CSSE 374: Logical Architecture

Shawn Bohner

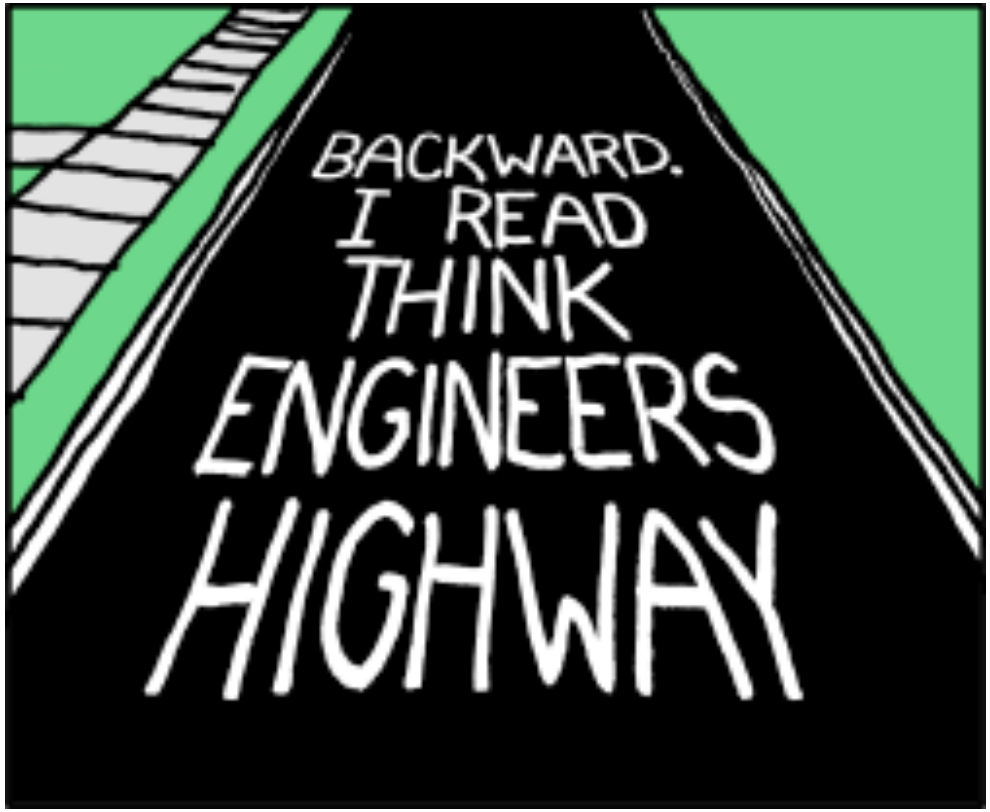
Office: Moench Room F212

Phone: (812) 877-8685

Email: bohner@rose-hulman.edu

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

An Engineering Decision…



Learning Outcomes: O-O Design

Demonstrate object-oriented design basics like domain models, class diagrams, and interaction (sequence and communication) diagrams.

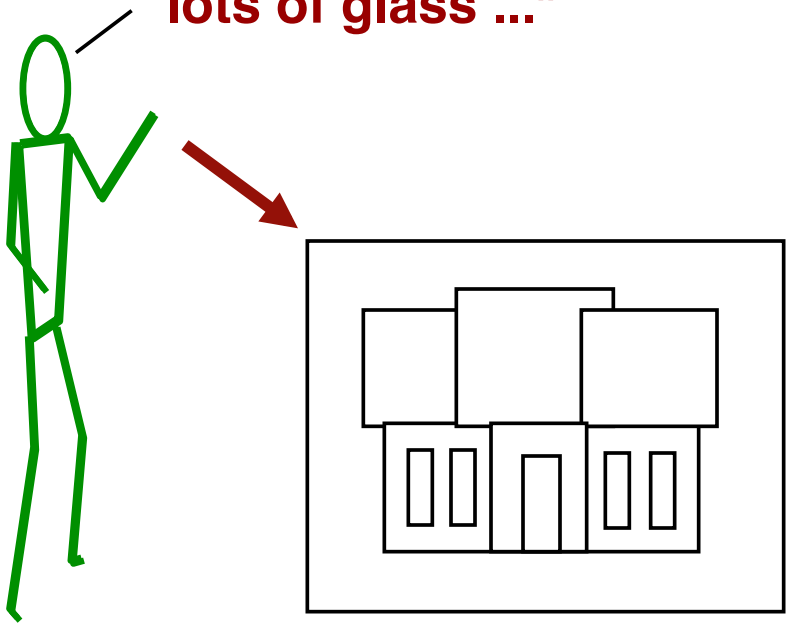
http://enterprisegeeks.com/blog/2009/07/

- Describe Software Architecture
- Explain why architecture is important
- Get into the layers…

From Requirements to Architecture

Customer Requirements

"four bedrooms, three baths, lots of glass ..."



Architectural Design

How do we get from here

…to there?

- **Think for 15.26667 seconds…**
- **Turn to a neighbor and discuss it for a minute**

Software Architecture Definitions

- The large-scale motivations, constraints, organization, patterns, responsibilities, and connections [between components] of the system.

