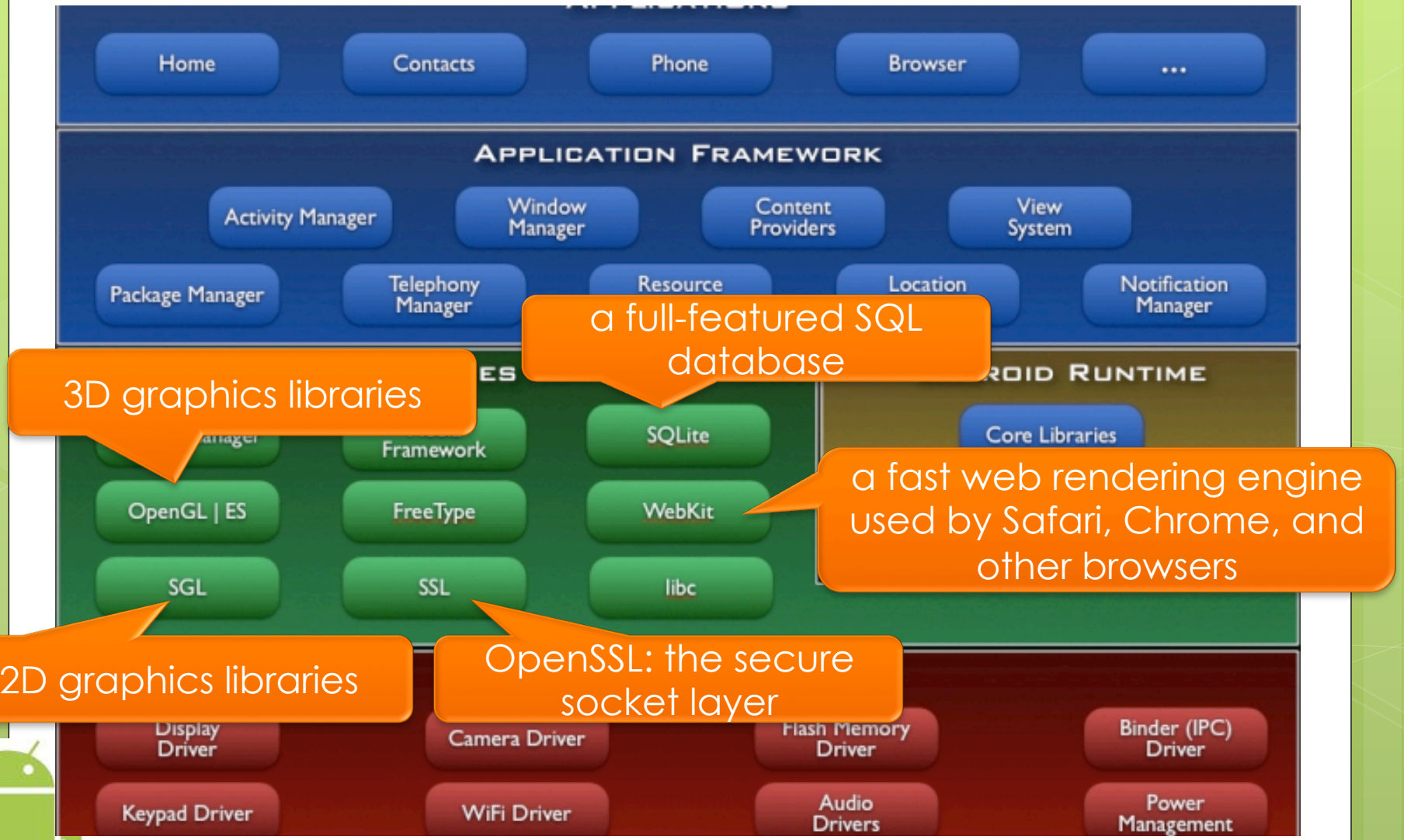


manages access to the display subsystem and seamlessly composites 2D and 3D graphic layers from multiple applications

Media support: MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF

bitmap and vector font rendering

The Android stack



Android NDK

The Android NDK is a toolset that lets you embed components that make use of native code in your Android applications.

Android applications run in the Dalvik virtual machine. The NDK allows you to implement parts of your applications using native-code languages such as C and C++. This can provide benefits to certain classes of applications, in the form of reuse of existing code and in some cases increased speed.

When to develop in NDK

"Using native code does not result in an automatic performance increase, but always increases application complexity."

"In general, you should only use native code if it is essential to your application, not just because you prefer to program in C/C++."

Android applications

Pre-Installed Applications:

phone, email, calendar, web browser, and contacts. These function both as user applications and to provide key device capabilities that can be accessed by other applications. Pre-installed applications may be part of the open source Android platform, or they may be developed by an OEM for a specific device.

User-Installed Applications:

Android provides an open development environment supporting any third-party application. The Android Market offers users hundreds of thousands of applications.

