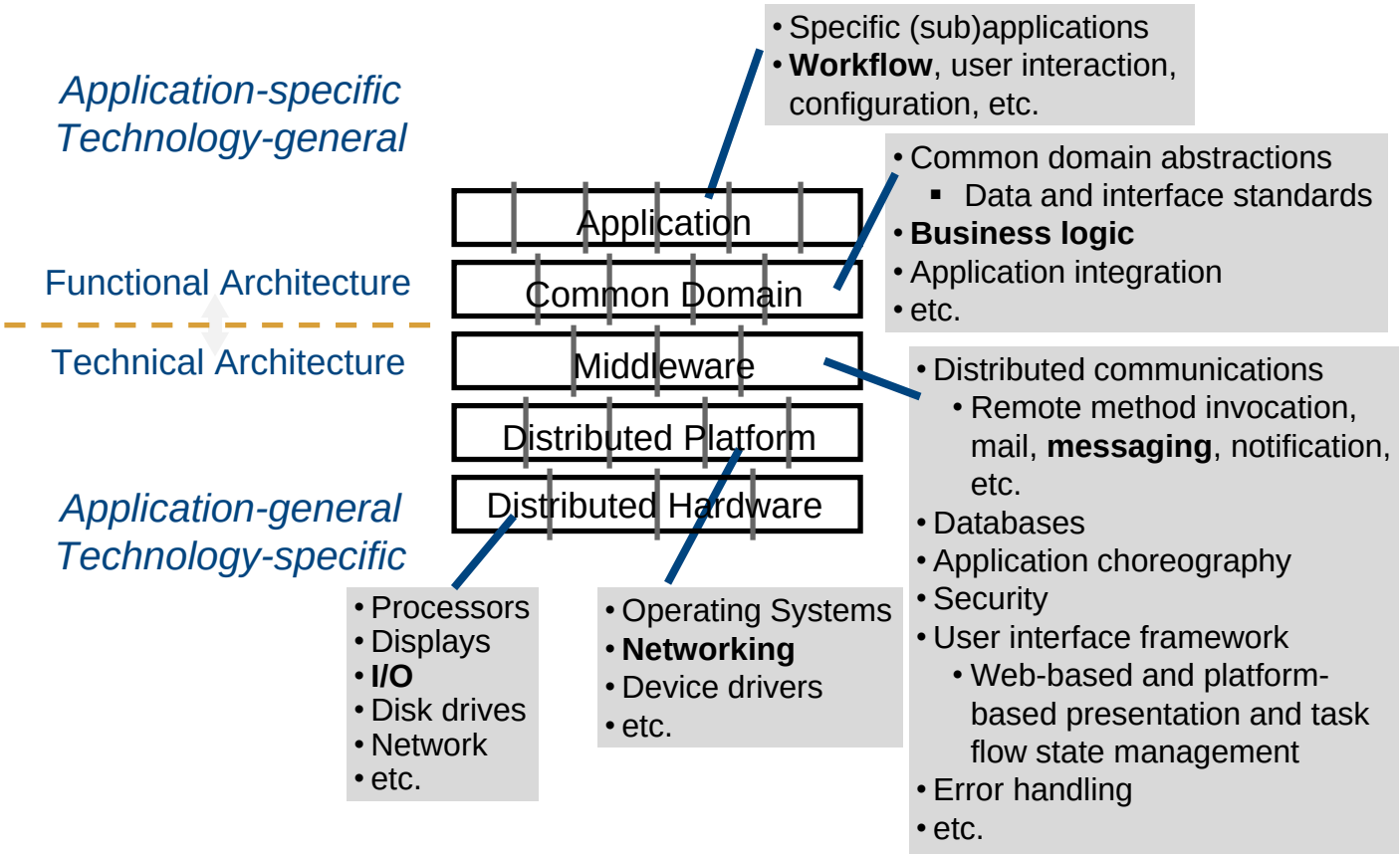
Architecture Perspective: Technology Platform Layers