Craig Larman 2003

- The structure or structures of the system, which comprise software components, the externally visible properties of those components, and the relationships among them.

Bass, et al, 1998

Architectural Building Blocks

Component – a unit of computation or a data store

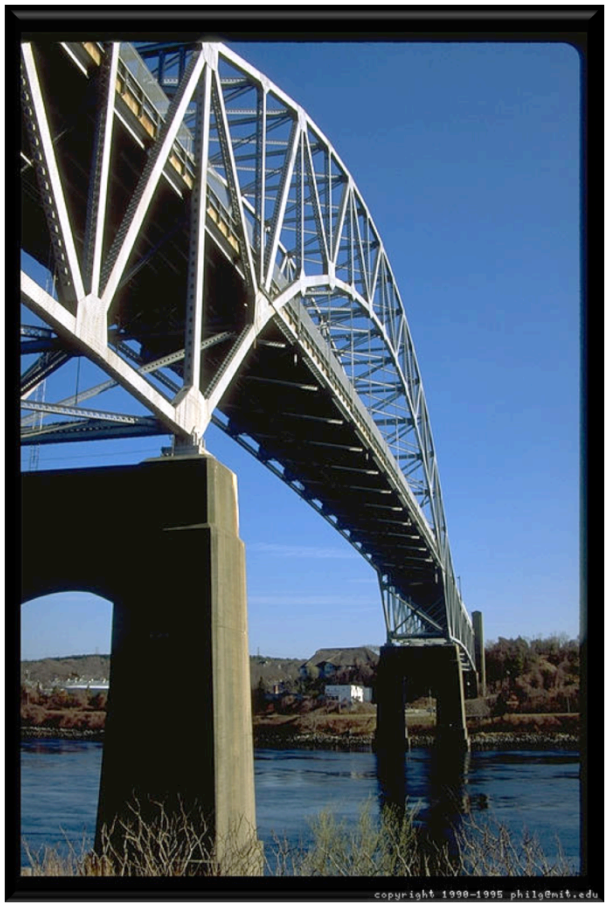
Connector – an architectural element that models interactions among components and rules that govern those interactions

Configuration (or topology) – a connected graph (composite) of components and connectors which describe architectural structure

Why Software Architecture?

Enables the software engineer to:

1. Analyze the effectiveness of the design in meeting its stated requirements
2. Consider architectural alternatives at a stage when making design changes is still relatively easy
3. Reduce the risks associated with the construction of the software
4. Provide key Abstractions in reasoning about design
5. Establish a Design Plan



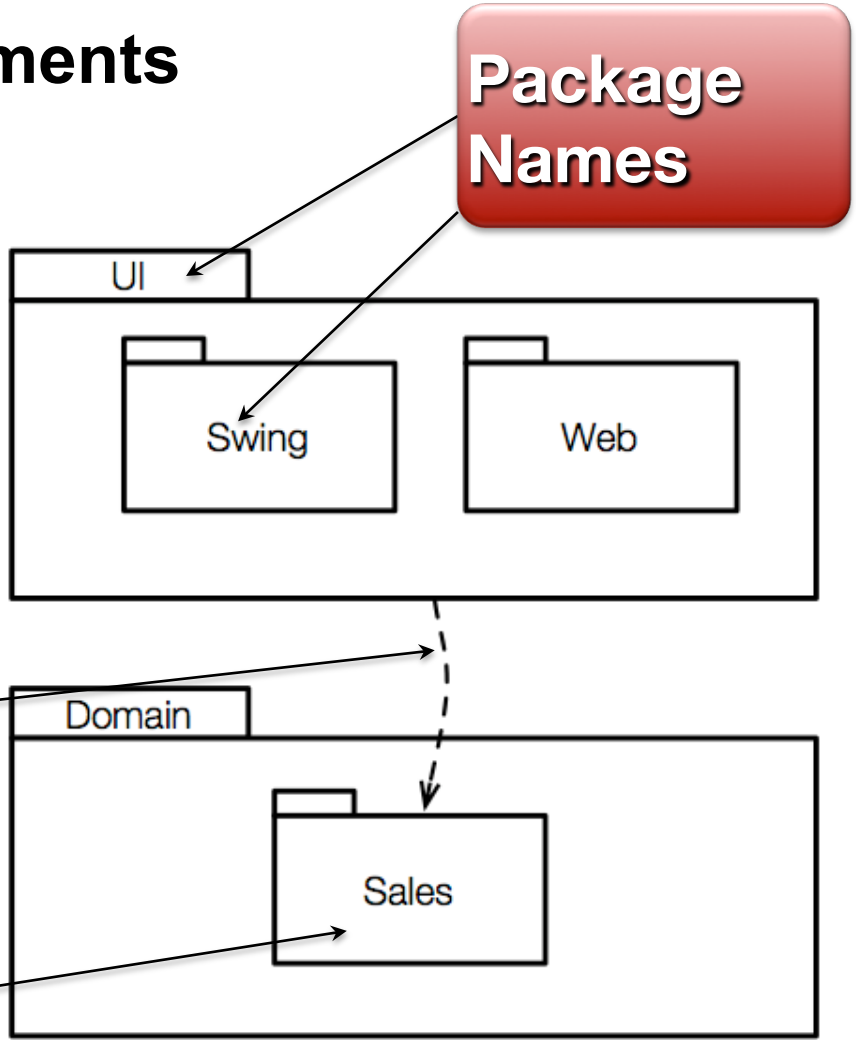
UML Architectural Views

- **Logical Architecture** – describes the system in terms of its organization in layers, packages, classes, interfaces and subsystems