Cloud-based services

Android Market: a collection of services that allow users to discover, install, and purchase applications from their Android device or the web.

The Market also provides community review, application license verification, and other security services.

Android Update Service: delivers new capabilities and security updates to Android devices, including updates through the web or over the air (OTA).

Application Services: Frameworks that allow Android applications to use cloud capabilities such as

- (backing up) application data and settings
- cloud-to-device messaging (C2DM) for push messaging.

Platform versions 2008 -2015

Oct.2015	Android 6.0	23	M	API Changes
Oct..2014	Android 5.1	22	LOLLIPOP_MR1	Platform Highlights
	Android 5.0	21	LOLLIPOP	
	Android 4.4W	20	KITKAT_WATCH	KitKat for Wearables Only
Set..2013	Android 4.4	19	KITKAT	Platform Highlights
Lug..2013	Android 4.3	18	JELLY_BEAN_MR2	Platform Highlights
	Android 4.2, 4.2.2	17	JELLY_BEAN_MR1	Platform Highlights
	Android 4.1, 4.1.1	16	JELLY_BEAN	Platform Highlights
Nov.2011	Android 4.0.3, 4.0.4	15	ICE_CREAM_SANDWICH_MR1	Platform Highlights
	Android 4.0, 4.0.1, 4.0.2	14	ICE_CREAM_SANDWICH	
	Android 3.2	13	HONEYCOMB_MR2	
Feb 2011	Android 3.1.x	12	HONEYCOMB_MR1	Platform Highlights
	Android 3.0.x	11	HONEYCOMB	Platform Highlights
	Android 2.3.4 Android 2.3.3	10	GINGERBREAD_MR1	Platform Highlights
Dic 2010	Android 2.3.2 Android 2.3.1 Android 2.3	9	GINGERBREAD	
Mag 2010	Android 2.2.x	8	FROYO	Platform Highlights
	Android 2.1.x	7	ECLAIR_MR1	Platform Highlights
Ott. 2009	Android 2.0.1	6	ECLAIR_0_1	
	Android 2.0	5	ECLAIR	
Apr 2009	Android 1.6	4	DONUT	Platform Highlights
	Android 1.5	3	CUPCAKE	Platform Highlights
	Android 1.1	2	BASE_1_1	
Sept. 2008	Android 1.0	1	BASE	

http://developer.android.com/guide/appendix/api-levels.html

Platform versions 2015-2019

Aug.2018	Android 9	28	P	Pie
	Android 8.1	27	O_MR1	
Aug. 2017	Android 8.0	26	O	Oreo
	Android 7.1.1 Android 7.1	25	N_MR1	
Aug. 2016	Android 7.0	24	N	Nougat
Oct. 2015	Android 6.0	23	M	Marshmallow

