National College of Ireland

Mobile Architecture & Security

Designing Mobile Apps

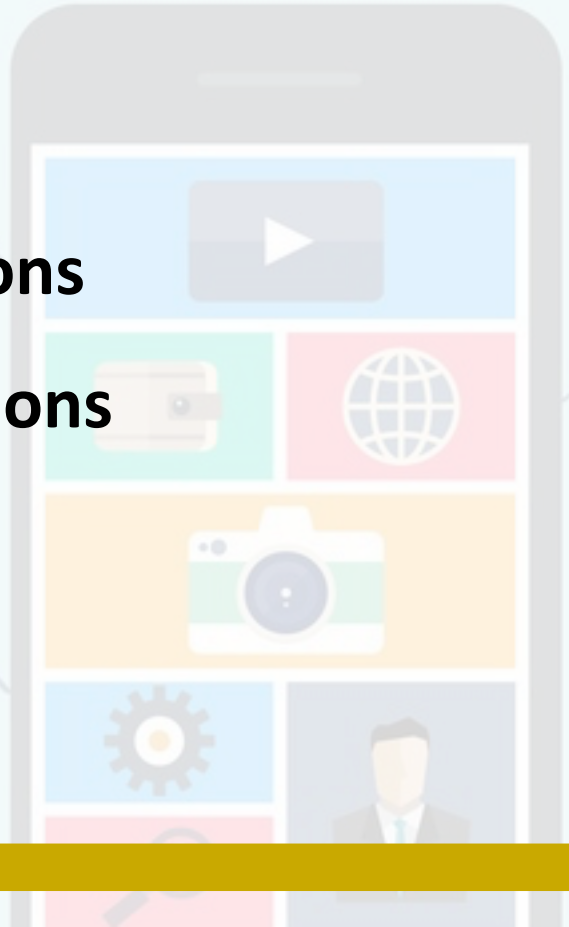
Mikhail.Timofeev@ncirl.ie

Overview

① Specific Design Issues

② Technology Considerations

③ Deployment Considerations



1.1 Specific Design Issues

- Authentication and Authorization
- Caching
- Communication
- Configuration Management
- Data Access
- Device Specifics
- Exception Management
- Logging
- Porting Applications
- Power Management
- Synchronization
- Testing
- User Interface
- Validation

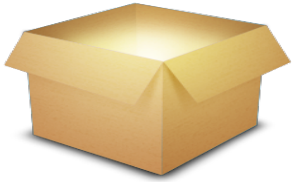
1.1 Specific Design Issues: Authentication and Authorization

- Designing an effective authentication and authorization strategy is important for the security and reliability of your application
- Mobile devices are usually designed to be single user devices
- Normally lack basic user profile and security tracking beyond just a simple password
- The discoverability of mobile devices over protocols such as Bluetooth can present users with unexpected risks
- Mobile application design can also be especially challenging due to connectivity interruptions
- Consider all possible connectivity scenarios, whether over the air or hard wired

1.1 Specific Design Issues: Authentication and Authorization

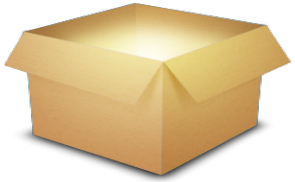
- Design authentication and authorization for both fully connected and occasionally connected scenarios:
  - synchronization over the air
  - cradled (PC/Mac) synchronization
  - Bluetooth discovery
  - synchronization over a Virtual Private Network (VPN)
  - local SD memory card synchronization
- Consider that different devices might have variations in their programming security models, which can affect authorization for resource access
- Do not assume that security mechanisms available on larger platforms will be available on a mobile platform
- Identify trust boundaries within your mobile application layers

1.1 Specific Design Issues: Caching



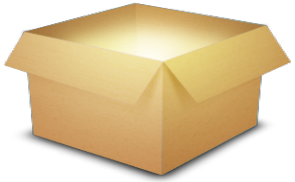
- Use caching to improve the performance and responsiveness of your application, and to support operation when there is no network connection
- Caching can optimize reference data lookups, avoid network round trips, and prevent unnecessarily duplicated processing
- When deciding what data to cache, consider the limited resources of the device; you will have less storage space available than on a desktop computer