- **Deployment Architecture** – describes the system in terms of the allocation of processes to processing units and network configurations

UML Package Diagrams

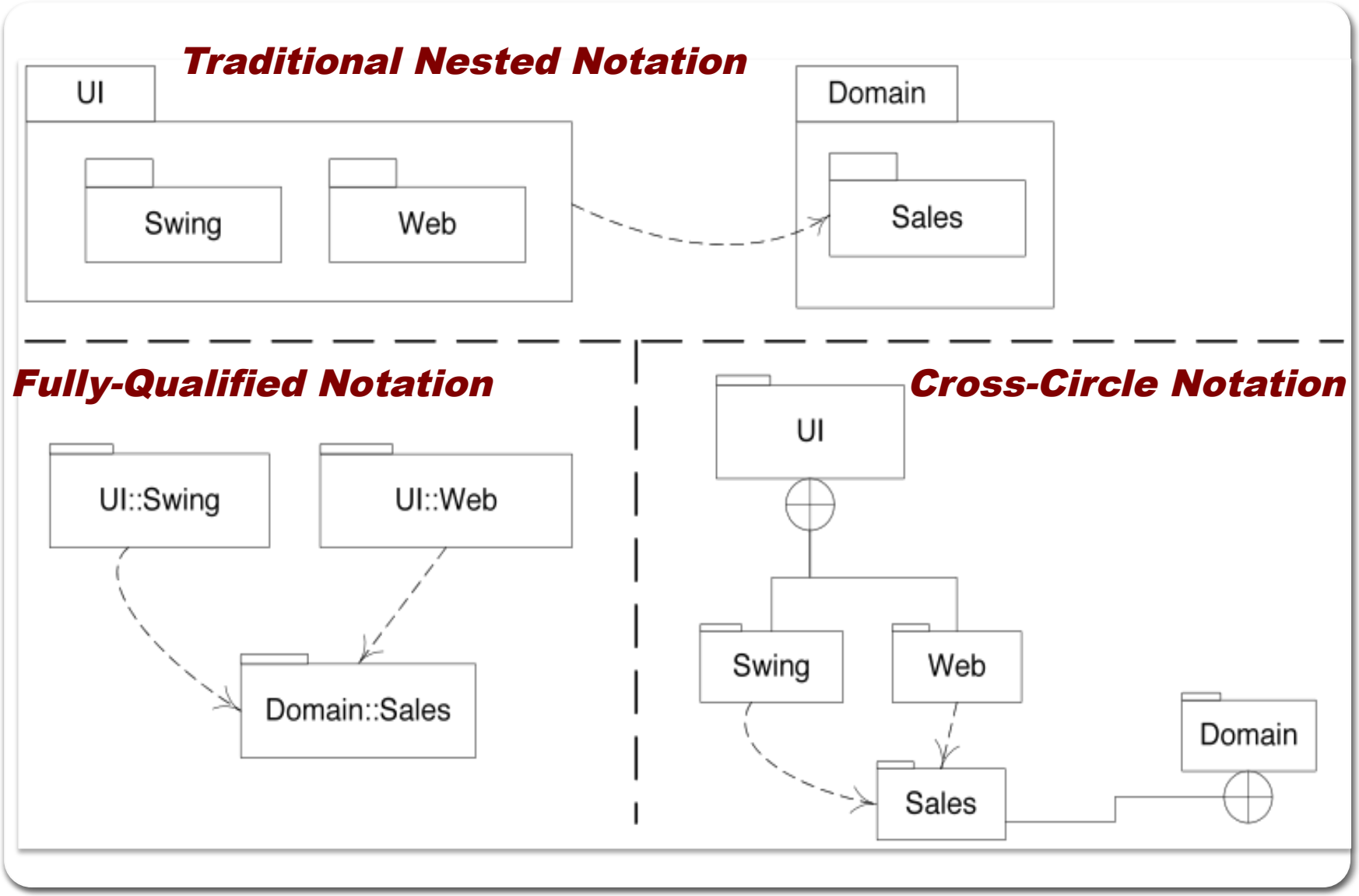
- Describes grouping of elements
- Can group anything:
  - Classes
  - Other packages
- More general than Java packages or C# namespaces