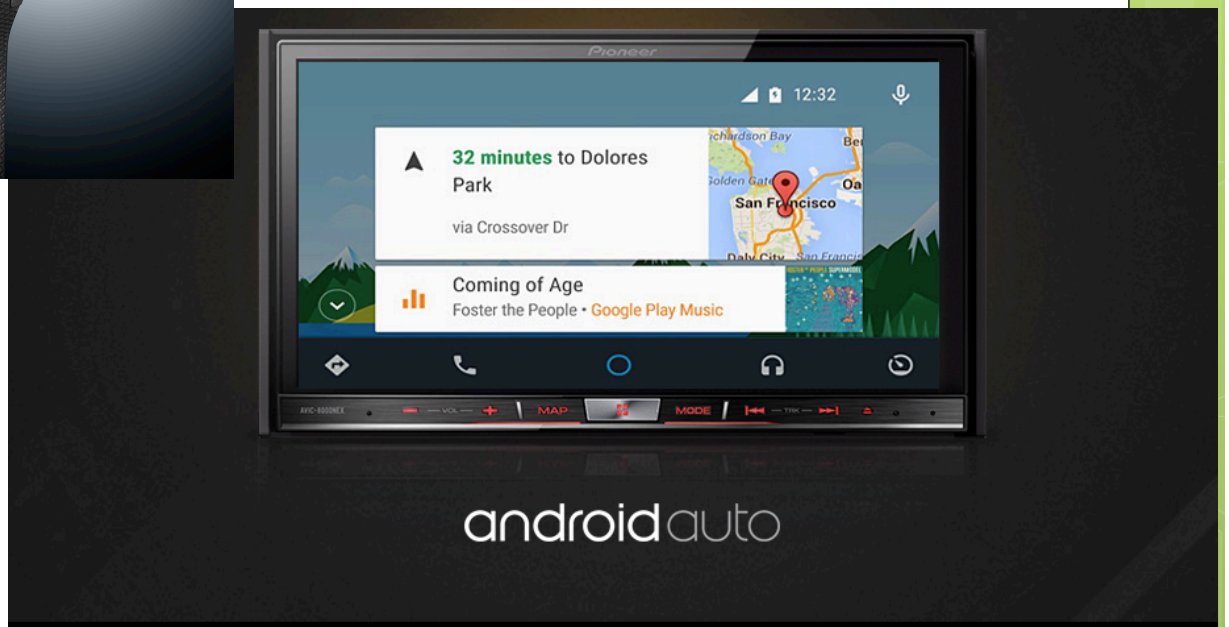
Functionalities by version

- 1.5 Integrazione con servizi Google
- 1.6 Sintesi vocale, ricerca vocale, gestures
- 2.0 Miglior supporto videocamera, multitouch
- 2.2 Migliori prestazioni. Open GL ES2.0, Javascript e Flash. Tethering. Installazione apps su SD
- 2.3 Video chat in GoogleTalk. UI migliorata, Download manager
- 3.0 Ottimizzata per tablet. Aggiunta la barra di sistema e Action Bar. Possibilità di criptare tutti i dati personali.
- 3.1 Supporto per le periferiche USB
- 4.0 UI completamente riprogettata. Prestazioni migliorate. Dettatura real time. Face Unlock. Fotocamera migliorata. "Contatti" con integrazione con i social network
- 5.0 Cambiamento grafica e animazione, high performance graphics, migliore efficienza (ART), 64 bit, migliori notifiche, Android TV, battery stats, tilt & heart rate sensors, migliore audio e camera

Functionalities by version

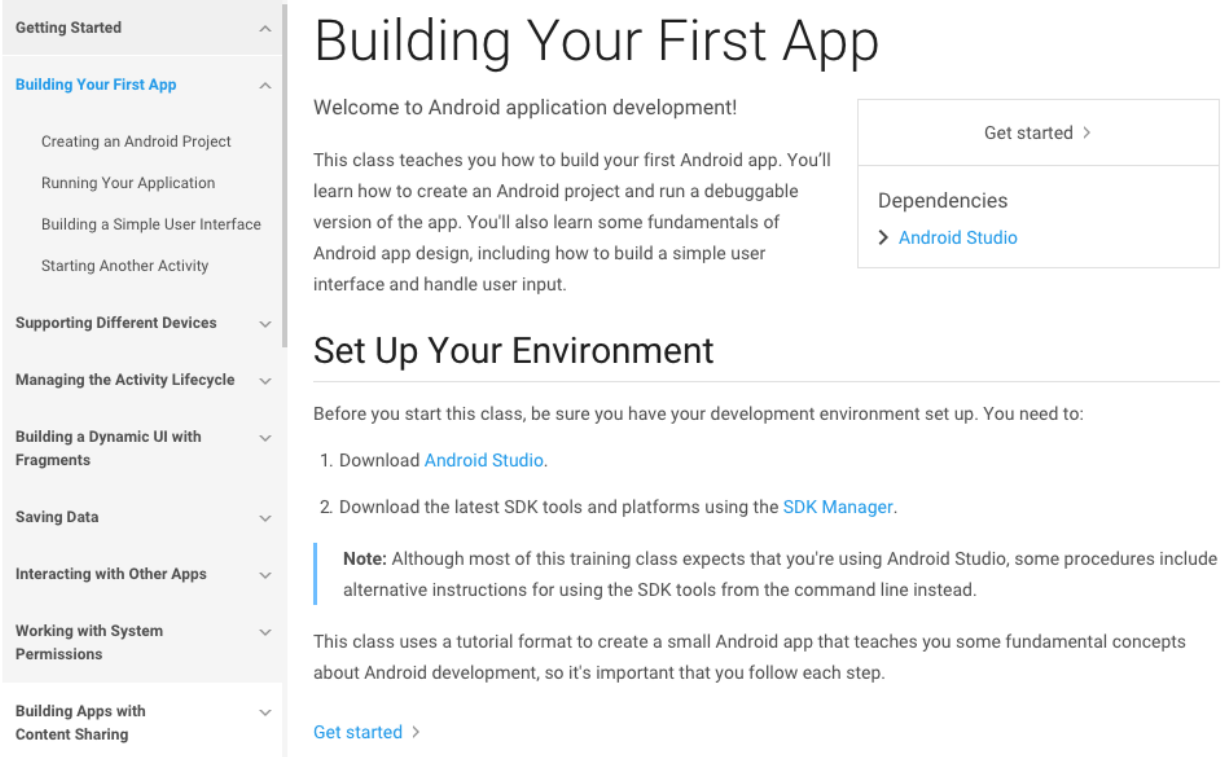
- 6.0 Supporto telefono Android wear, multiscreen, nuovo security model
- 7.0 Multiwindow
- 8.0 Java 8 API, Neural Network API, Media improvements (PIP)
- 9.0 Multi-Camera Support
  - IEEE 802.11mc Wi-Fi protocol—also known as Wi-Fi Round-Trip-Time (RTT)—to let you take advantage of indoor positioning in your apps.