1.1 Specific Design Issues: Caching



- Identify your performance objectives:
  - For example, determine your minimum response time and battery life
  - Test the performance of the specific devices you will be using
  - Most mobile devices use only flash memory, which is likely to be slower than the memory used in desktop computer
- Design for minimum memory footprint
  - Cache only data that is absolutely necessary for the application to function, or expensive to transform into a ready to use format
  - If designing a memory-intensive application, detect low memory scenarios and design a mechanism for prioritizing the data to discard as available memory decreases
  - However, consider caching any data, including volatile data, that the application will need in an occasionally connected or offline scenario
  - Also, ensure that the application can survive the situation where cached data is not available in offline or occasionally connected scenarios

1.1 Specific Design Issues: Caching



- Choose the appropriate cache location, such as on the device, at the mobile gateway, or in the database server.
  - Consider using SQL Server Compact edition (Windows Phone) or localStorage (iPhone, Android) for caching instead of device memory because memory consumed by the application may be cleared in low-memory situations
- Ensure that sensitive data is encrypted when caching, especially when caching data in removable memory media, but also consider encryption when caching data in device memory

1.1 Specific Design Issues: Communication

- Device communication includes wireless communication (over the air) and wired communication with a host computer, as well as more specialized communication such as Bluetooth or Infrared Data Association (IrDA)
- When communicating over the air, consider data security to protect sensitive data from theft or tampering
- If you are communicating through Web service interfaces, use mechanisms such as the WS-Secure standards to secure the data
- Keep in mind that wireless device communication is more likely to be interrupted than communication from a computer, and that your application might be required to operate for long periods in a disconnected state

1.1 Specific Design Issues: Communication

- Design asynchronous, threaded communication to improve performance and usability in occasionally connected scenarios
  - Limited bandwidth connections common on mobile devices can reduce performance and affect usability, especially if they block the user interface
  - Use appropriate communication protocols, and consider how the application will behave when multiple connection types are available
  - Consider allowing users to choose the connection to use, and to switch off communication to preserve battery life when appropriate.
- If you are designing an application that will run on a mobile phone, consider the effects of receiving a phone call during communication or program execution

1.1 Specific Design Issues: Communication

- Protect communication over untrusted connections, such as Web services and other over the air methods
  - Consider using encryption and digital signatures for sensitive data, and ensure that data passed over a VPN is protected
  - However, consider the effects of communication security on performance and battery life
- If you must access data from multiple sources, interoperate with other applications, or work while disconnected, consider using Web services for communication

1.1 Specific Design Issues: Configuration Management

- When designing device configuration management, consider how to handle device resets, as well as whether you want to allow configuration of your application over the air or from a host computer
- Choose an appropriate format for configuration information
  - Consider a binary format over XML to minimize memory use
  - Consider using compression library routines to reduce the memory requirements for storing configuration and state information
  - Ensure that you encrypt sensitive data stored in configuration files
- Ensure that your design supports restoration of configuration after a device reset
  - Consider how you will synchronize configuration information over the air and with a host computer when cradled, and ensure that you are familiar with the techniques used by different manufacturers for loading configuration settings

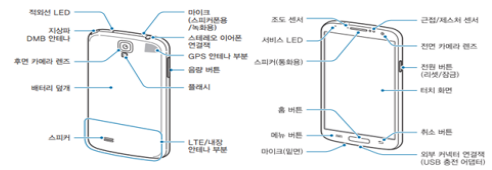
1.1 Specific Design Issues: Data Access

- Data access on a mobile device is constrained by unreliable network connections and the hardware constraints of the device itself
- When designing data access, consider how low bandwidth, high latency, and intermittent connectivity will affect your design
- Consider using a local device database that provides synchronization services, such as SQL Server Compact Edition or localStorage
- Program for data integrity
  - Files that remain open during device suspend and power failures may cause data integrity issues, especially when data is stored on a removable storage device
  - Include exception handling and retry logic to ensure that file operations succeed
  - To ensure data integrity in cases where the device loses power or connectivity, consider using transactions with SQL Server Mobile.