Dependency Line

Fully qualified name is:
Domain::Sales

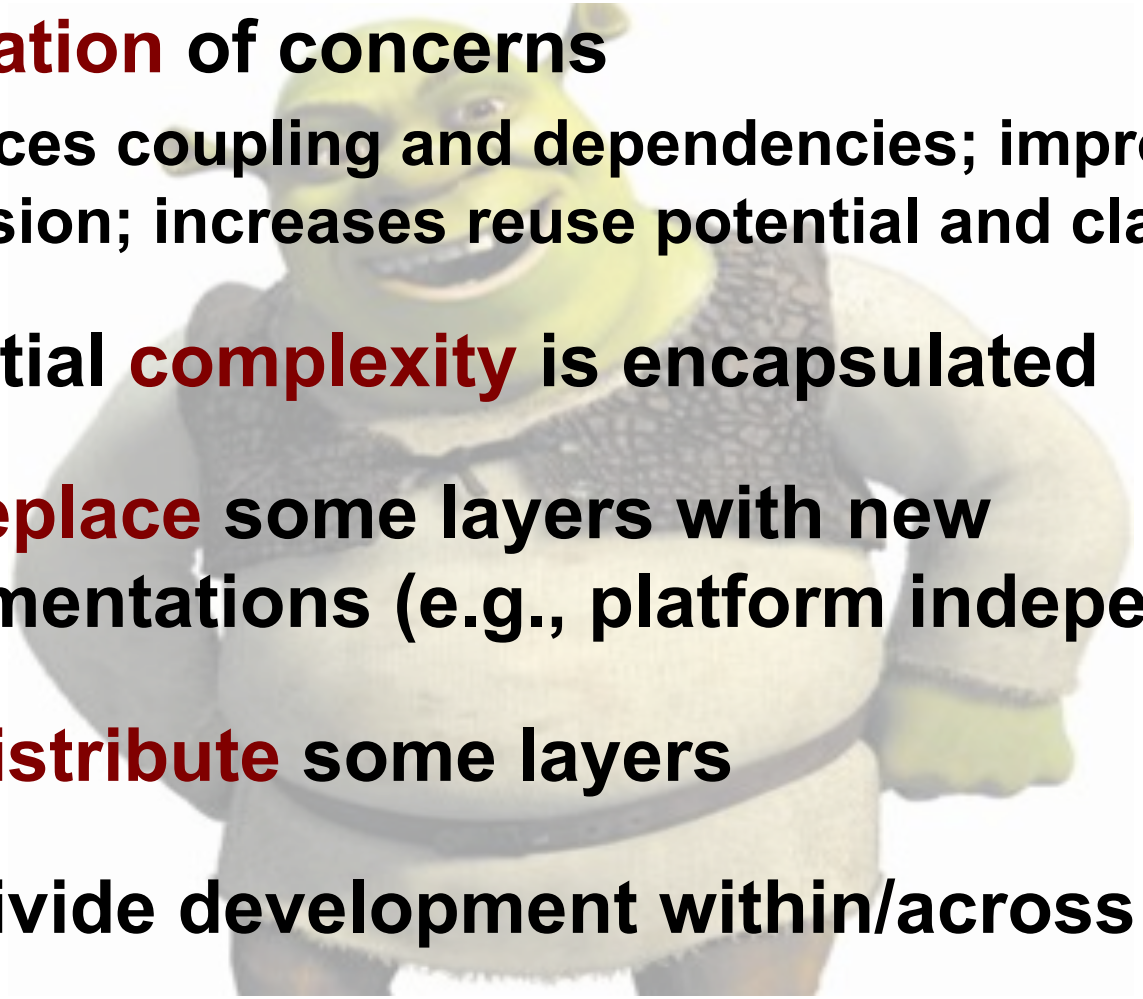
Alternative Nesting Notations



Designing with Layers Solves Problems

- Rippling source code changes
- Intertwining of application and UI logic
- Intertwining of application logic and technical services
- Difficult division of labor

Benefits of Architecture Layers



- Separation of concerns
  - Reduces coupling and dependencies; improves cohesion; increases reuse potential and clarity
- Essential complexity is encapsulated
- Can replace some layers with new implementations (e.g., platform independence)
- Can distribute some layers
- Can divide development within/across teams

Common Layers in More Detail (1 of 2)

GUI windows
reports
speech interface
HTML, XML, XSLT, JSP, Javascript, ...

UI
(AKA **Presentation**, View)

handles presentation layer requests
workflow
session state
window/page transitions
consolidation/transformation of disparate data for presentation

Application
(AKA Workflow, Process, Mediation, App Controller)

handles application layer requests
implementation of domain rules
domain services (*POS*, *Inventory*)
- services may be used by just one application, but there is also the possibility of multi-application services

Domain
(AKA Business, Application Logic, Model)

very general low-level business services used in many business domains
CurrencyConverter

Business Infrastructure
(AKA Low-level Business Services)

Common Layers in More Detail (2 of 2)

very general low-level business services used in many business domains
CurrencyConverter

Business Infrastructure
(AKA Low-level Business Services)

(relatively) high-level technical services and frameworks
Persistence, *Security*

Technical Services
(AKA Technical Infrastructure,
High-level Technical Services)

low-level technical services, utilities, and frameworks
data structures, threads, math, file, DB, and network I/O

Foundation
(AKA Core Services, Base Services,
Low-level Technical Services/Infrastructure)