Android wear – Android auto



Android Auto for the Vehicle You Already Own

Hands on?



Getting Started

Building Your First App

- Creating an Android Project
- Running Your Application
- Building a Simple User Interface
- Starting Another Activity

Supporting Different Devices

Managing the Activity Lifecycle

Building a Dynamic UI with Fragments

Saving Data

Interacting with Other Apps

Working with System Permissions

Building Apps with Content Sharing

Building Your First App

Welcome to Android application development!

This class teaches you how to build your first Android app. You'll learn how to create an Android project and run a debuggable version of the app. You'll also learn some fundamentals of Android app design, including how to build a simple user interface and handle user input.

Get started >

Dependencies

> Android Studio

Set Up Your Environment

Before you start this class, be sure you have your development environment set up. You need to:

1. Download Android Studio.
2. Download the latest SDK tools and platforms using the SDK Manager.

> **Note:** Although most of this training class expects that you're using Android Studio, some procedures include alternative instructions for using the SDK tools from the command line instead.

This class uses a tutorial format to create a small Android app that teaches you some fundamental concepts about Android development, so it's important that you follow each step.

Get started >

http://developer.android.com/training/basics/firstapp/index.html

Si, ma…

MainActivity.java - FirstApplication - [~/AndroidStudioPr

app

FirstApplication > app > src > main > java > it > unitn > latemar > firstapplication > MainActivity

Android

- app
  - manifests
    - AndroidManifest.xml
  - java
    - it.unitn.latemar.firstapplication
      - MainActivity
    - it.unitn.latemar.firstapplication (androidTest)
  - res
    - drawable
    - layout
      - activity_main.xml
      - content_main.xml
    - menu
    - mipmap
    - values
      - colors.xml
      - dimens.xml (2)
      - strings.xml
      - styles.xml (2)
- Gradle Scripts

MainActivity.java | content_main.xml | activity_main.xml

```
package it.unitn.latemar.firstapplication

import ...

public class MainActivity extends AppComp

    @Override
    protected void onCreate(Bundle savedI
        super.onCreate(savedInstanceState
        setContentView(R.layout.activity_
        Toolbar toolbar = (Toolbar) findV
        setSupportActionBar(toolbar);

        FloatingActionButton fab = (Float
        fab.setOnClickListener((view) → {
                Snackbar.make(view, "Repl
                        .setAction("Actio
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Me
        // Inflate the menu; this adds it
        getMenuInflater().inflate(R.menu.
        return true;
    }

    @Override
    public boolean onOptionsItemSelected(
        // Handle action bar item clicks
        // automatically handle clicks on
        // as you specify a parent activi
        int id = item.getItemId();
```

8. Click the **Finish** button to create the project.

Your Android project is now a basic "Hello World" app that contains some default files. Take a moment to review the most important of these:

`app/src/main/res/layout/activity_my.xml`

This XML layout file is for the activity you added when you created the
Following the New Project workflow, Android Studio presents this file v
of the screen UI. The file contains some default interface elements fro
including the app bar and a floating action button. It also includes a se
content.

`app/src/main/res/layout/content_my.xml`

This XML layout file resides in `activity_my.xml`, and contains some
that displays the message, "Hello world!".

