MobServ Notes

Andrea Palmieri

1. Mobile app landscape

APPLICATION = APPLICATION SERVICES (COMPUTE,STORAGE)+MANAGEMENT SERVICES(NETWORK,DB)

- **Native app** = written specifically for some platform (Java, Swift). (+) performances, UX and security (-) users limited to specific platform, high cost and long time of development
- **Web app** = Written using web standards and compatible across platforms. (+) Good UX, low cost of development, maximize users and quick to develop (-) not high quality and security relies on browser
- **Hybrid app** = Write once, compiled everywhere using webkit to link with specific API. (+) Large user base, low costs and quick to develop (-) Bad UX and quality

Revenues approaches: Paid, in-app purchase, advertising, freemium **Context aware**: User, Device,Time/space and environment to make the app aware of the context and improve the experience.

2. Android

Based on custom Linux OS, including new libraries, custom VM and java application framework.

Development phases=Setup, develop, debug and test (iterating) and publish.

Features: Application framework (reuse of components), integrated browser, 2D and 3D graphics libraries, media support, connectivity, sensors..

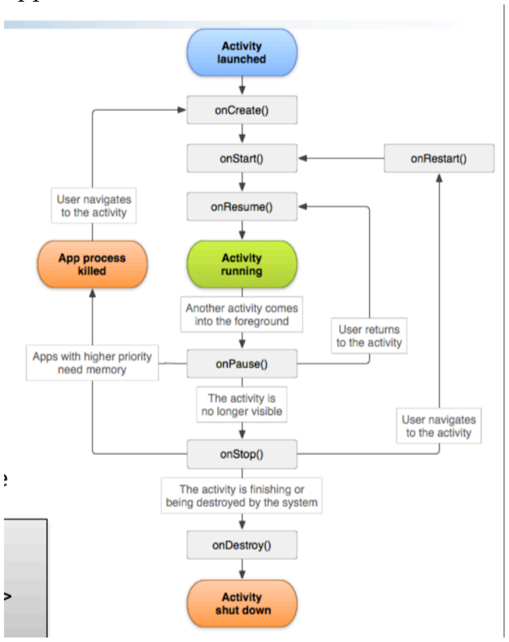
Android stack 1. Apps (system and not), each in its sandbox 2. Java API framework = Content provider, view system and Managers (Activity manager, package manager, location manager, notification manager…) 3. Native C/C++ Libraries (SQLite, Libc..) + Android runtime ART, every app has its own process and instance of Dalvik VM. 4. Hardware abstraction layer (audio, camera, bluetooth..) 5. Kernel (drivers)

Android components

- Activities: object which has a life cycle and perform some actions. It provides a screen (view) for the user interaction. Task is a collection of activities that user interact with when performing a certain job (from home to LIFO)
- Services: Component without UI used in the background (ex. media player). It can be part of its own process or in the context of another application's process.
- Content Providers: Manages access to data on the device and share data between applications (ex. list of contacts). Use of unique URI to identify data in a provider.
- Broadcast receivers: a component that responds to system-wide broadcast (even handler) initiated by other application or by the system (ex. battery is fully charged).

Other important stuff

- Views: object that knows how to draw itself on the screen.
- Fragments are used for a more dynamic and flexible UI designs. Represents a behavior or a portion of user interface in an Activity. A fragment must always be embedded in an activity and the fragment's lifecycle is directly affected by the host activity's lifecycle.
- Intent: used to start activities, *"intention to do something"*. Can be **implicit** (specify an action) or **explicit** (specify a target). Data can be added to the intent using `setExtra`. In the manifest, every app can declare the intent that can be handled using *intent filters*.


Activity launched
onCreate()
onStart()
onRestart()
User navigates to the activity
onResume()
App process killed
Activity running
Another activity comes into the foreground
User returns to the activity
Apps with higher priority need memory
onPause()
The activity is no longer visible
User navigates to the activity
onStop()
The activity is finishing or being destroyed by the system
onDestroy()
Activity shut down

Android market

- While mobile Internet requests are growing rapidly worldwide, there are regional differences in the devices used
- Smartphones with touchscreens, WiFi connectivity, and advanced Operating Systems are growing in popularity
- Mobile Internet Devices like tablets and gaming devices are changing the way the mobile Internet is accessed
- Apple seems to be the top device manufacturer driven by the worldwide adoption of iOSdevices
- The launch of new Android devices over the past years has led to rapid growth in the Android platform