1.1 Specific Design Issues: Data Access

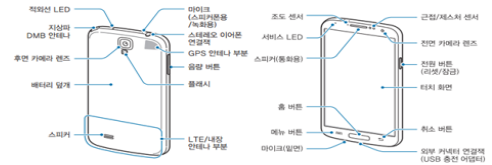
- Do not assume that removable storage will always be available, as a user can remove it at any time; check for the existence of a removable storage device before writing
- If you use XML to store or transfer data, consider its overall size and impact on performance
  - XML increases both bandwidth and local storage requirements
  - Use compression algorithms or a non-XML transfer method.
- Minimize performance impact by designing for efficient database access and data processing

1.1 Specific Design Issues: Device Specifics



- Mobile device design and development is unique due to the constrained and differing nature of device hardware.
- Optimize the application for the device by considering factors such as screen size and orientation, network bandwidth, memory storage space, processor performance, and other hardware capabilities
- Consider device-specific capabilities that you can use to enhance your application functionality such as accelerometers, graphics processing units, GPS, haptic (touch, force, and vibration) feedback, compass, camera, and fingerprint readers

1.1 Specific Design Issues: Device Specifics



- If you are developing for more than one device, design first for the subset of functionality that exists on all of the devices and then customize the code to detect and use device-specific features when they are available
- Consider limited memory resources and optimize your application to use the minimum amount of memory. When memory is low, the system may release cached intermediate language (IL) code to reduce its own memory footprint, return to interpreted mode, and thus slow overall execution
- Create modular code to allow easy module removal from executables. This covers cases where separate smaller executable files are required due to constraints in device memory size.
- Consider using programming shortcuts as opposed to following pure programming practices that can inflate code size and memory consumption. For example, examine the cost of using pure object-oriented practices such as abstract base classes and repeated object encapsulation. Consider using lazy initialization so that objects are instantiated only when required.

1.1 Specific Design Issues: Exception Management

- Good exception handling in your mobile application prevents sensitive exception details from being revealed to the user, improves application robustness, and helps keep your application from remaining in an inconsistent state when an error occurs.
- Design your application to recover to a known good state after an exception occurs without revealing sensitive information to the end user.
- Catch exceptions only if you can handle them, and do not use exceptions to control logic flow. Ensure that you design a global error handler to catch unhandled exceptions.
- Design an appropriate logging and notification strategy that stores sufficient details about exceptions, but bear in mind memory and storage limitations of mobile devices.

1.1 Specific Design Issues: Logging

- Because of the limited memory available on mobile devices, logging and instrumentation should be limited to only the most necessary cases; Consider the fact that some of the logs might have to be generated on the device and must be synchronized with the server during periods of network connectivity.
- There is no Event Log mechanism in Windows Mobile. Consider using a third-party logging mechanism that supports the .NET Compact Framework, such as OpenNetCF, NLog, or log4Net
- If you carry out extensive logging on the device, consider logging in an abbreviated or compressed format to minimize memory and storage impact. Alternatively, consider remote logging instead of logging on the device

1.1 Specific Design Issues: Logging

- Consider using platform features such as health monitoring on the server, and mobile device services on the device, to log and audit events. Explore adding remote health monitoring capabilities using the Open Mobile Alliance Device Management (OMA DM) standard.
- Synchronize between the mobile database logs and the server database logs to maintain audit capabilities on the server. If you have an Active Directory infrastructure, consider using the System Center Mobile Device Manager to extract logs from mobile devices. See "Technology Considerations" for the requirements of Mobile Device Manager.
- Do not store sensitive information in log and audit files unless absolutely necessary, and ensure that any sensitive information is protected through encryption.
- Decide what constitutes unusual or suspicious activity on a device, and log information based on these scenarios.

1.1 Specific Design Issues: Porting Applications

- Developers often want to port part or all of an existing application to a mobile device. Certain types of applications will be easier to port than others, and it is unlikely that you will be able to port the code directly without modification.
- If you are porting a rich client application from the desktop, rewrite the application in its entirety. Rich clients are rarely designed to suit a small screen size and limited memory and disk resources.
- If you are porting a Web application to a mobile device, consider rewriting the UI for the smaller screen size. Also, consider communication limitations and interface chattiness as these can translate into increased power usage and connection costs for the user.
- If you are porting a RIA client, carry out research to discover which code will port without modification. Consult the "Technology Considerations" section of this chapter for specific advice.

1.1 Specific Design Issues: Porting Applications

- Research and utilize tools to assist in porting. For example, Java-to-C++ convertors are available. When converting from Smartphone to Pocket PC code, Visual Studio allows you to change the target platform and provides warnings when you are using Smartphone-specific functionality. You can also link Visual Studio Desktop and Mobile projects to discover what is portable between the two projects.
- Do not assume that you can port custom controls to a mobile application without modification. Supported APIs, memory footprint, and UI behavior are different on a mobile device. Test the controls as early as possible so that you can plan to rewrite them or find an alternative if required.

