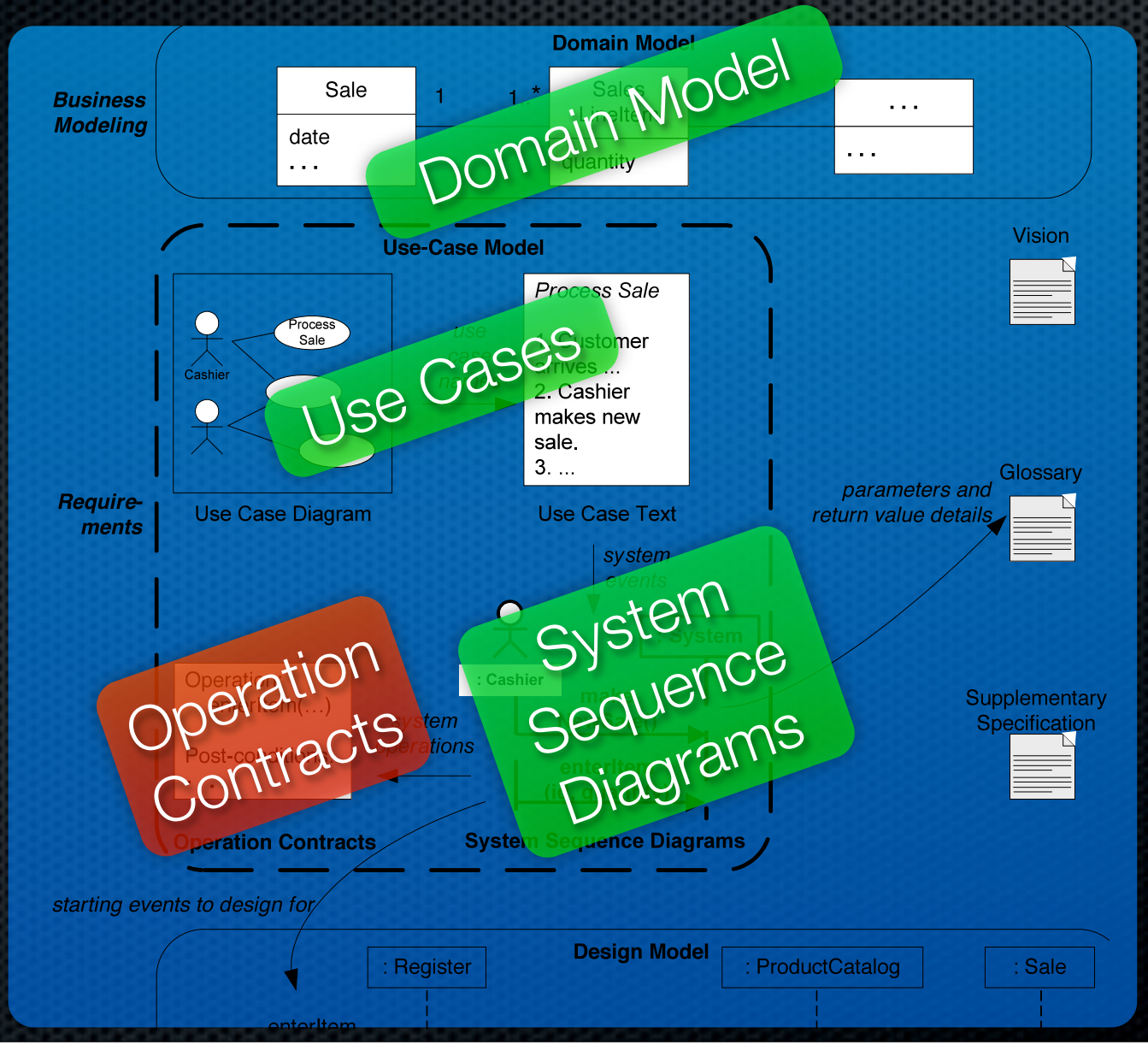
Operation Contracts and From Analysis to Design

