Architecture Perspective: Technology Platform Layers

Technology Platform Supporting Application Functionality
Lecture Objectives

• Look at a common layer architecture
• Understand the application-technology boundary
• Begin discussion of principal layers
• Recognize difference between layers/partitions and distribution tiers
A Very Common Layered Architecture
“Platform” Layers

**Application-specific**
- Specific (sub)application
- Workflow, user interaction, configuration, etc.

**Technology-general**
- Common domain abstractions
  - Data and interface standards
  - Business logic
  - Application integration
  - etc.

**Functional Architecture**
- Distributed communications
  - Remote method invocation, mail, messaging, notification, etc.
- Databases
- Application choreography
- Security
- User interface framework
  - Web-based and platform-based presentation and task flow state management
  - Error handling
  - etc.

**Technical Architecture**
- Processors
- Displays
- I/O
- Disk drives
- Network
- etc.

**Application-general**

**Technology-specific**
- Operating Systems
- Networking
- Device drivers
- etc.
Our Focus

- Focus on the application/technology boundary
- Choose/Design technology to serve required functionality

Diagram:

- Application-specific Technology-general
- Functional Architecture
- Technical Architecture
- Application-general Technology-specific

Diagram:

- Application
- Common Domain
- Middleware
- Distributed Platform
- Distributed Hardware
Three Principal Layers

Presentation Layer
Handle the interaction between the user and the software
Range from command line or text UI to rich, fat GUI clients
Faceless services offered to external applications
“User” is a separate application

Domain Layer
Concerned with providing application specific functionality:
computation, flow control, activity dispatching, etc.

Data Source Layer
Concerned with managing the system databases and access
to other systems that do work on behalf of the application
Architecture Design Approach

- Focus on system-level architecture
  - Major functional components and how they interact
  - Decisions that will be hard to change

- Top-down through a few levels of abstraction/detail
  - The system as layers and tiers
    - Application domain layer, presentation layer, and data source layer
  - Optional designs that identify major components within those layers
  - Technological approaches to implement common design patterns

- Why not Bottom UP?

- Yes, there will be code (and data models and HTML pages and XML documents and ...), but a very large part of that is pre-defined, pre-designed, and auto-generated
  - Where do you plug in your part?
A Sequence of Design

Start with the **domain** layer
Move down to the **data** layer
Move up to the **presentation** layer

Does Layer Design Order Matter?
Web-Based, N-Tier, Layered, Scalable Architectures

Web Client -> Web Server
  HTML, CGI, ASP, Java
  Web Browser
  Applets, .NET

Web Server -> Application Server
  Java, .NET, etc
  Business Object Services
  Business Object Engine

Application Server -> Database Servers

...
Layers and Partitions ≠ Distribution Tiers

Each vertical and horizontal partition is a *possible* place to distribute or replicate functionality

- But it is not required

- All this can be on a **single computer**
  - Indeed, for a very simple application, it could all be in **one class**!
  - With each layer as a separate subroutine

- Distribution is expensive in performance, development, hardware, etc.
Questions