1.1 Specific Design Issues: Power Management

- Power is the major limiting design factor for mobile devices. All design decisions should take into account how much power the device consumes, and their effect on overall battery life. If you have a choice, consider devices that can draw power from Universal Serial Bus (USB) or other types of data connections.
- Implement power profiles to increase performance when the device is plugged into external power and not charging its battery. Allow the user to turn off features of the device when not in use or when not required. Common examples are screen backlighting, hard drives, GPS functions, speakers, and wireless communications.

1.1 Specific Design Issues: Power Management

- To conserve battery life, do not update the UI while the application is running in the background.
- Choose protocols, design service interfaces, and batch communications in such a way as to transfer the smallest number of bytes possible over the air. Consider both power usage as well as network speed when choosing communication methods, and consider deferring nonessential wireless communications until the device is using external power.
- If you are considering using the 3G hardware communications protocol, consider that while it is significantly faster, it also currently uses much more power than its predecessors, such as the Edge protocol. When you are using 3G, be sure to communicate in batched bursts and to shut down communication at times when it is not required.

1.1 Specific Design Issues: Synchronization

- Consider whether you want to support over the air synchronization, cradled synchronization, or both. Because synchronization will often involve sensitive data, consider how to secure your synchronization data, especially when synchronizing over the air.
- If your users will be synchronizing with a host computer, consider including cradled synchronization in your design. If your users must synchronize data when away from the office, consider including over the air synchronization in your design.

1.1 Specific Design Issues: Synchronization

- Ensure that the application can recover when synchronization is reset or when synchronization is interrupted, and decide how you will manage synchronization conflicts.
- Ensure that synchronization communication is protected, perhaps using encryption and digital certificates, and use secure channels. Be especially sure to apply appropriate authentication and authorization when using Bluetooth synchronization.
- If you must support bidirectional synchronization to SQL Server, consider using merge replication synchronization. Remember that merge synchronization will synchronize all of the data in the merge set, which may require additional network bandwidth and can adversely affect performance.
- Consider store and forward synchronization using WCF rather than e-mail or SMS (text message), as WCF guarantees delivery and works well in occasionally connected scenarios.

1.1 Specific Design Issues: Testing

- Mobile application debugging can be much more costly than debugging a similar application on a computer. Consider this debugging cost when deciding which devices, and how many devices, your application will support. Also keep in mind that it can be harder to get debug information from the device, and that device emulators do not always perfectly simulate the device hardware environment.
- Understand your debugging costs when choosing which devices to support. Factor in tools support, the cost of initial (and perhaps replacement) test devices, and the cost of software-based device emulators.
- If you have access to the physical device you are targeting, debug your code on the actual device rather than using an emulator. If the device is not available, use an emulator for initial testing and debugging. Consider that an emulator might run code more slowly than the actual device.

1.1 Specific Design Issues: Testing

- As soon as you obtain the physical device, switch to running code on the device connected to a normal computer. Test scenarios where your device is fully disconnected from any network or connection, including being disconnected from a computer debugging session, and perform final testing on your device when not connected to a computer. Add temporary or permanent mechanisms to debug problems in this scenario. Consider the needs of people who will support the device.
- If you are an OEM and your device has not yet been created, note that it is possible to debug a mobile program on a dedicated x86-based Windows CE computer. Consider this option until your device is available.

1.1 Specific Design Issues: User Interface

- When designing the UI for a mobile application, do not try to adapt or reuse the UI from a desktop application. Design your device UI so that it is as simple as possible and tailored specifically for pen-based input and limited data entry capabilities where appropriate. Consider the fact that your mobile application will run in full screen mode and will only be able to display a single window at a time; and, therefore, blocking operations will prevent the user from interacting with the application.
- Design for a single window, full screen UI. If your device will be a single user device running only the main application, consider using kiosk mode.

1.1 Specific Design Issues: User Interface

- Take into account the various screen sizes and orientations of your target devices when designing your application UI. Also, consider the limitations imposed by the small screen size, limited API, and reduced range of UI controls compared to desktop environments.
- Design for usability by supporting touchscreen or stylus-driven UI. Place menu bars and other controls at the bottom of the screen (expanding upwards when required) to prevent the user's hands from obscuring the display. Support touchscreen input by making buttons large enough, and lay out controls so that the UI is usable using a finger or stylus for input.
- Give the user visual indication of blocking operations; for example, an hourglass cursor.