Systems will have many, but not necessarily all, of these

Exercise on Logical Architecture

- Break up into your project teams

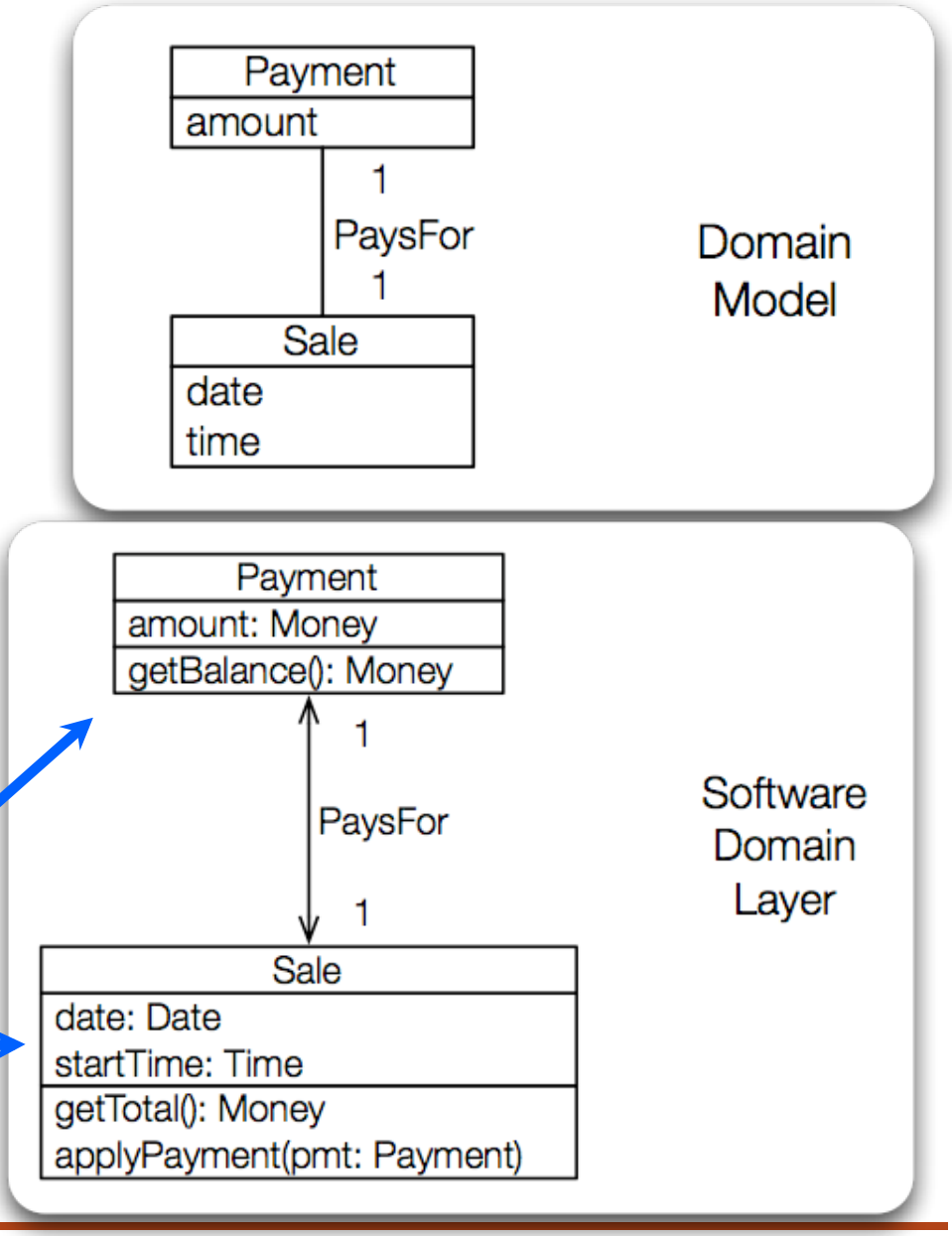
- Given the following packages:
  - Rental Process GUI, Rental
  - Provision GUI, Provision
  - Payment GUI, Payment
  - Membership, Security
  - Persistence, Directory Services

- Draw a BBVS Logical Architecture diagram with the UI, Domain, and Technical Services Layers