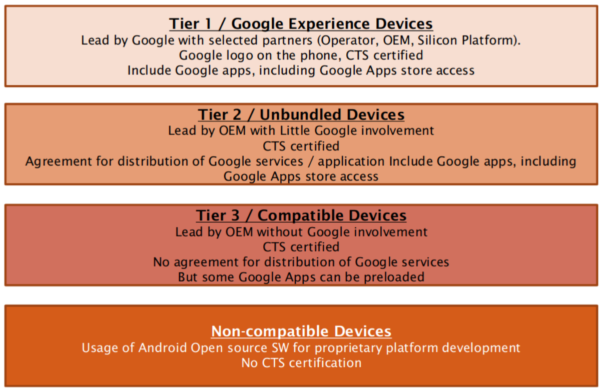
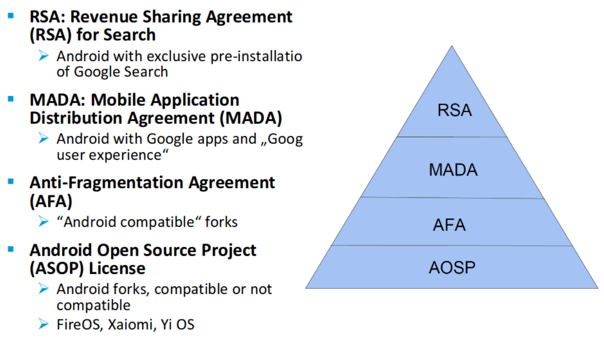

Tier 1 / Google Experience Devices
Lead by Google with selected partners (Operator, OEM, Silicon Platform).
Google logo on the phone, CTS certified
Include Google apps, including Google Apps store access
Tier 2 / Unbundled Devices
Lead by OEM with Little Google involvement
CTS certified
Agreement for distribution of Google services / application Include Google apps, including Google Apps store access
Tier 3 / Compatible Devices
Lead by OEM without Google involvement
CTS certified
No agreement for distribution of Google services
But some Google Apps can be preloaded
Non-compatible Devices
Usage of Android Open source SW for proprietary platform development
No CTS certification

Figure 1: Google android device category

Is Android open?

- Android is fully open for the SW developer ecosystem but completely closed for the handset OEM ecosystem
- Android governance model is an elaborate set of control points, allowing Google to bundle its own services and control the HW & SW on every handset:
  - Private branches available to select partners
  - Closed review process. Google is the only authority to accept or reject a code submission from the community
  - Speed of evolution. Google dictates high speed innovations.
  - Gated developer community. Android market is exclusive distribution channel for the apps, only available on exclusive license agreement
  - Anti-fragmentations agreement signed by OHA members committing not to release non CTS compliant handsets
  - Private roadmap only available to main partners
  - Android Trademark. Android branding is fully controlled by Google

Mobile design



- **RSA: Revenue Sharing Agreement (RSA) for Search**
  - Android with exclusive pre-installatio of Google Search
- **MADA: Mobile Application Distribution Agreement (MADA)**
  - Android with Google apps and „Goog user experience“
- **Anti-Fragmentation Agreement (AFA)**
  - “Android compatible“ forks
- **Android Open Source Project (ASOP) License**
  - Android forks, compatible or not compatible
  - FireOS, Xaiomi, Yi OS

Figure 2: Android pyramid

Remember: User Device (UD), User interface (UI), User Apps (UA), User Experience (UX). Design is not just what it looks like and feel like. Design is how it works: Users' expectations and common conventions are directly translated to Value and Trust.

User interface

Gives you access to computing capabilities by abstracting the machine. Allows user interact with a machine: Input $\rightarrow$ system $\rightarrow$ output

Natural user interface (NUI)

Invisible UI helping human to continuously learn complex interaction. Intuitive UI usable without learning. Applicable to wearable devices as well as smartphones/tablets.