Technology Platform Supporting Application Functionality

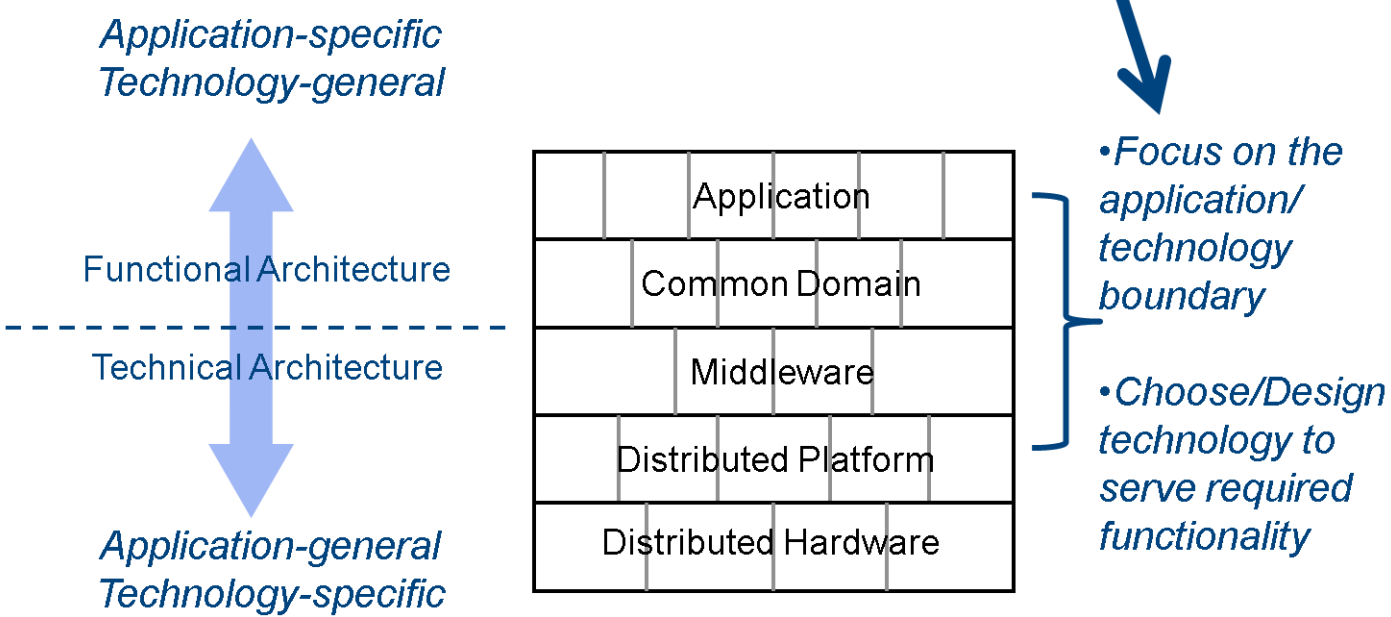
Lecture Objectives

- Look at a common layer architecture
- Understand the application-technology boundary
- Begin discussion of **principal** layers
- Recognize difference between layers/partitions and distribution tiers

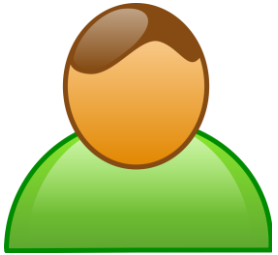
A Very Common Layered Architecture
"Platform" Layers

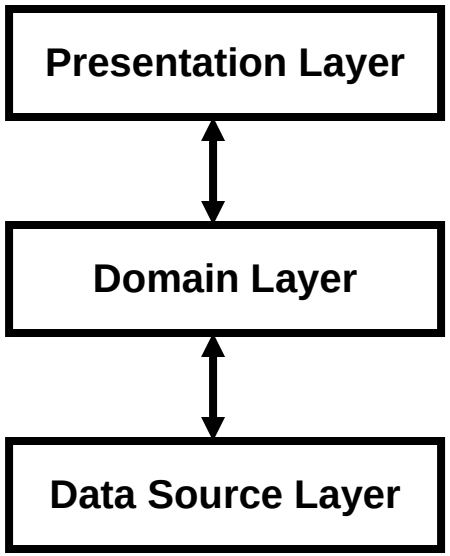


Our Focus



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.

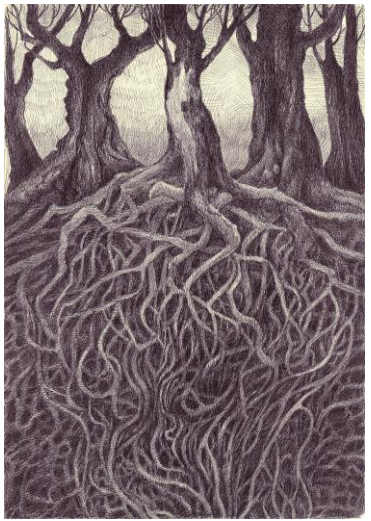
Data Source Layer

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**
- 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?

Why not Bottom UP?



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?