Curt Clifton

Rose-Hulman Institute of Technology

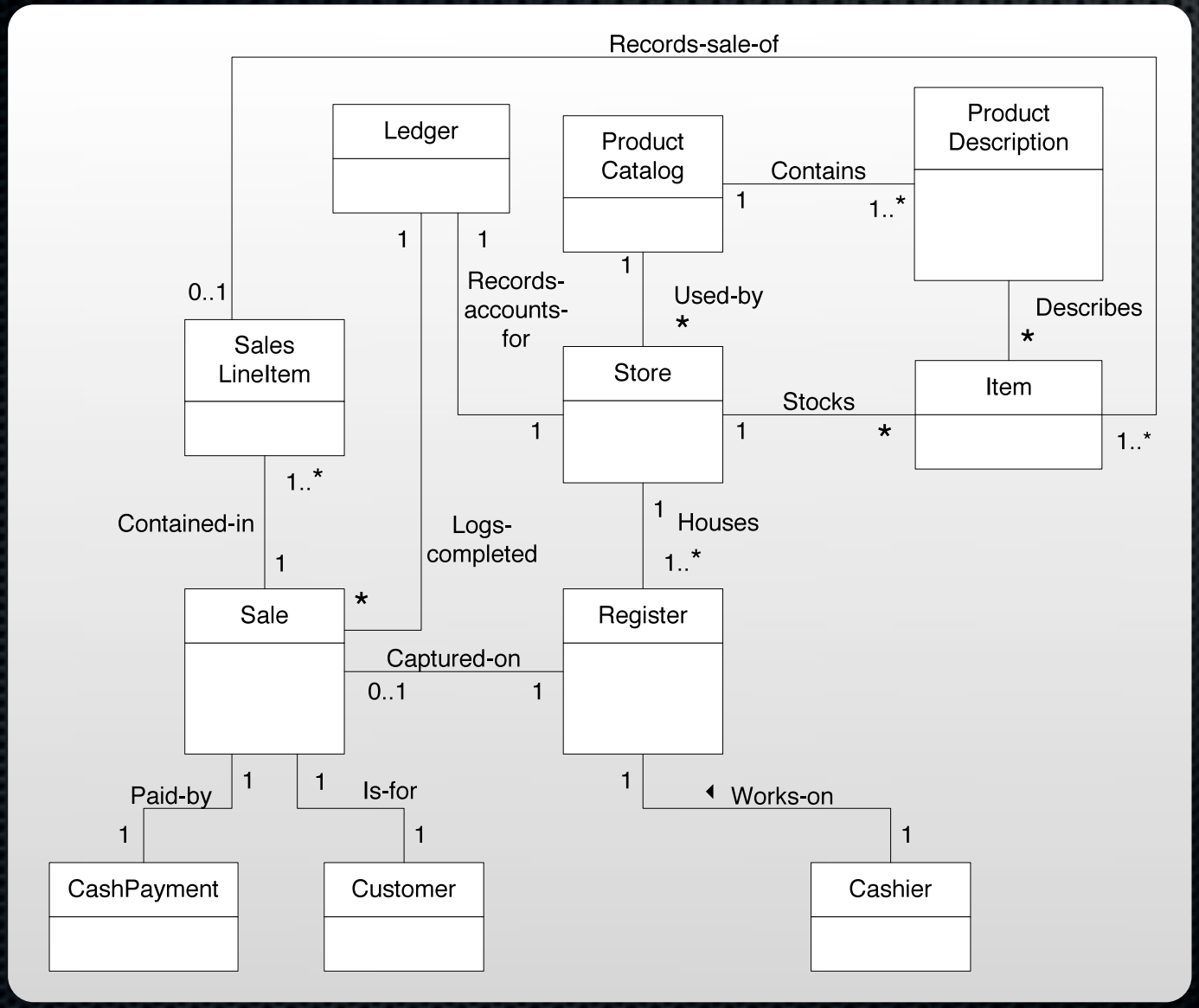
No reading quiz questions today

Where Are We?