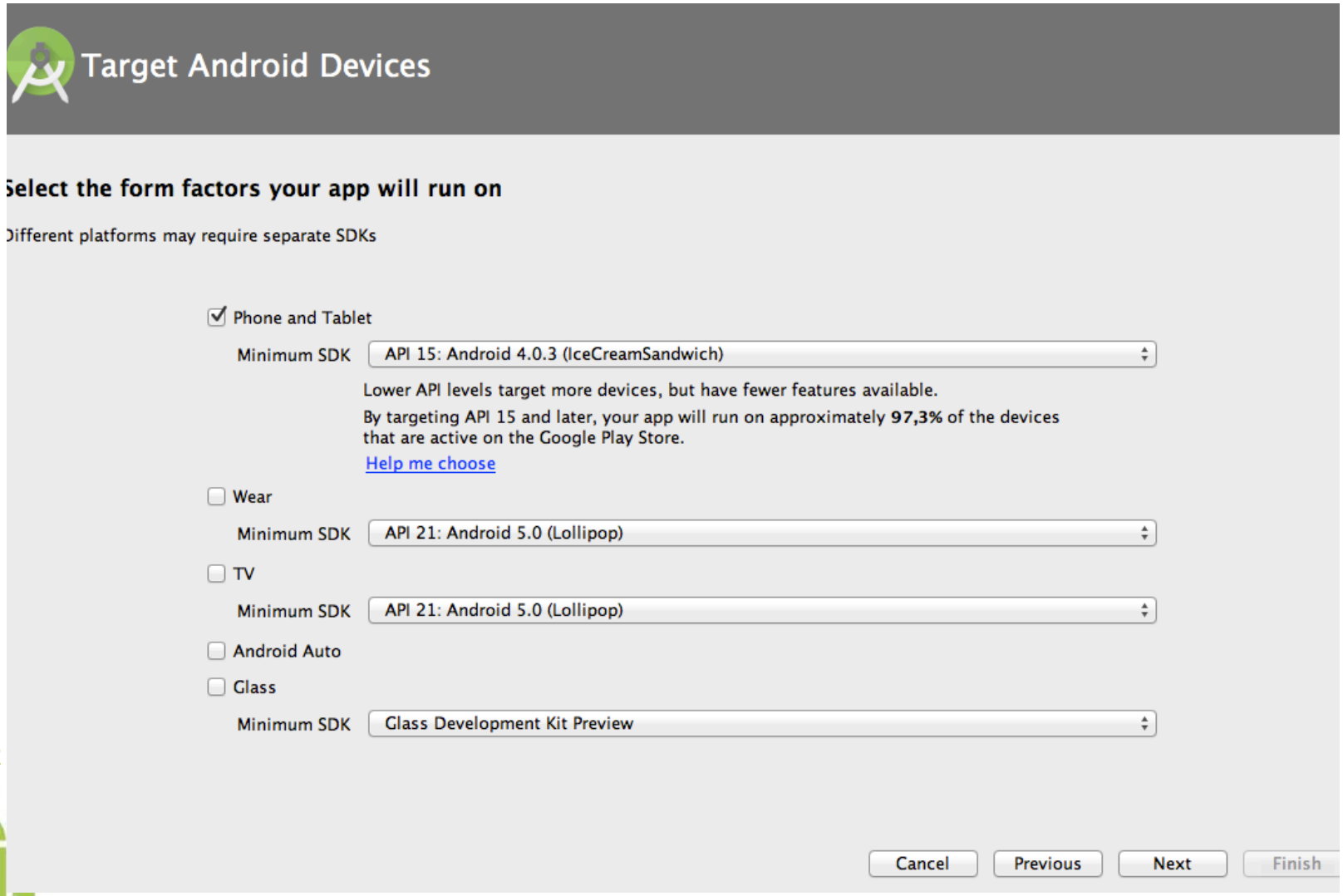
`app/src/main/java/com.mycompany.myfirstapp/MyActivity.java`

A tab for this file appears in Android Studio when the New Project work
file you see the class definition for the activity you created. When you b
class starts the activity and loads the layout file that says "Hello World!

`app/src/main/AndroidManifest.xml`

The manifest file describes the fundamental characteristics of the app
components. You'll revisit this file as you follow these lessons and add

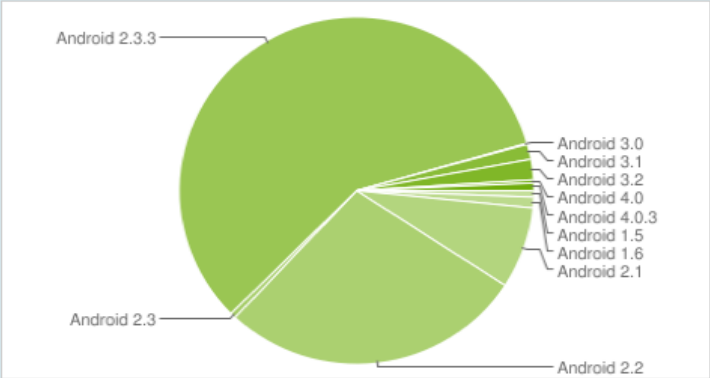
Target devices (Android Studio)



Android version distribution Feb 2012

http://developer.android.com/resources/dashboard/platform-versions.html

The following pie chart and table is based on the number of Android devices that have accessed Android Market within a 14-day period ending on the data collection date noted below.



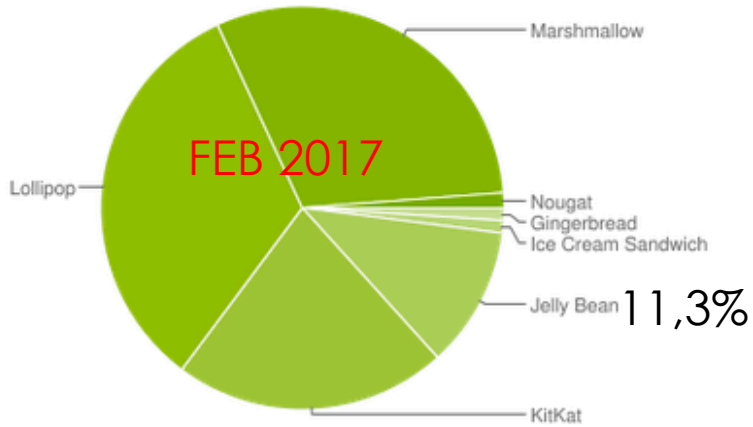
Platform	Codename	API Level	Distribution
Android 1.5	Cupcake	3	0.6%
Android 1.6	Donut	4	1.0%
Android 2.1	Eclair	7	7.6%
Android 2.2	Froyo	8	27.8%
Android 2.3 - Android 2.3.2	Gingerbread	9	0.5%
Android 2.3.3 - Android 2.3.7		10	58.1%
Android 3.0	Honeycomb	11	0.1%
Android 3.1		12	1.4%
Android 3.2		13	1.9%
Android 4.0 - Android 4.0.2	Ice Cream Sandwich	14	0.3%
Android 4.0.3		15	0.7%