1.1 Specific Design Issues: Validation

- Use validation to protect the device and your application, and to improve usability. Validating input values before submitting them to a remote server can reduce communication roundtrips and improve the performance and usability of the application, especially in occasionally connected or disconnected scenarios.
- Validate data input by the user where possible to prevent unnecessary communication and server roundtrips. This also makes the application more responsive when the user enters invalid values.
- Validate all data received during communication with a host computer and during over the air communication.
- Ensure that you protect hardware resources, such as the camera and initiation of phone calls, by validating code and actions that automatically initiate these features.
- Consider the limited resources and performance of the device by designing efficient validation mechanisms that have the minimum memory footprint.

1.2 Technology Considerations

- HTML5/CSS3/JavaScript
  - Various JS frameworks
  - Any text-editor
- Objective-C
  - iOS SDK
  - Xcode
- Java
  - Android SDK
  - Eclipse

- Microsoft Visual Basic or Visual C#
  - Microsoft Silverlight for Mobile / .NET Compact Framework
  - Visual Studio
- Mobile Development Frameworks
  - PhoneGap (JavaScript)
  - Appcelerator (JavaScript)
  - Codename One (Java)

1.4 Performance Considerations

Native

- The term “native app” refers to an app that has been built specifically for one of the existing OS by using native programming languages, frameworks, APIs, etc.
- The apps are generally distributed through app “stores,” such as the Apple iTunes store for iOS or the Google Play store for Android

Web App

- A Web app is often defined as a Web site that mimics a native app — giving a great user experience by doing things such as:
  - formatting content to suit the screen size of the device
  - navigating almost instantly between links
  - working without an Internet connection and being able to swipe through to content

Hybrid

- The term “native app” refers to an app that has been built specifically for one of the existing OS by using native programming languages, frameworks, APIs, etc.

1.4 Performance Considerations

1.4 Performance Considerations: Native

Pros

- Far more popular with consumers
- Better UX than responsive web apps
- Incorporates all smartphone features like a camera or GPS
- Benefits from inclusion in app stores, for organic and promoted discovery
- Native apps can operate without an Internet connection
- It stays on the mobile device once installed (unless it's uninstalled)

Cons

- Built only for a particular operating system
- Time-consuming and higher development costs
- Needs app store review and approval with every update
- Needs additional app marketing strategy

1.4 Performance Considerations: Web App

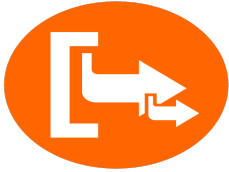
Pros

- Lower development cost (usually) compared to native app
- Single URL that works for web and mobile
- Simpler, less expensive marketing because it's a single brand property
- Greater speed and flexibility of deploying updates or bug fixes
- Faster and wider audience reach since one browser fits all
- Easier and cheaper to find teams with web development skills
- Can be packaged like a native app through PhoneGap

Cons

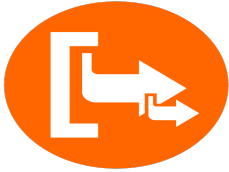
- Doesn't work properly on some devices and old browsers
- Doesn't incorporate all the smartphone features like the accelerometer or GPS, or phone dialing, that a native mobile app can leverage *(an update based on comment feedback: APIs and code libraries are evolving that could solve these issues for web apps)*
- Works only when mobile internet/ wi-fi is available (except if using localStorage) vs. native apps that can run locally

1.3 Deployment Considerations: Windows Mobile



- Microsoft Exchange ActiveSync® technology using a Windows Installer file (MSI).
- Over the air, using HTTP, SMS, or CAB files to provide install and run functionality.
- Mobile Device Manager—based, using Active Directory to load from a CAB or MSI file.
- Post load and autorun, which loads a company-specific package as part of the operating system.
- Site loading, manually using an SD card.

1.3 Deployment Considerations: Android & iOS



Android

- Distributing Through an App Marketplace (publish instantly)
- Distributing Your Apps by Email
- Distributing Through a Website

iOS

- Exclusively through AppStore (2 weeks review process)

What about Web or Hybrid Apps?

National College of Ireland

Questions?

MTimofeev@ncirl.ie