ITP 442 Mobile App Project

Software Architecture

What is Software Architecture?

- Definition:
  - A software system's architecture is the set of principal design decisions about the system
- Software architecture is the blueprint for a software system's construction and evolution
- Design decisions encompass every facet of the system under development
  - Structure
  - Behavior
  - Interaction
  - Non-functional properties

What is “Principal”?

- “Principal” implies a degree of importance that grants a design decision “architectural status”
  - It implies that not all design decisions are architectural
  - That is, they do not necessarily impact a system’s architecture
- How one defines “principal” will depend on what the stakeholders define as the system goals

Design for Programming

- Typically an app is divided into layers
  - A layer is a black box with a contract that defines an input and output
- To increase the cohesion and decoupling of the software
  - The layers, if well designed, help to decouple and increase the cohesion
  - Cohesion indicates strongly related software modules
  - Coupling measure the level of dependency between two software module.

Principles

- Single Responsibility Principle
  – A module should have a single responsibility, and that responsibility should be entirely encapsulated by the module
- Open Closed Principle
  – A module should be open for extension but closed for modifications
- Liskov's Substitution Principle
  – Derived types must be completely substitutable for their base types

Principles

- Interface Segregation Principle
  - Clients should not be forced to depend upon interfaces that they don't use
- Dependency Inversion Principle
  - High-level modules should not depend on low-level modules. Both should depend on abstractions. Abstractions should not depend on details. Details should depend on abstractions.
- SOLID – the "first five principles"
  - Single responsibility, Open-closed, Liskov substitution, Interface segregation and Dependency inversion

Design Pattern

- Design pattern is a general reusable solution to a commonly occurring problem within a given context.
- It's a description or template for how to solve a problem.
  - It's not a finished design that can be transformed into source code.
- There are many types of design patterns

Mobile Design Patterns

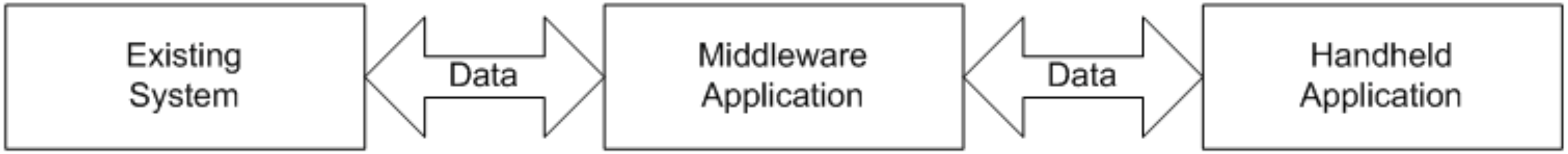
- Model View Controller
- Singleton
  - AppDelegate is a singleton
- Chain of Responsibility
  - Think of "First Responder"

Best Practices

- Use Automatic Reference Counting
- Use AppDelegate as Singleton
  - Create all common and singleton objects in AppDelegate and then expose them by UIResponder Category
- Create a property for every ivar and use self to access it

Mobile Architecture Overview

- Most mobile systems extend an existing business system or interface with an existing system.
- There are typically three major components to a mobile architecture:
  - An existing system
  - A middleware application
  - A handheld application



Mobile Architecture

- The reason a middleware application is usually needed is to provide data transformation, apply business logic, and be a central point of communication for the devices.
- If a new business system is being developed or rewritten then no middleware may be necessary; the appropriate logic can be built into the system to communicate with the devices from the start.
- However most business systems are not rewritten very often and it is economically unfeasible to rewrite them just to 'mobilize' them.
- Furthermore a middleware server may also serve a configuration management server.
- The architectures shown here are real-world architectures from actual projects. These mobile systems are in production in numerous locations.

Resources

- http://www.theshulers.com/whitepapers/mobile_architecture/index.html
- http://www.slideshare.net/MassimoOliviero/architecting-ios-project?next_slideshow=1