MOBILE APPLICATION DEVELOPMENT

By Ankit Attkan
Department of Computer Engineering
National Institute of Technology, Kurukshetra

NATIVE APPS

- Native apps are installed onto the device itself and are developed especially for a particular mobile operating system.
- These apps are available on app stores such as Apple App Store, Google Play Store, etc.
- If an app made for Android OS then it will not work on Apple iOS or Windows OS.
- Need to build separate apps for each operating system to deploy app across all major operating systems.
- This means we have to spend more money and more effort (time, resources).

NATIVE APPS

- **Pros**
  - Native can be used offline, which makes them faster to open and access anytime.
  - In some cases, the performance is faster because they store information (data) locally and only synchronize with the server after the user is done using the app.
  - They allow the user to use device-specific hand gestures. Android and iOS are gradually developing different conventions for interaction, and a native app responds the way its user expects.
  - Native apps get the approval of the app store they are intended for, which means most of the time the user can be assured of improved safety and security of the app.
  - They allow direct access to device hardware that is either more difficult or impossible with a mobile app (camera, accelerometer, etc.)

NATIVE APPS

- **Cons**
  - More expensive to develop, especially when the app needs to be compatible with multiple mobile operating systems, thus multiplying the development costs.
  - Cost of app maintenance is higher (especially if this app supports more than one mobile platform).
  - Getting the app approved for the various app stores can prove to be long and tedious for the developer
  - Use of the app is contingent on the user's willingness to download and install the app onto their mobile device

WEB APPS

- Web Apps are basically internet-enabled applications that are accessible via the mobile device's Web browser.
- Users don't need to downloaded and install the app onto mobile device in order to access it.
- The app is written as web pages in HTML and CSS, with the interactive parts in Jquery, JavaScript or similar language.
- Meaning that single web app can be used on most devices capable of surfing the web, regardless of the operating system they use.

WEB APPS

- **Pros**
  - They are instantly accessible to users via a browser across a range of devices (iPhone, Android, Windows, etc.).
  - They are much easier to update or maintain by the developer. If you want to change the design or content of a mobile web app, you simply publish the update to the server and the changes are immediately visible.
  - They are much easier for users to discover since their pages can be displayed in search results and listed in common search engines such as Google or Bing.
  - Visitors to your regular website can be automatically sent to the mobile web app when they are on a handheld mobile device (using device detection).
  - Just like a standard website, mobile websites / web app can be developed as database-driven web applications that act very much like native apps.
  - The development is considerably more time and cost-effective than development of a native app, using programming languages and technologies that are more commonly understood and have a much larger developer base.

WEB APPS

- **Cons**
  - Mobile Web apps only have limited scope as far as accessing a mobile device's features is concerned (device-specific hand gestures, sensors, etc.).
  - There are so many variations between web browsers and browser versions and phones that it makes it challenging to develop a stable web-app that runs on all devices without any issues.
  - They are not listed in 'App Stores'. So if someone is looking for your app in the app store, they will be unable to discover it though such means.
  - Since there is no regularised quality control system for Web apps, users may not always be guaranteed safety and security of the app.
  - Web apps are unavailable when offline, even as a basic version.

NATIVE VS WEB APPS

Native Apps	Web Apps
Mobile apps are developed for a specific platform, such as iOS for the Apple iPhone or Android	Web Apps are accessed via the internet browser and will function according to the device you're viewing them on
They are downloaded and installed via an Application store such as Google Play, Apple Store, etc. and have access to system resources, such as GPS and camera of the device.	Web apps are not native to a particular system and there is no need to be downloaded or installed.
Mobile apps may work offline.	Web apps need an active internet connection.
Native Apps are comparatively faster.	Web Apps are comparatively slower.
It is difficult to have a native mobile app approved by Application Store.	Application store approval is not required, so web apps can be launched easily.
Native apps have more safety and security.	Web apps have comparatively low security.
Maintaining and consistently update of native apps cause more cost.	These apps can be set to update themselves or automatically.