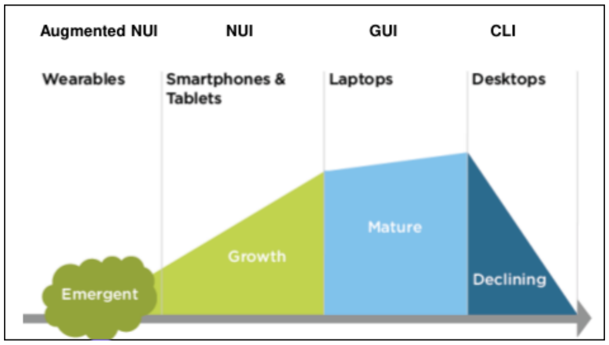


Figure 3: A-NUI, NUI, GUI, CLI

Content first then navigation

Lots of time spent for navigation to get the content

- Speed matters : how long does it take to see the content

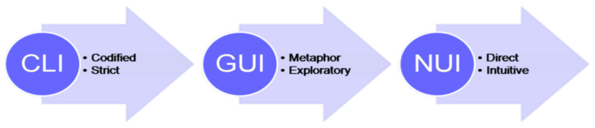


Figure 4: Natural User Interface

- Space matters : save screen by minimizing the tabs/menu & navigation bar

The content itself could serve as the interface - Interact with guessable, physical, and realistic gestures - Make use of skills learned through a lifetime of living

Example of Flickr regarding *screen size*:
Flickr's mobile web experience takes 60 plus navigation options down to six. They focused on the key actions performed by users. You need to know your audience and prioritize what really matters for them.

Performance

Connection is not always fast, so you need to reduce request and file size (speed is not important just on mobile). Take advantage of new technologies. Design choices impact application performance.

Context

Time and place play an important role in how mobile phone are used, for example in the evening, people do not want to use their smartphones for work. It is unlikely to get someone's full attention: partial attention or a short burst.

Capabilities

Example: location detection, orientation, audio, proximity, ecc...
Reinvent ways to meet people's needs!!! For example, augumented reality to navigate space around you.

Smart app

Awareness of user environment by adding relevant information on the present as a function of user preferences/profile:

- User becomes part of the experience
- User experiences a better decision-making

My environment (physical location/social context) may influence my actions. Context is what make mobile device a powerful medium.

You cannot use the same ideas directly from the desktop UI world. You need to have a mobile mindset. Achieve great performance through appropriate design:

- new interaction paradigm

- quick use, quick storage
- present useful information quickly
- keep it simple, clear and precise
- address a specific need
- one at a time please
- no background applications

Interaction type:

- Lookup/Find : urgent local info
- Explore/Play : local actions to pass time
- Check In/Status : repeatable important micro-tasking
- Edit/Create : one-shot urgent micro-tasking

Content over navigation

As a general rule, content takes precedence over navigation on mobile:

- immediate access to content and not the site map
- adjust the structure accordingly to the app usage /audience

Input

- Text input has to be minimized
- Designing interfaces that require less attention in dynamic environments
- Hardware buttons have some significant advantages on software (touch screen) buttons when it comes to eyes-free interaction
- Place interactive elements at bottom of screen
- Users should get enough feedback from the device
- Good defaults
- Auto-capitalize, auto-completion, and predictive text entry
- Alternate input methods (QR code, speech, images)
- gesture-based or shake-based text entry methods

Visual design

Good design matters:

1. Users look at multiple screenshots
2. Users read the reviews, check stars and download rate
3. Users judge the app based on the quality of its icon
4. User evaluate the app for 20-50 seconds

Visual design elements: messaging and branding, look and feel, layout, color, typography, graphics, logo.

Visual design guidelines

- Use visual design to deliver your message and use branding to reinforce the message
- Use context and user preferences to adapt look and feel and evoke user actions

- Start building layout early (Different layout for different devices)
- Know your screen before dealing with the colors and font type
- Know the physiology and culture of colors
- Appropriate font type and size improve the readability
- Use graphics to establish a visual experience

Information architecture

How the information/content is structured and shaped and how users interact with it through different devices. How intuitive is to find information and perform tasks.

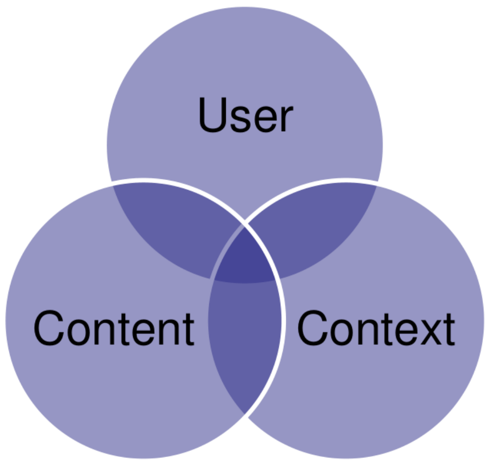


Figure 5: User, content, context

Information Architecture: Usability

- Ease of learning: faster the second time and so on.
- Recall: remember how from one session to the next
- Productivity: perform tasks quickly and efficiently
- Minimal error rates: if they occur, good feedback so user can recover
- High user satisfaction: confident of success

Information Architecture: Findability Ease with which information can be found: UI design, accessibility, search engine. Evaluated through: tree testing and usability testing.