Domain Model
Business Modeling
Sale
date
. . .
1
1 *
. . .
. . .
Domain Model
Use-Case Model
Process Sale
Cashier
Use Case Diagram
Process Sale
2. Cashier makes new sale.
3. ...
Use Case Text
Use Cases
Vision
Glossary
parameters and return value details
Require-ments
system events
: Cashier
Operation Contracts
System Sequence Diagrams
Supplementary Specification
starting events to design for
Design Model
: Register
: ProductCatalog
: Sale

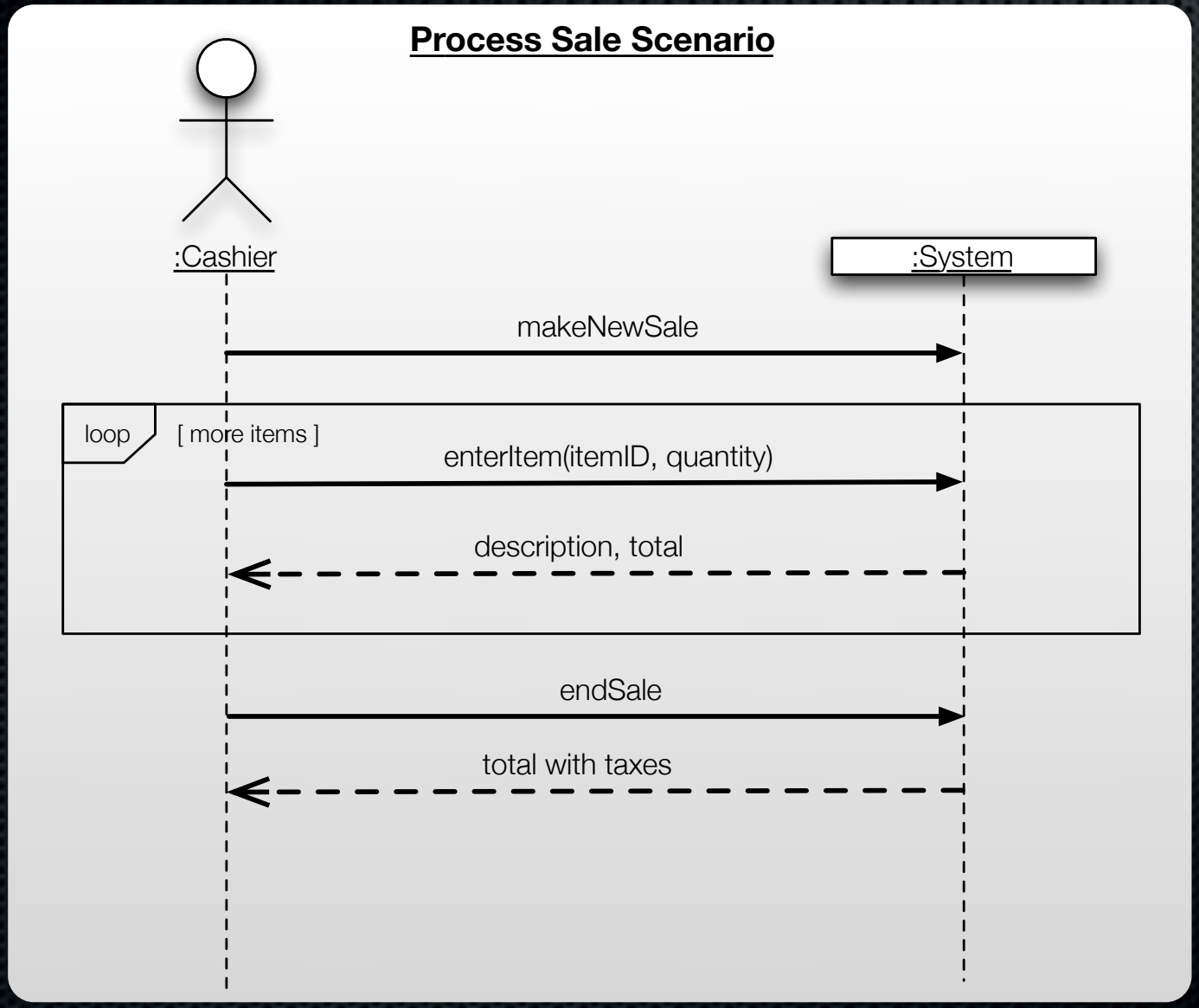
From Functional Use Cases to Object-Oriented System