Tiers vs. Layers

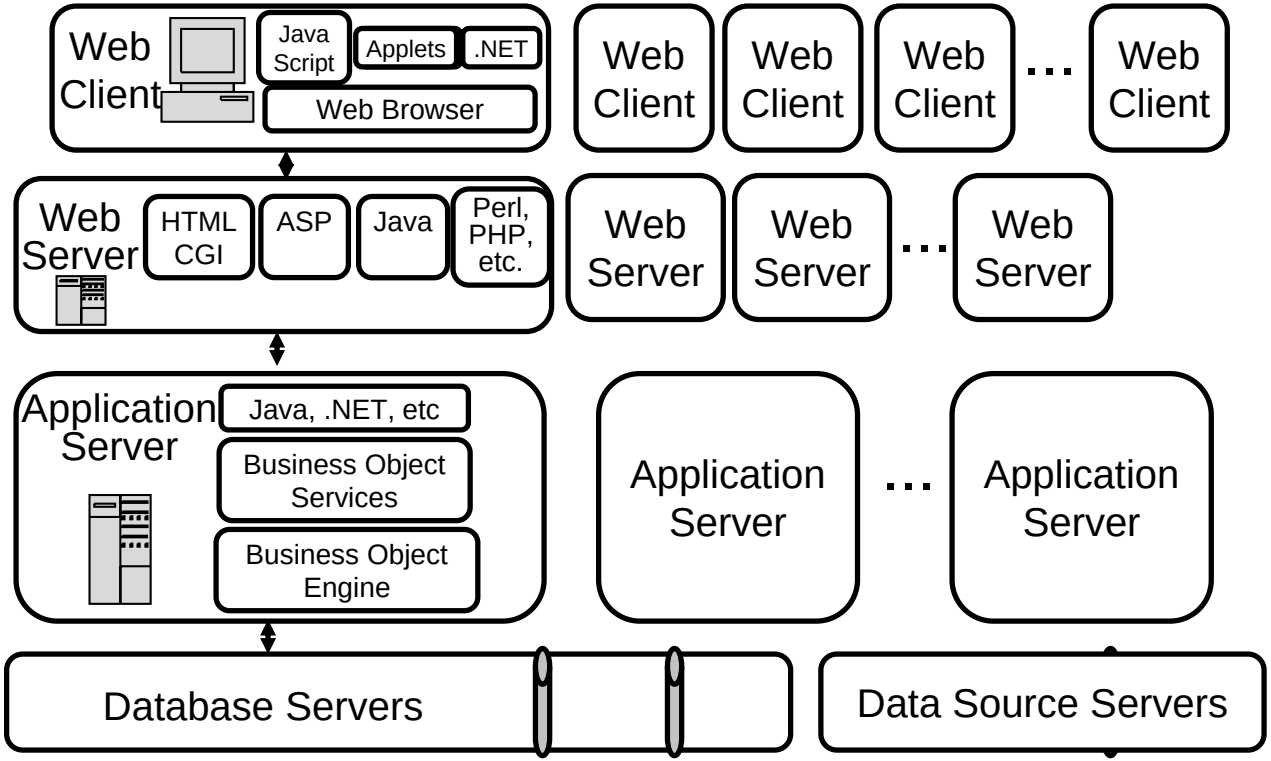
Layers (logical):

- All about the **how** “the code” is organized
- No assumptions about where “code” runs