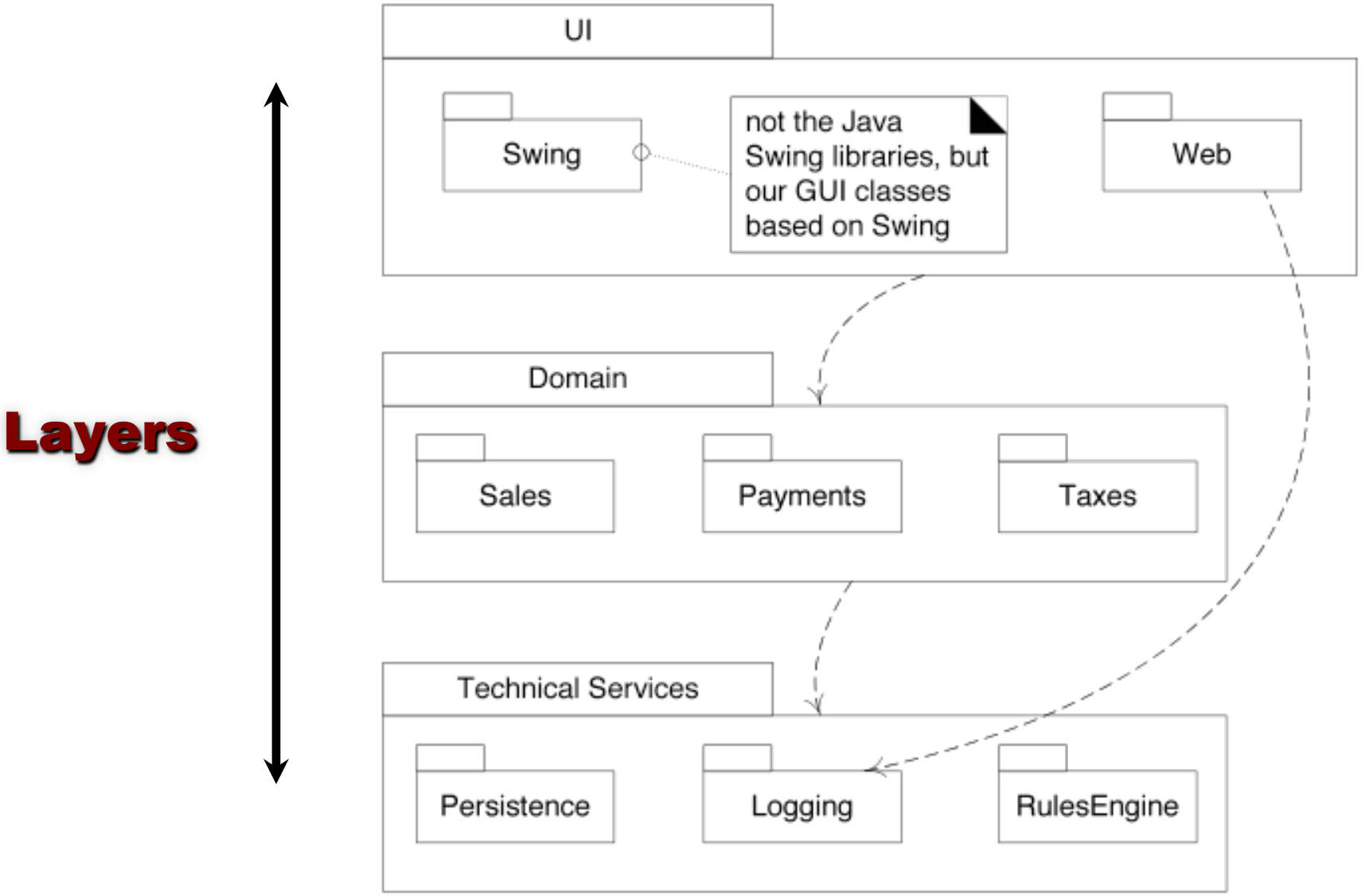
Designing the Domain Layer

- Create software objects with names and information similar to the real-world domain
- Assign application logic responsibilities

"Domain Objects"