Importance of logo and a good app name

- Law of subtraction: FedEx
- Law of Representation: Tour de France
- Law of association/reminder: Amazon
- Law of avant-garde: Apple logo

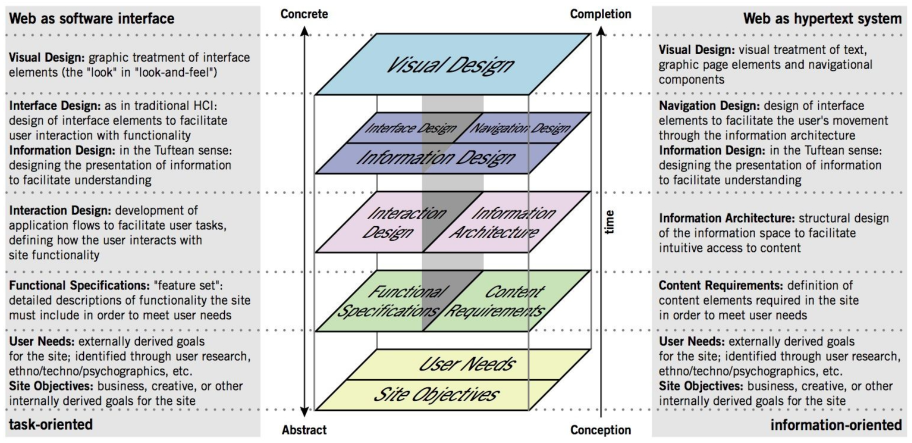

Web as software interface
Visual Design: graphic treatment of interface elements (the "look" in "look-and-feel")
Interface Design: as in traditional HCI: design of interface elements to facilitate user interaction with functionality
Information Design: in the Tuftean sense: designing the presentation of information to facilitate understanding
Interaction Design: development of application flows to facilitate user tasks, defining how the user interacts with site functionality
Functional Specifications: "feature set": detailed descriptions of functionality the site must include in order to meet user needs
User Needs: externally derived goals for the site; identified through user research, ethno/techno/psychographics, etc.
Site Objectives: business, creative, or other internally derived goals for the site
task-oriented
Concrete
Abstract
Visual Design
Interface Design
Navigation Design
Information Design
Interaction Design
Information Architecture
Functional Specifications
Content Requirements
User Needs
Site Objectives
Completion
time
Conception
Web as hypertext system
Visual Design: visual treatment of text, graphic page elements and navigational components
Navigation Design: design of interface elements to facilitate the user's movement through the information architecture
Information Design: in the Tuftean sense: designing the presentation of information to facilitate understanding
Information Architecture: structural design of the information space to facilitate intuitive access to content
Content Requirements: definition of content elements required in the site in order to meet user needs
User Needs: externally derived goals for the site; identified through user research, ethno/techno/psychographics, etc.
Site Objectives: business, creative, or other internally derived goals for the site
information-oriented

Figure 6: User, content, context

4. Golden rules

1. Decide what to build focusing on the user's needs.
2. Visit app stores to get insights of competitors. What can be improved?
3. Explore possible solutions, test and improve. Focus on targeted user base.
4. Sketch, mock-up, stories, interviews to gather feedbacks from users. The more you do, the better it is.
5. Build a basic prototype and use feedbacks.
6. Iterate the process.
7. Start coding from the interface, than backend -> Top-down approach.
8. Beta test the app and study the usability -> test before submit
9. Release and fix the bug.