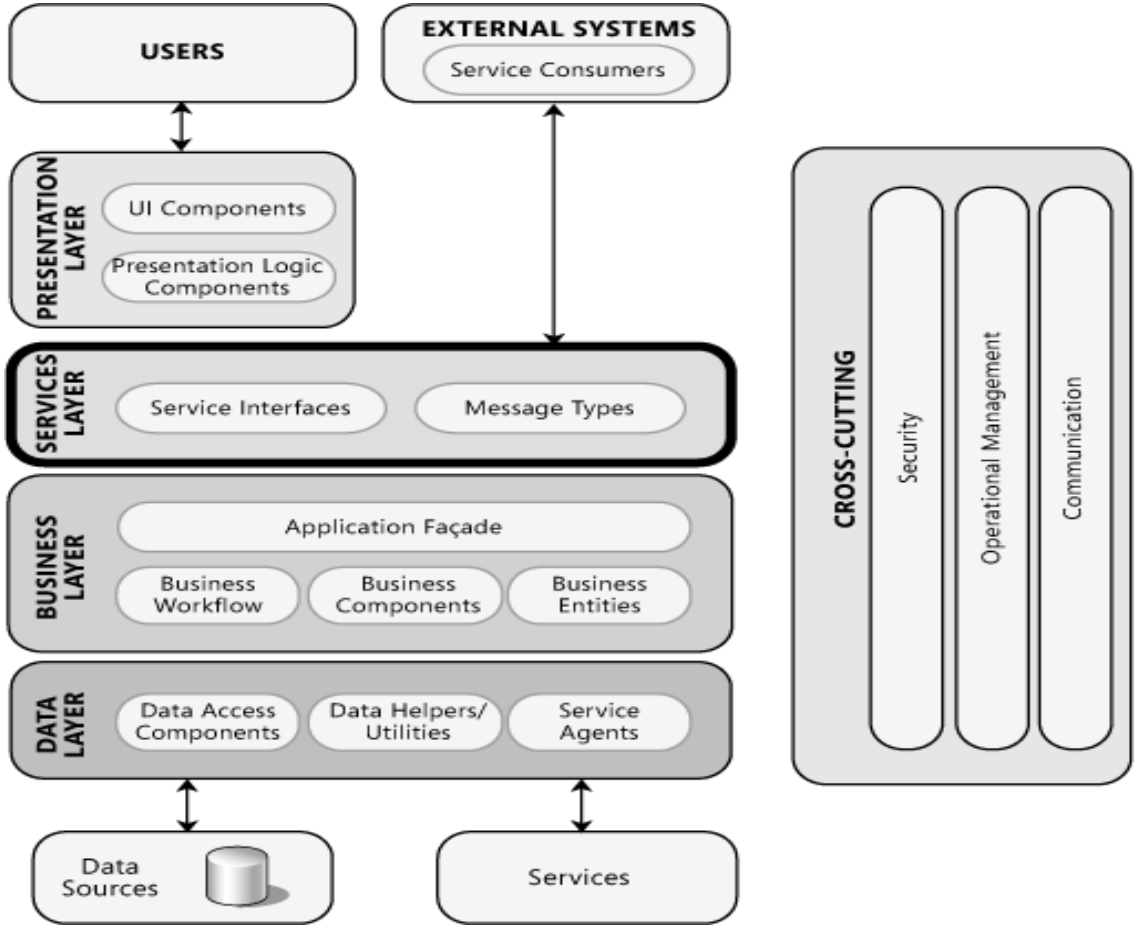
Tiers (physical):

- All about **where** the code executes
- The places onto which code is deployed

Web-Based, N-Tier, Layered, Scalable Architectures



Layers



https://docs.microsoft.com/en-us/previous-versions/msp-n-p/ee658109(v=pandp.10)

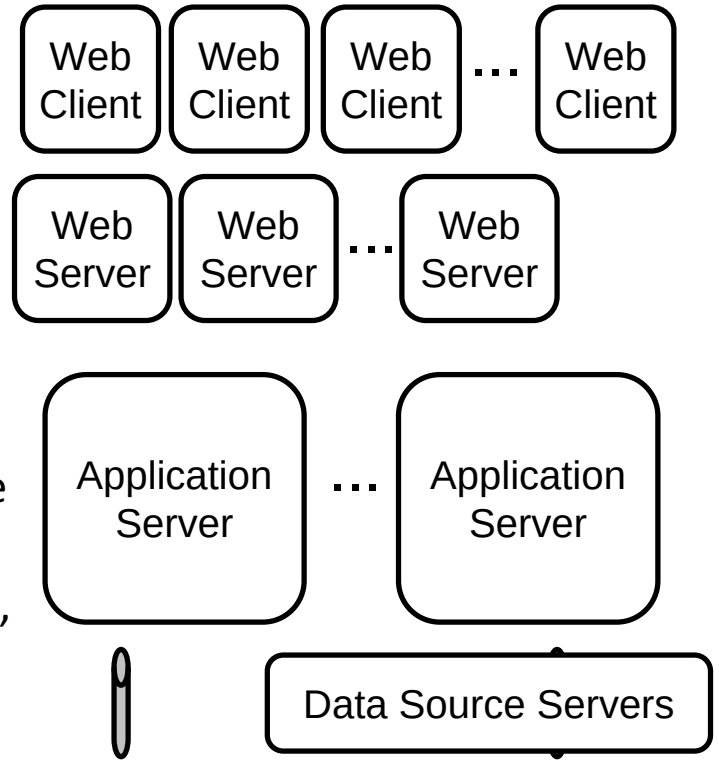
Layers and Partitions ≠> Distribution Tiers

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