Domain Model



Conceptual
Classes
Associations
Attributes

System Sequence Diagrams



System Events

Operation Contracts

From SSDs, messages coming into the system

- Used to give more details for **system operations**
- Together, all the system operations from all the use cases give the public **system interface**

Conceptually, it's like the whole system is a single object and the system operations are its public methods

Example

(At most) one OC per System Operation

Any uses cases where this operation appears

Contract CO2: enterItem

Operation:	enterItem(itemID: ItemID, quantity: Integer)
Cross Refs:	Use Cases: Process Sale
Preconditions:	There is a sale underway
Postconditions:	• a *SalesLineItem* instance, *sli*, was created • *sli* was associated with the current *Sale* • *sli.quantity* became *quantity* • *sli* was associated with a *ProductDescription* based on *itemID* match

Noteworthy assumptions

Most important section

Q1,2

Details

Postconditions

- Describe changes in the state of objects in the domain model
- Typical sorts of changes:
  - Created instances
  - Form associations
  - Break associations
  - Change attributes

Not actions performed during the operation. Rather, **observations about what is true** after the operation.

Q3,4

Postconditions

- Describe changes in the state of objects in the domain model
- Typical sorts of changes:
  - Create instances
  - Form associations
  - Break associations
  - Change attributes

- a *SalesLineItem* instance, *sli*, was created
- *sli* was associated with the current *Sale*
- *sli.quantity* became *quantity*
- *sli* was associated with a *ProductDescription* based on *itemID* match

Postconditions

- Use past tense
- Give names to instances
- Capture the information from the system operation parameters by noting changes to domain objects
- Can be (somewhat) informal

- a *SalesLineItem* instance, *sli*, was created
- *sli* was associated with the current *Sale*
- *sli.quantity* became *quantity*
- *sli* was associated with a *ProductDescription* based on *itemID* match