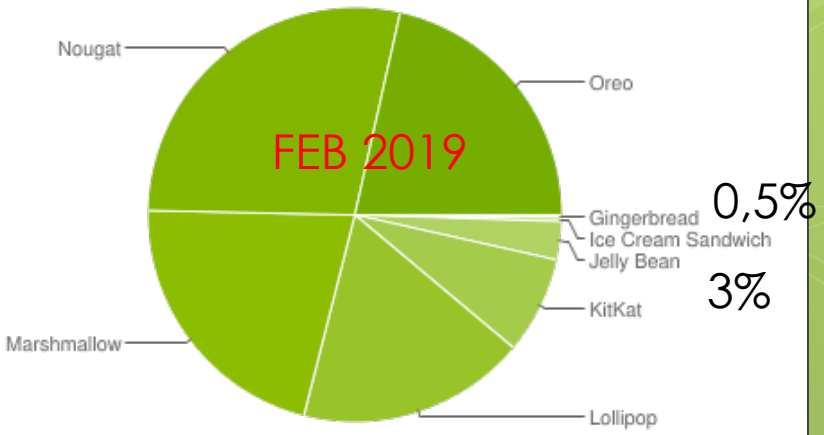
Data collected during a 14-day period ending on February 1, 2012

Android version distribution – Feb 2017

http://developer.android.com/resources/dashboard/platform-versions.html

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	1.0%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	1.0%
4.1.x	Jelly Bean	16	4.0%
4.2.x	(2011)	17	5.7%
4.3		18	1.6%
4.4	KitKat	19	21.9%
5.0	Lollipop	21	9.8%
5.1		22	23.1%
6.0	Marshmallow	23	30.7%
7.0	Nougat	24	0.9%
7.1		25	0.3%





Data collected during a 7-day period ending on February 6, 2017.

Any versions with less than 0.1% distribution are not shown.

Android Java packages

Marco Ronchetti
Università degli Studi di Trento

Basic components

android.app

- implements the Application model for Android

android.content

- implements the concept of Content providers

android content.pm

- Package manager: permissions, installed {packages, services, provider, applications, components}

android.content.res

- Access to resources

android.provider

- Contacts, MediaStore, Browser, Setting

GUI basics

android.view

- Menu, View, ViewGroup + listeners

android.view.animation

android.view.inputmethod

- Input methods framework

android.widget

- UI controls derived from View (Button, Checkbox…)

android.gesture

- create, recognize, load and save gestures

Graphics

android.graphics

- low level graphics tools such as canvases, color filters, points, and rectangles that let you handle drawing to the screen directly.
- Bitmap, Canvas, Camera (3D transformation, not the camera!) , Color, Matrix, Movie, Paint, Path, Rasterizer, Shader, SweepGradient, Typeface

android.graphics.drawable

- variety of visual elements that are intended for display only, such as bitmaps and gradients

android.graphics.drawable.shapes

android.opengl

- opengl-related utility classes, not the opengl!

javax.microedition.khronos.opengles

javax.microedition.khronos.egl

javax.microedition.khronos.nio

android.renderscript

- low-level, high performance means of carrying out mathematical calculations and 3D graphics rendering

Text rendering

android.text

- classes used to render or track text and text spans on the screen

android.text.method

- Classes that monitor or modify keypad input.

android.text.style

- Text styling mechanisms

android.service.textservice

- Provides classes that allow you to create spell checkers

android.view.textservice

- Use spelling checkers

Database, Web and location

android.database

- classes to explore data returned through a content provider.

android.datebase.sqlite

- the SQLite database management classes that an application would use to manage its own private database. Applications use these classes to manage private databases.

android.webkit

- tools for browsing the web.

android.location

- Address, Geocoder, Location, LocationManager, LocationProvider

com.google.android.maps

Network and telephony

android.net

- Socket-level network API - help with network access, beyond the normal java.net.* APIs.

android.net.wifi

android.bluetooth

android.nfc

- Near Field Communication (NFC) is a set of short-range wireless technologies, typically requiring a distance of 4cm or less to initiate a connection. NFC allows you to share small payloads of data between an NFC tag and an Android-powered device, or between two Android-powered devices.