5. Android and Machine Learning

Common use cases: recognizing text, detecting faces, identifying landmarks, scanning barcodes, labeling images. On-device vs.in the cloud

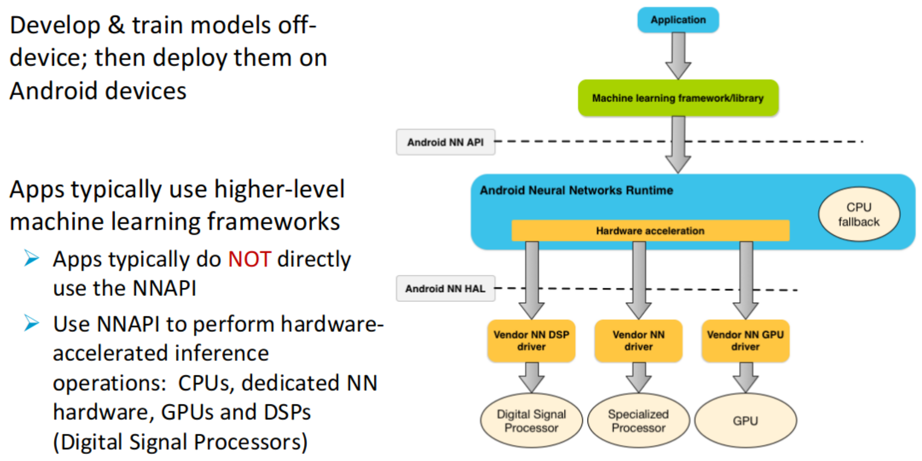
- On-device APIs: can process data quickly, not general-use CPU alone; No N/W latency, no need for network connection, no privacy / data sensitivity consideration, Cheap computational cost on device
- Cloud-based APIs: leverage greater processing power, Google Cloud Platform's machine learning technology accuracy

You can use one of the following:

1. Android Neural Networks API (NNAPI)
2. ML Kit for Firebase: mobile SDK by google

What is TensorFlow?
Open-source software library for dataflow programming. Also used for ML such as neural networks. Easy to train with CPU and GPU distributed computing. Parallel NN training: train multiple NNs and multiple GPUs. Large community + Open Source

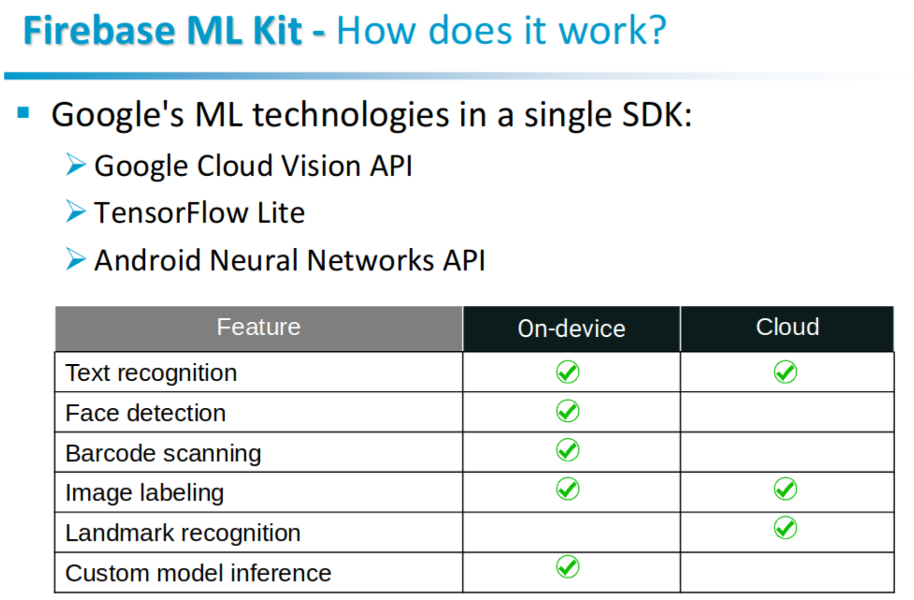
Use cases: Voice/Sound Recognition, Text Based Applications, Image Recognition...


Develop & train models off-device; then deploy them on Android devices

Apps typically use higher-level machine learning frameworks

- Apps typically do NOT directly use the NNAPI
- Use NNAPI to perform hardware-accelerated inference operations: CPUs, dedicated NN hardware, GPUs and DSPs (Digital Signal Processors)

Figure 7: Neural Network API



Firebase ML Kit - How does it work?

- Google's ML technologies in a single SDK:
  - Google Cloud Vision API
  - TensorFlow Lite
  - Android Neural Networks API

Feature	On-device	Cloud
Text recognition	✓	✓
Face detection	✓	
Barcode scanning	✓	
Image labeling	✓	✓
Landmark recognition		✓
Custom model inference	✓	

Figure 8: Firebase ML Kit