Mobile Device Architecture

CS 4720 – Mobile Application Development
The Way Back Time

• When a phone was a phone...
• Plus a story!
Oh yes... this was a phone...

The Motorola DynaTAC 8000X
• 1983
• 13 x 1.75 x 3.5
• 2.5 pounds
• $3,995
• + Monthly Fee
• + Pay per minute

Then there was the bag phone...
Nokia Invents Mobile Phone Gaming!

• Obviously I’m talking about the N-Gage!
• ...
• Okay, remember Snake on the old Nokia phones?
• Other early apps include: basic contact apps, Pong, and Tetris
Third-Party Apps Begin

- Mobile phones stopped being a novelty
- Batteries got better, form factors improved, coverage improved, plans were... better...
- The handset manufacturers didn’t want to write all the applications for these new phones
- However... they didn’t want to open up their platform...
- The first mobile web platform was born
WAP

• Wireless Application Protocol
• Basically it’s a stripped-down HTTP that was meant to be better at transmitting over the unreliable mobile network
• WAP used WML instead of HTML – used a “card” mentality
• Two popular WAP sites? CNN and ESPN
In-App Purchases Before Apps

- SMS...
When did it all change?

• With the Internet full of images and media...
• And other handheld devices selling like gangbusters (Game Boy)...
• What changed with phones?
• Phones started running known operating systems (Windows CE and Linux)
• Now bigger players were involved, and handset manufactures decided to open up
And what’s happened since?

- The mobile market is seriously fractured
- Who do you develop for?
- How do you test for EVERY phone?
- Which market works best?
- How do you port your app between platforms?
- Which tools do you use? Can you use?
And Now Google

- The Open Handset Alliance is an attempt to effectively “get everyone on the same page”
- Open Source
- Familiar Environments and Tools
- Secure OS (Linux w/ app signing)
- No Royalties or Developer Fees
The Three-Tiered Architecture

**Presentation tier**
The top-most level of the application is the user interface. The main function of the interface is to translate tasks and results to something the user can understand.

**Logic tier**
This layer coordinates the application, processes commands, makes logical decisions and evaluations, and performs calculations. It also moves and processes data between the two surrounding layers.

**Data tier**
Here information is stored and retrieved from a database or file system. The information is then passed back to the logic tier for processing, and then eventually back to the user.
The Three-Tiered Architecture

• For a web application...
  – The browser + dynamically generated HTML is the presentation layer
  – Middleware files (function-specific PHP, Java servlets on Tomcat) contain the business logic
  – The database server is the data layer
It's not news to you

- The concepts of the three-tiered architecture apply to many design scenarios
  - Keep the presentation separate so it's lightweight, easier to maintain, and can be tested separately
  - Keep the logic separate so you can change the logic as needed without having to change the presentation too much
  - Keep the data separate because you should NEVER build a system based on the current data values
Model-View-Controller

• This is the definition of what MVC is

• The MVC pattern maps:
  – Identifies what the user is asking for
  – Loads a particular resource
  – Displays the pertinent info about that resource back to the user

• To Model, Controller, View (in that order)
MVC

MODEL

UPDATES

VIEW

MANIPULATES

CONTROLLER

USER

SEES

USES
MVC

MySQL

Model

Controller

Dispatcher

Routes

Web Server

Browser

View
Controller

• The role of the controller is basically traffic cop
• It takes the request from the user and (with the assistance of the server and routing rules) turns it into a method call of sorts
• It finds the appropriate model to load
• It finds the appropriate view to load
• It returns the whole thing back to the user
Model

- The model is the representation of the data
- This may or may not be directly linked to a database (but often is in larger apps)
- A model is often translated directly into a DB table, with the columns as its attributes
- Think “class definition w/ DB backend”
- Often contains relationship rules (a Student has many Classes, for instance)
• The closest thing to what you’ve been dealing with so far is the view
• It’s effectively an HTML template that will be populated with the appropriate data from the loaded model
• It often has PHP (or whatever) embedded in it
• All UI components go here
Putting it all Together

• So, if you were building a blog, what might some of the models be?
• What are the resources that should have addresses to them?
• How do they relate to each other?
Non-shocker of the day

- We need to consider the same things for a mobile architecture
- Why? What added concerns do we have when we consider mobile applications?
  - Presentation Layer concerns
  - Logic Layer concerns
  - Data Layer concerns
Mobile Architectures

• Rich Mobile Architecture
  – Business and some data services on the phone itself
  – Good for apps that have to run “off the grid”

• Thin Mobile Architecture
  – Most business and all data services on the server
  – Good for apps that require phone services, but does require Internet connectivity

• Rich Internet Application
  – Eschews the use of any phone resources other than a browser
  – Good for apps that can run on anything with a browser
Which are we doing?

• Rich Internet Application
  – Well, it's certainly not this one... why not?
• Rich app or Thin app?
• Do both follow the three-tiered architecture structure? Why or why not?
Rich Mobile Architecture
The Presentation Layer

• Remember: it's a phone!
  – Simple = good
  – People have different sized fingers
  – User actions call functions which execute features; user actions != features
  – Phones can have varying amounts of power/resources
  – Phones can be on or off the cellular grid at any point
Presentation Approach

- Remember your client type
- Determine how you will present data in a coherent, unified method
- Determine how you will guard against untrusted input
- Ensure you have factored out your business logic
- Determine how you will pass data between layers (i.e. how you will call the service, how you will get more info about a building, etc)
The Business Layer

• For the most part, these are your web services and related functionalities
  – Each of your three web services you are using
Business Approach

- Identify FEATURES that will exist at this level
- Build components that support a feature's execution
- Hide implementation details from the presentation layer
- Determine if (how) you will cache information on the device
- Map out use cases
The Data Layer

• This will be your module that talks to the database
• Will be intertwined with the business layer to some degree
Data Approach

- For each feature, determine what data is required
- Build SQL queries around the features
- Ensure that you are using prepared statements to guard against incorrect data entry (or injection)
- Determine how you will manage connections
- Determine if you will batch up commands into one big command
Your Mobile Architecture

- Your Approach:
- Android/iOS UI which calls...
- ... your “business logic” cope
- ... that connects to some data store (local or remote)
The Android Architecture
Your Main Components

• Activities – represent a single screen with a UI
• Services – represents a process running in the background
• Content Provider – a link back to the data
• Broadcast Receiver – listens for system-wide messages to respond to
• Application – a set of Activities that make up a cohesive unit
• Intent – a message to be passed
The Activity

- Activity launched
  - onCreate()
  - onStart()
  - onPause()
  - onResume()
  - onRestart()

- App process killed
  - User navigates to the activity
  - Apps with higher priority need memory

- Activity running
  - Another activity comes into the foreground
  - User returns to the activity

- onStop()
  - The activity is no longer visible

- onDestroy()
  - The activity is finishing or being destroyed by the system

- Activity shut down
  - User navigates to the activity
The Intent

startActivity()

Activity A

Intent

Android System

onCreate()

Activity B