android.telephony

- monitoring the basic phone information, plus utilities for manipulating phone number strings, SMS
- CellLocation, PhoneNumberUtils, TelephonyManager

android.telephony.gsm

- Obtain Cell location of GSM

android.telephony.cdma

- Obtain Cell location of CDMA - CDMA2000 is a family of 3G mobile technology standards

Media and speech

android.media

- manage various media interfaces in audio and video
- MediaPlayer, MediaRecorder, Ringtone, AudioManager, FaceDetector.

android.media.effect

- apply a variety of visual effects to images and videos

android.hardware

- support for hardware features, such as the camera and other sensors

android.drm

- Digital right management

android.mtp

- interact directly with connected cameras and other devices, using the PTP (Picture Transfer Protocol)

android.speech

- base class for recognition service implementations

android.speech.tts

- Text to Speech

General utilities

android.utils

- date/time manipulation, base64 encoders and decoders, string and number conversion methods, and XML utilities.

android.sax

- XML parsing

android.test

- A framework for writing Android test cases and suites

android.preference

- manage application preferences and implement the preferences UI. Using these ensures that all the preferences within each application are maintained in the same manner and the user experience is consistent with that of the system and other applications

android.os

- basic operating system services, message passing, and inter-process communication
- Binder (ipc), FileObserver (changes in files) Handler e Looper (for dealing with message threads), BatteryManager, PowerManager

Still useful java packages

java.lang (e subpackages)

java.math

java.net + javax.net

java.io

java.nio

java.sql+javax.sql

- (android.database preferable if possible)

java.util

Other still useful packages

javax.crypto
javax.security
javax.xml

org.w3c.dom
org.xml.sax

org.apache.http (e subpackages)

Introducton to Applications

Marco Ronchetti
Università degli Studi di Trento

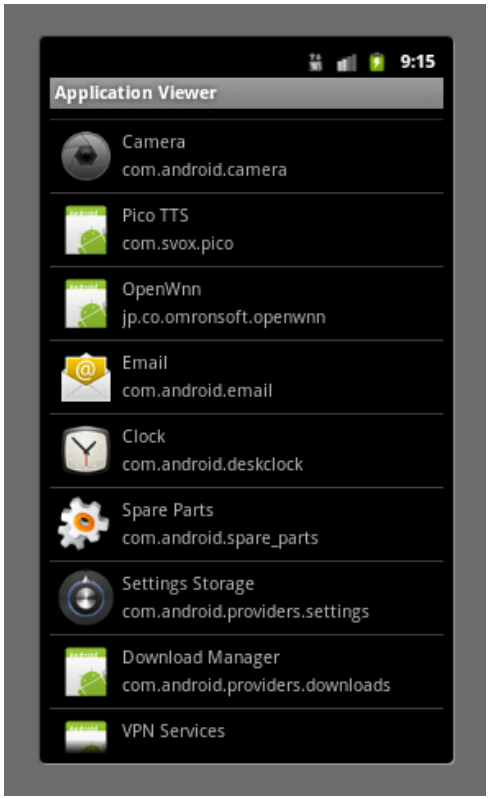
An Android *application* typically consists of one or more related, loosely bound activities for the user to interact with.

Android has an application launcher available at the Home screen, typically in a sliding drawer which displays applications as icons, which the user can pick to start an application.

Android ships with a rich set of applications that may include email, calendar, browser, maps, text messaging, contacts, camera, dialer, music player, settings and others.

Application Launcher

You can replace it



See e.g. http://xjaphx.wordpress.com/2011/06/12/create-application-launcher-as-a-list/

Application package

An application is a single APK (application package) file. An APK file roughly has three main components.

- Dalvik executable: all your Java source code compiled down to Dalvik executable. This is the code that runs your application.
- Resources: everything that is not code (images, audio/video clips, XML files describing layouts, language packs, and so on.
- Native libraries: e.g. C/C++ libraries.

Signing applications

Android applications must be signed before they can be installed on a device

To distribute your application commercially, you'll want to sign it with your own key.