Q5

When to Use Operation Contracts

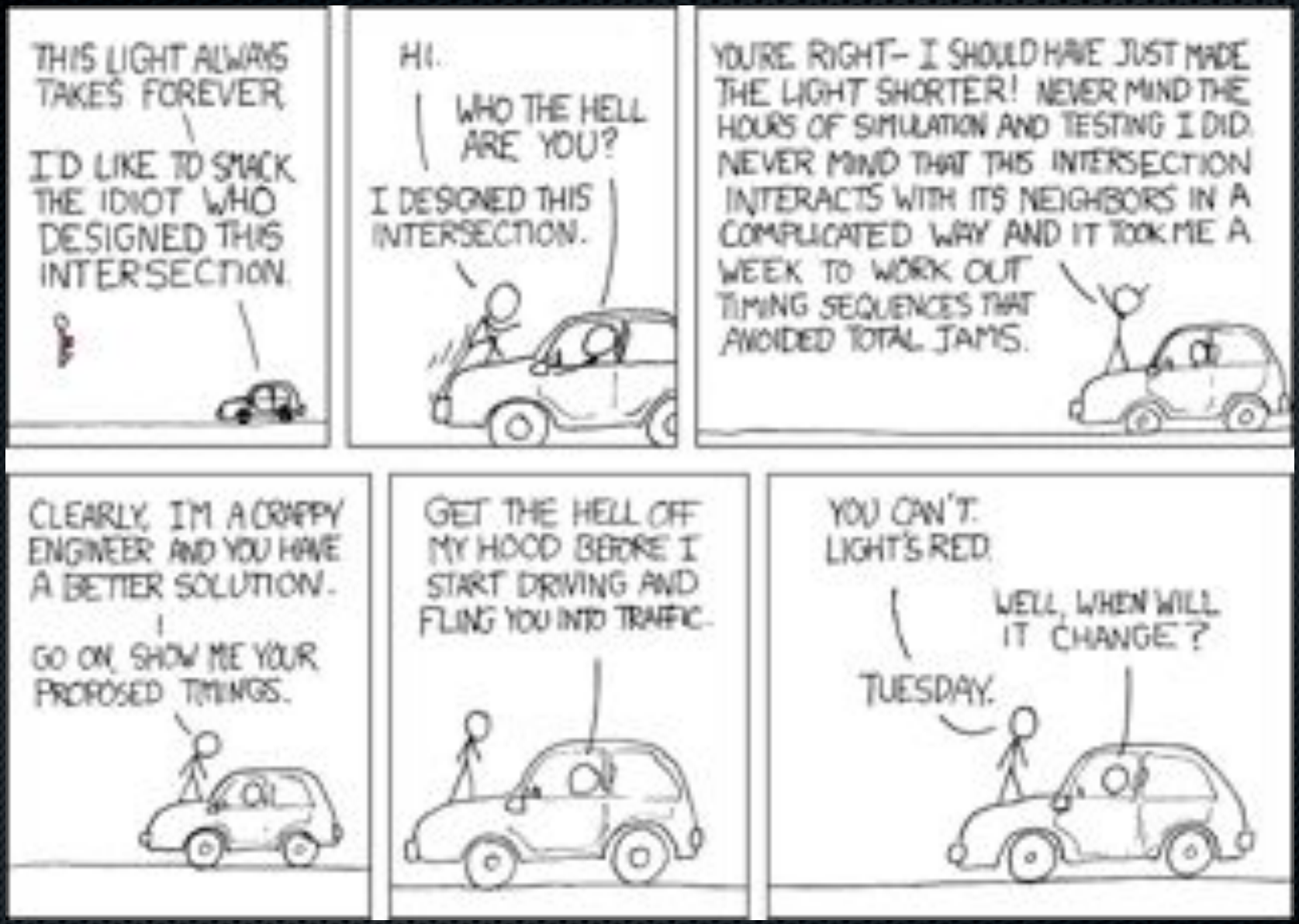
- **Not always**!
- When detail and precision are important:
  - When details would make use cases too verbose
  - When we don't know the domain and want a **deeper analysis** (while deferring design)
- To help **validate** the domain model
- To **associate** system operations with particular objects

Creating Operation Contracts

- Identify system operations from SSDs
- Identify system operations that **warrant** OCs
  - Complex, subtle, or unclear from use case
- Make sure postconditions consider:
  - Created instances
  - Formed associations
  - Broken associations
  - Changed attributes

Example....

Q7



You can look at practically any part of anything manmade around you and think ‘some engineer was frustrated while designing this.’ It's a little human connection.

From Requirements to Design

Recall…

- **Analysis**: Do the right thing
- **Design**: Do the thing right

Leaving Analysis Behind?

Unknown/unusual activities are high risk

- Not really
- We'll learn more about the problem while designing (and implementing) a solution
  - **Refine the requirements** when that happens
  - Choose **high risk** activities for early iterations to **provoke changes** to the requirements
- "Just enough" analysis is often useful