MOBILE SOFTWARE ENGINEERING

- It is the adaptive process of creating a software product that is intended to be used by people while they are mobile (on move).
- The process must take into account the specific characteristics of mobile usage to deliver the required quality.
- Characteristics like
  - Mobility (constant change of position)
  - Device capabilities (CPU, Less Memory, Limited battery, Small screen size).
  - Operator plan and communication costs
  - Interaction possibilities/ user experience: Multipoint-touch, Gestures and motion detection, Sensors (acceleration, tilt, GPS, compass), Haptic feedback, Speech-to-Text (STT)/ Text-to-Speech(TTS), Camera (face detection/recognition).
- Quality is defined as "meeting or exceeding customer's expectations", which influences user acceptance further influences market success.

MOBILE SOFTWARE ENGINEERING

- **Key features includes:**
  - Choosing an appropriate software methodology for your mobile project
  - Understanding how target handsets dictate the functionality of your application
  - Performing thorough, accurate, and ongoing feasibility analyses
  - Mitigating the risks associated with preproduction handsets
  - Keeping track of handset functionality through configuration management
  - Designing a responsive, stable application on a memory restrictive system
  - Designing user interfaces for a variety of devices with different user experiences
  - Testing the application thoroughly on the target handsets
  - Incorporating third-party requirements that affect where you can sell your application
  - Deploying and maintaining a mobile application

MOBILE SOFTWARE ENGINEERING

- **Choosing a Software Methodology**
  - Developers can easily adapt most modern software methodologies to mobile development.
  - Whether opts for traditional Rapid Application Development (RAD) principles or more modern variants of Agile Software Development like Scrum, mobile applications have some unique requirements.
  - **Understanding the Dangers of Waterfall Approaches**
    - The short development cycle might tempt some to use a Waterfall approach, but beware of the inflexibility issue. (**inability to return to previous stages)**
    - Changes to target handsets (especially preproduction models), ongoing feasibility, and performance concerns, and the need for quality assurance to test early and often on the target devices (not just the emulator) make it difficult for strict waterfall approaches to succeed with mobile projects.

MOBILE SOFTWARE ENGINEERING

- **Understanding the Value of Iteration**
  - Because of the speed, iterative methods are the most successful strategies adapted to mobile development.
  - Rapid prototyping enables developers and personnel responsible for quality assurance to have opportunity to evaluate the feasibility and performance of mobile app and adapt to the changes.
- **Gathering Application Requirements**
  - Requirements analyses for mobile applications can be more complex than that of traditional desktop applications.
  - As multiple handsets are involved, generally two approaches for determining project requirements are:
    - lowest common denominator method:
    - customization method.

MOBILE SOFTWARE ENGINEERING

- **Lowest common denominator method**:
  - Design the app to run across large number of devices.
  - Only requirements that can be met by all devices are included, by considering the most inferior handset.
  - requirements such as input methods, screen resolution, and the Software Development Kit (SDK) version.
- **Customization method:**
  - app is tailored for specific handsets.
  - This method works well for projects targeting a small number of target handsets.
  - Advantage of utilizing specific phone features.
  - Drawbacks include source code fragmentation (many branches of the same code), increased testing requirements, and more difficult to add new handsets in the future.

MOBILE SOFTWARE ENGINEERING

- **Incorporating Third-Party Requirements**
  - Android License Agreement Requirements
  - Google Maps API License Agreement Requirements (if applicable)
  - Third-Party API Requirements (if applicable)
  - Android Market Requirements
  - Mobile Carrier/Operator Requirements
  - Other Application Store Requirements (if applicable)
  - Application Certification Requirements (if applicable)

  Incorporating these requirements into mobile project plan early is essential not only for keeping your project on schedule but also adding them at later stage can be risky and time consuming.

MOBILE SOFTWARE ENGINEERING

- **Managing a Handset Database**
  - Creating a handset database is a great way to keep track of both marketing and device specification details for target handsets.
  - The phone database is best implemented early, when project requirements are just determined and target handsets are determined.
  - **Determining which Handsets to Track**
    - Some app development companies track only the handsets, they actively develop for,
    - Whereas some also track handsets that can be included in the future, or lower priority handsets.
    - Handsets can be included in the database during the Requirements phase but also later as change in project scope.