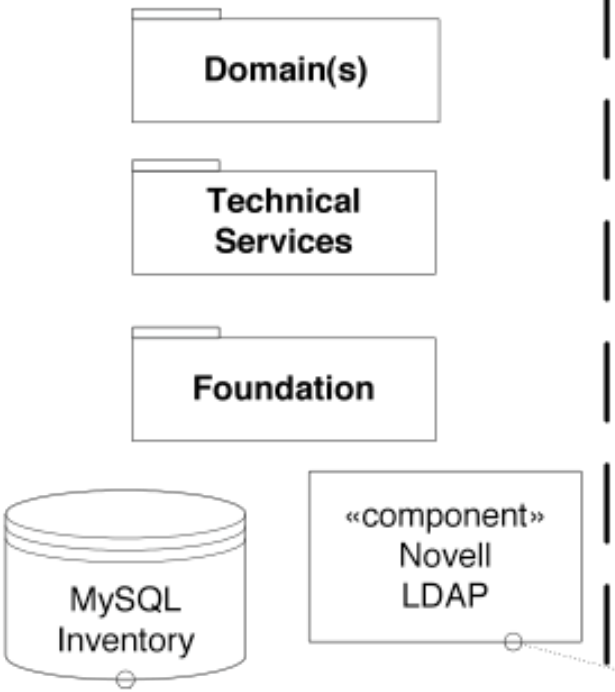


Terminology: Layers vs. Partitions

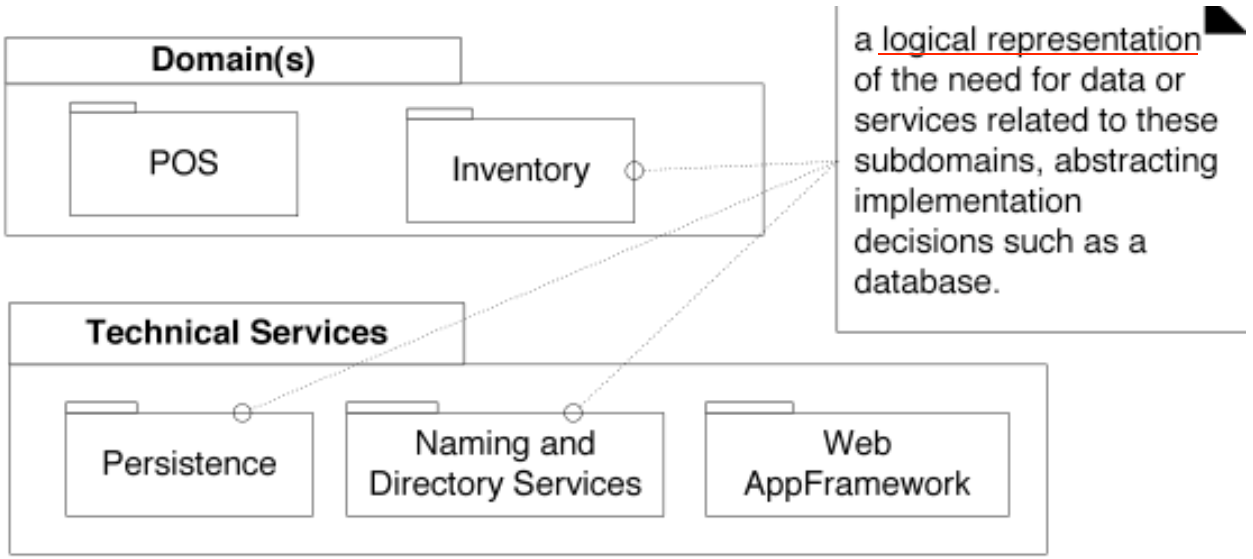


Common Mistake: Showing External Resources

Worse



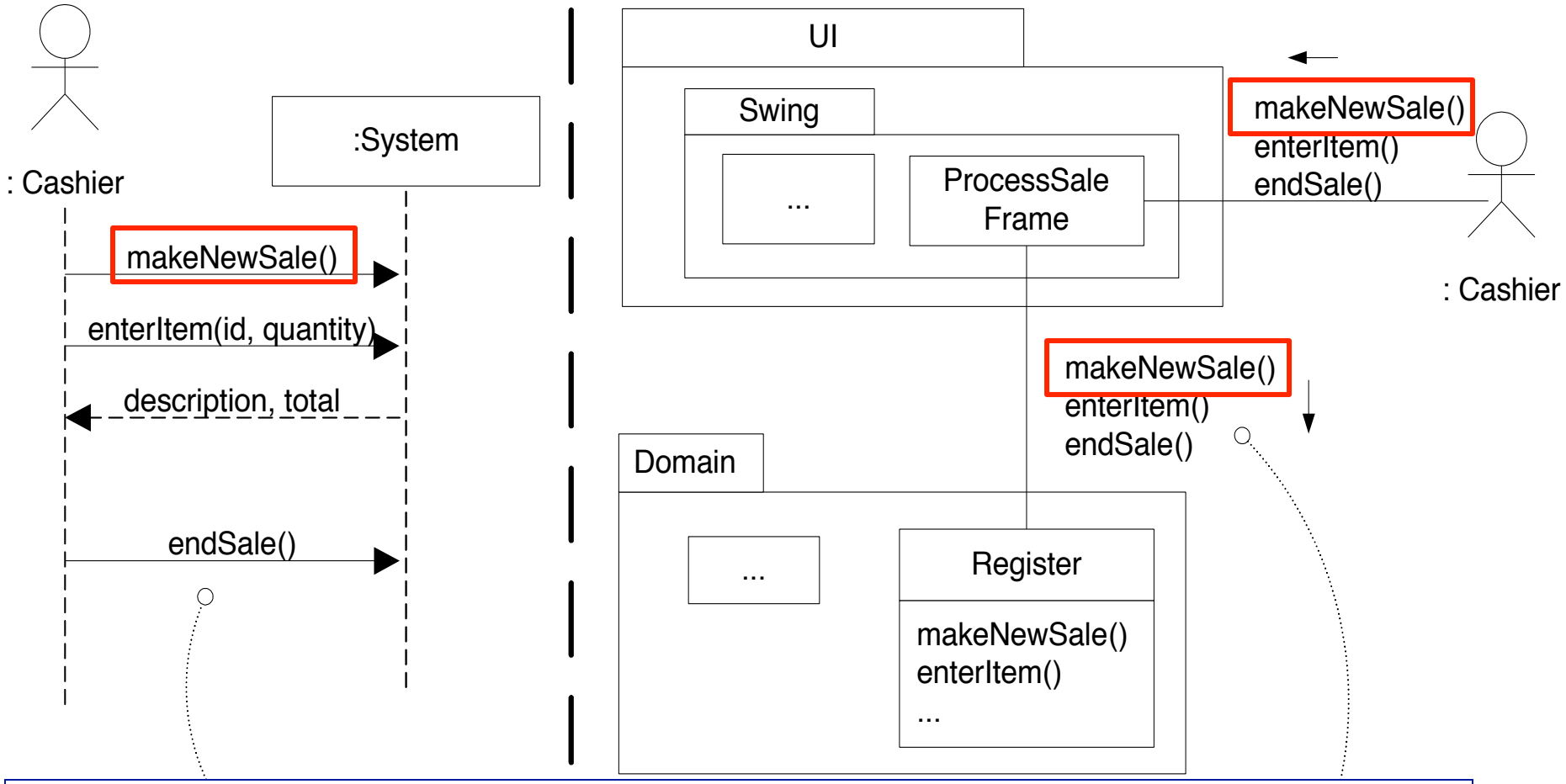
Better



Model-View Separation Principle

- Do not connect non-UI objects directly to UI objects
  - E.g., A Sale object shouldn't have a reference to a UI object (e.g., Jframe)
- Do not put application logic in UI object methods
  - A UI event handler should just delegate to the domain layer
- Note: Model → domain layer;
  View → UI layer

From SSDs to Layers



System operations on the SSDs will become the messages sent from the UI layer to the domain layer

Common Object Design Techniques

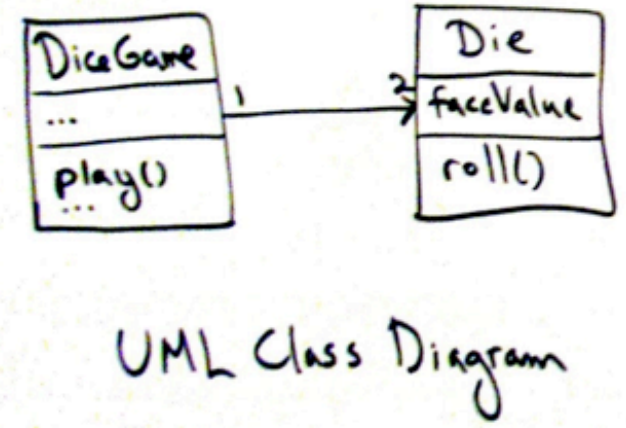
- **Just code it:** design while coding, heavy emphasis on refactoring and powerful IDEs
- **Draw, then code:** sketch some UML, then code it
- **Just draw it:** generate code from diagrams

http://www.virginmedia.com/movies/galleries/previews/indiana-jones-idols.php?ssid=7