Distributing applications

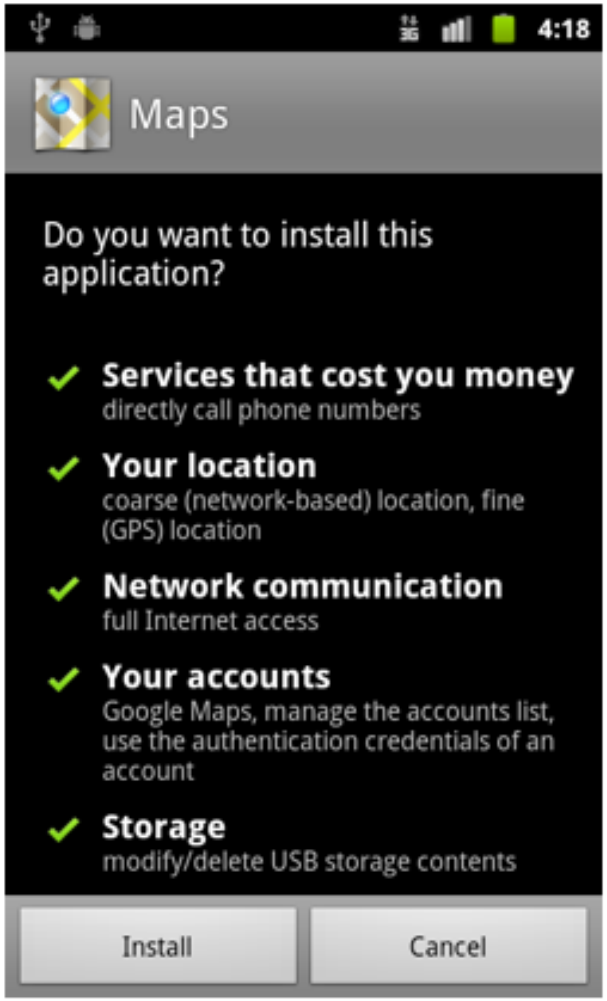
Unlike the iPhone, on Android, there can be many different Android stores or markets. Each one can have its own set of policies with respect to what is allowed, how the revenue is split, and so on.

The biggest market currently is Android Market run by Google

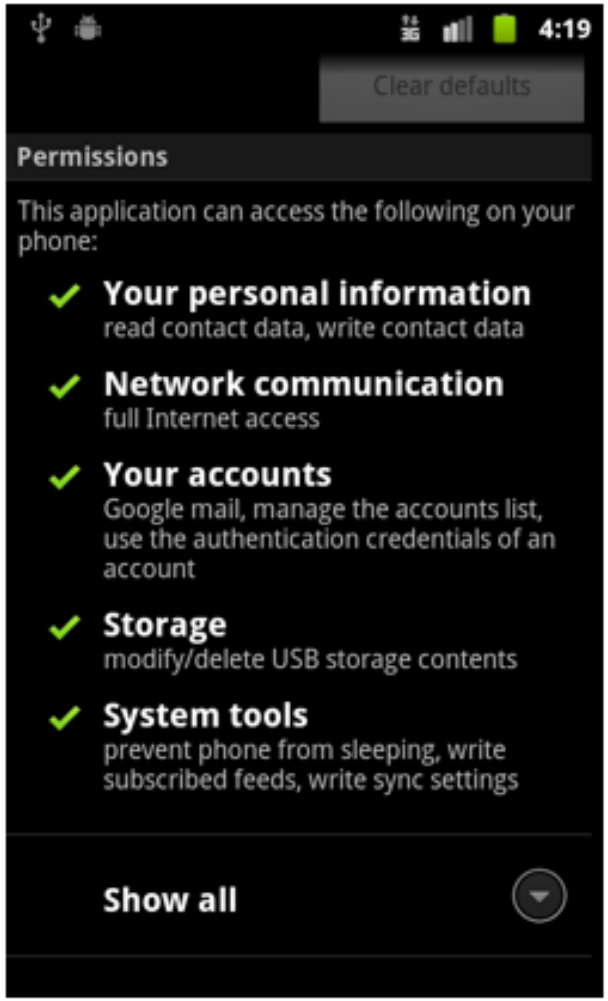
Applications can also be distributed via the web. When you download an APK file from a website by using the Browser, the application represented by the APK file automatically gets installed on your phone.

Granting and checking permissions

Permissions at Application Install -- Google Maps



Permissions of an Installed Application -- gMail



Impostazioni->Altro->Gestione Applicazioni -> …

Security

Android has a security framework.

http://source.android.com/devices/tech/security/index.html

The Android File System can be encrypted.

Encryption on Android uses the dm-crypt layer in the Linux kernel.

Security model

Android OS is a multi-user Linux in which each application is a different user.

By default, the system assigns each application a unique Linux user ID (the ID is unknown to the application). The system sets permissions for all the files in an application so that only the user ID assigned to that application can access them.

Each process has its own virtual machine (VM), so an application's code runs in isolation from other applications.

By default, every application runs in its own Linux process.

Principle of least privilege

Principle of least privilege (or "need to know")

Each application, by default, has access only to the components that it requires to do its work and no more.

A variation of "information hiding", or "Parnas' principle".

Data sharing

It's possible to arrange for two applications to share the same Linux user ID, in which case they are able to access each other's files.

Applications with the same user ID can also arrange to run in the same Linux process and share the same VM (the applications must also be signed with the same certificate).

An application can request permission to access device data such as the user's contacts, SMS messages, the mountable storage (SD card), camera, Bluetooth, and more. All application permissions must be granted by the user at install time.

Process lifetime

Android

- starts the process when any of the application's components need to be executed,
- shuts down the process when
  - it's no longer needed
  - the system must recover memory for other applications.