MOBILE SOFTWARE ENGINEERING

- **Storing Handset Data**
  - The handset database should be designed to contain all information about any handset that would be helpful for developing and selling applications. This data should include:
    - Important handset technical specification details (screen resolution, hardware details, supported media formats, input methods, localization)
    - Any known issues with handsets (bugs and important limitations)
    - Handset carrier information
    - Firmware upgrade information (as changes might have no impact on the application or warrant an entirely separate handset entry)
    - Actual testing handset information (which handsets have been purchased or loaned through manufacturer or carrier loaner programs, how many are available)

MOBILE SOFTWARE ENGINEERING

- **Using Handset Data**
  - Product designers use the database to develop the most appropriate application user interface for the target phones.
  - Media artists use the database to generate application assets such as graphics, videos, and audio in supported media file formats and resolutions appropriate for the target phones.
  - Project managers use the database to determine the handsets that must be acquired for development and testing purposes on the project and development priorities.
  - Software developers use the database to design and develop applications compatible with target handset device specifications.
  - Quality assurance personnel use the database to design and develop test plans target handset device specifications and to test the application thoroughly.
  - Marketing and sales professionals use the database to estimate sales figures for released applications.

MOBILE SOFTWARE ENGINEERING

- **Assessing Project Risks**
  - Mobile projects need to be aware of the outside influences that can affect their project schedule and whether the project requirements can be met.
  - Some of the risk factors include identifying and acquiring target handsets and continually reassessing application feasibility.
  - **Identifying Target Handsets**
    - Each handset will have different capabilities, a different user interface, and unique device limitations.
    - Target handsets are generally determined in one of two ways:
      - Develop for popular "killer" phone.
      - Develop an application for maximum coverage.
  - **Acquiring Target Handsets**
    - Getting proper simulator (emulator) or actual device is sometime quite easy
    - For preproduction handsets, join manufacturer and operator developer programs.

MOBILE SOFTWARE ENGINEERING

- **Determining Feasibility of Application Requirements**
  - Mobile developers are at the mercy of the handset limitations, which vary in terms of memory and processing power.
  - True feasibility assessment can be done only on the physical handset, not the software emulator.
  - Mobile developers most constantly revisit feasibility, application responsiveness, and performance throughout the development process.
- **Understanding Quality Assurance Risks:** The quality assurance team has its work cut out as the testing environment is generally ideal.
  - **Testing Early, Testing Often**
    - Get target handsets in-hand as early as possible.
    - Don't wait until the last minute to gather the test hardware.
  - **Testing on the Handset**
    - *Testing on the emulator is helpful, but testing on the handset is essential.*

MOBILE SOFTWARE ENGINEERING

- **Mitigating the Risk of Limited Real-World Testing Opportunities**
  - Ideal testing generally takes place in a lab (at some location which has primary cell tower, satellite fixes and related phone signal strength, availability of data services, LBS information, locale information)
  - The quality assurance team needs to get creative to mitigate the risks of testing while narrowing range of these factors.
  - For example, essential to test application when the phone has no signal (and in airplane mode, and such) to make sure it don't crash and burn under such conditions that each user experience at some point in time.
- **Testing Client-Server Applications**
  - Make sure thorough server testing is part of the overall test plan—not just the phone client portion of the overall solution.

MOBILE SOFTWARE ENGINEERING

- **Writing Essential Project Documentation**
  - Good documentation serves a variety of purposes in mobile development.
  - Some documentation you should consider including in your project includes:
    - Requirements Analysis and Prioritization
    - Risk Assessment and Management
    - Application Architecture and Design
    - Feasibility Studies including Performance Benchmarking
    - Technical Specifications (Overall, Server, Handset-Specific Client)
    - Detailed User-Interface Specifications (General, Handset-Specific)
    - Test Plans, Test Scripts, Test Cases (General, Handset-Specific)
    - Scope Change Documentation