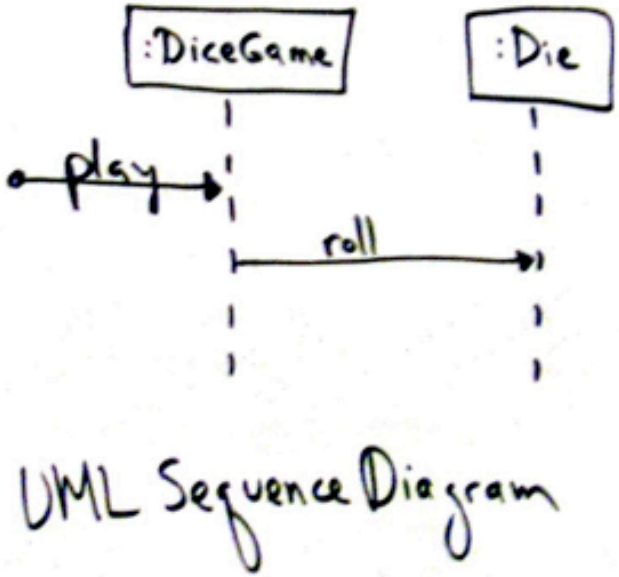
Static vs. Dynamic Modeling

- Static models
  - Class diagrams
- Dynamic models
  - Sequence diagrams
  - Communication diagrams

Interaction Diagrams



UML Class Diagram

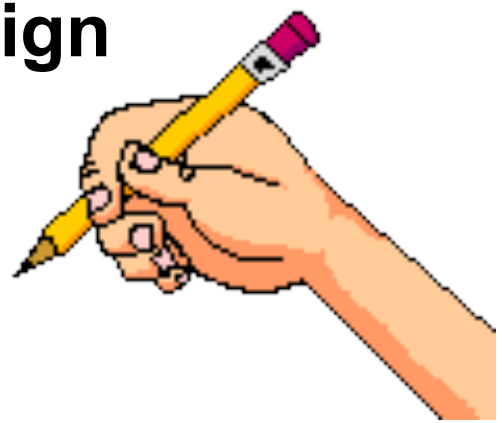


UML Sequence Diagram

Spend time on interaction diagrams, not just class diagrams

Prefer Design Skill over UML skill

- UML is only a tool for object design



- The real skill is the design,
  …NOT the diagramming
- Fundamental object design requires knowledge of:
  - Principles of responsibility assignment
  - Design patterns

Homework and Milestone Reminders

- Read Chapter 15 on Interaction Diagrams
- Homework 2 – Video Store SSDs and Operations Contracts
  - Due by 5:00pm on Tuesday, December 14th, 2010
- Milestone 3 – Junior Project SSDs, OCs, and Logical Architecture
  - Finish Analysis Model (SSDs, OCs)
  - Logical Architecture - Package Diagrams, and
  - 1st (initial) Version of System
  - Due by 11:59pm on Friday, January 7th, 2010