MOBILE SOFTWARE ENGINEERING

- **Choosing a Source Control System**
  - Mobile development considerations for developers to handle configuration management for any mobile project:
    - Ability to keep track of source code (Java, Swift) and binaries (Android packages, etc.)
    - Ability to keep track of application resources by handset configuration (Graphics, etc.)
    - Integration with the developer's chosen development environment (like Eclipse, Xcode, etc.)
- **Designing Mobile Applications**
  - Here, developer must consider the constraints that handset imposes and decide type of application framework is best for a given project.
  - **Understanding Mobile Device Limitations**
    - The memory and processing power constraints of all target handsets must be kept in mind when designing and developing mobile applications.

MOBILE SOFTWARE ENGINEERING

- **Exploring Common Mobile Application Architectures**
  - Mobile applications have traditionally come in two basic models: stand-alone applications and network-driven applications.
    - Stand-alone applications are packaged with everything it require and rely on the handset to do all kind of work. All processing work is done locally, in memory, and is subject to the limitations of the device. An example of a reasonable stand-alone application is a Solitaire game.
    - Network-driven applications provide a lightweight handset client but rely on the network to provide a portion of its content and functionality. Good examples of network-driven applications include
      - Customizable content such as ringtone and wallpaper applications
      - Applications with noncritical process and memory intensive operations that can be offloaded to a powerful server and the results delivered back to the client
      - Any application that provides additional features at a later date without a full update to the binary

MOBILE SOFTWARE ENGINEERING

- **Designing for Application Interoperability**
  - Mobile application designers should consider how they will interface with other applications on the handset, including other applications written by the same developer.
  - Some issues to address are
    - Will your application rely on other content providers?
    - Are these content providers guaranteed to be installed on the phone?
    - Will your application act as a content provider? What data will it provide?
    - Will your application have background features? Act as a service?
    - Will your application rely on third-party services or optional components?
    - Will your application expose its functionality through a remote interface (AIDL)?

MOBILE SOFTWARE ENGINEERING

- **Developing Mobile Applications**
  - Mobile application implementation follows the same design principles as other platforms.
  - The steps mobile developers take during implementation are fairly straightforward:
    - Write and compile the code.
    - Run the application in the software emulator.
    - Test and debug the application in the software emulator.
    - Package and deploy the application to the target handset.
    - Test and debug the application on the target handset.
    - Incorporate changes from team and repeat until application is complete.
- **Testing Mobile Applications**
  - ***Test early, test often, test on the actual device.***

MOBILE SOFTWARE ENGINEERING

- **Deploying Mobile Applications**
  - Need to determine what methods to be used for distributing mobile applications.
  - One can market applications directly or use some leverage third-party marketplaces like the Android Market, App-store, etc.
- **Determining Target Markets**
  - Developers must take into account any requirements imposed by third parties distribution mechanisms.
  - They might impose quality requirements such as testing certifications.
  - Distributers might also impose content restrictions such as barring objectionable content.
- **Supporting and Maintaining Mobile Applications**
  - Mobile hardware on the market changes quickly, and mobile application need to stay on top of the market. To achieve this, some consideration are required
    - **Maintaining Adequate Application Documentation:** Keeping adequate development and testing documentation, including specifications and test scripts, is even more vital as maintenance might be done by some other engineer.

MOBILE SOFTWARE ENGINEERING

- **Managing Live Server Changes:**
  - Always treat any live server with the care it deserves.
  - This means backups and upgrades need to be timed appropriately.
  - Data needs to be safeguarded and user privacy maintained at all times.
  - Rollouts should be managed carefully because live mobile application users might rely on its availability.
  - Do not underestimate the server-side development or testing needs.
  - Always test server rollouts in a safe testing environment before "going live."
- **Identifying Low-Risk Porting Opportunities**
  - Analyze handset similarities to identify easy porting projects.
  - An application was originally developed for a specific handset, but has several popular handsets with similar specifications.
  - Porting an existing application sometimes is as straightforward task as generating a new build (with appropriate versioning) and testing the application on the new handsets.

THANK YOU