Logical Architecture

A very short introduction

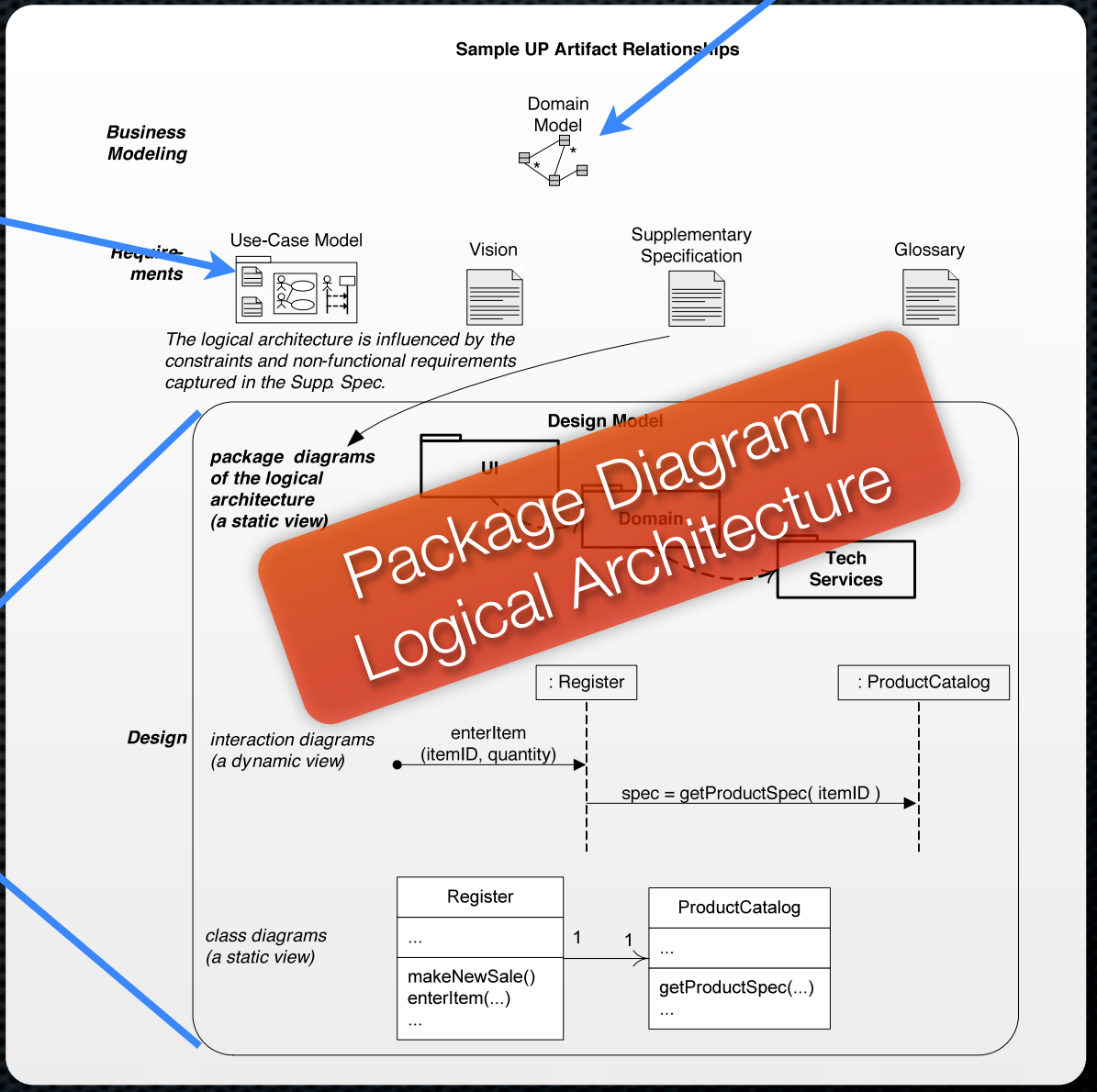
www.lostateminor.com

Where Are We?

Domain Model

Use Case Model including System Sequence Diagrams and Operation Contracts

Design Model



Logical Architecture

- Large-scale organization of the **software** classes into:
  - Packages (a.k.a., namespaces)
  - Subsystems
  - Layers
- **Logical**, since implementation/deployment decisions are deferred

Layered Architectures

- **Very common** for object-oriented systems
- **Coarse-grained grouping** of components based on **shared responsibility** for major aspects of system
- Typically **higher layers call lower ones**, but not vice-versa

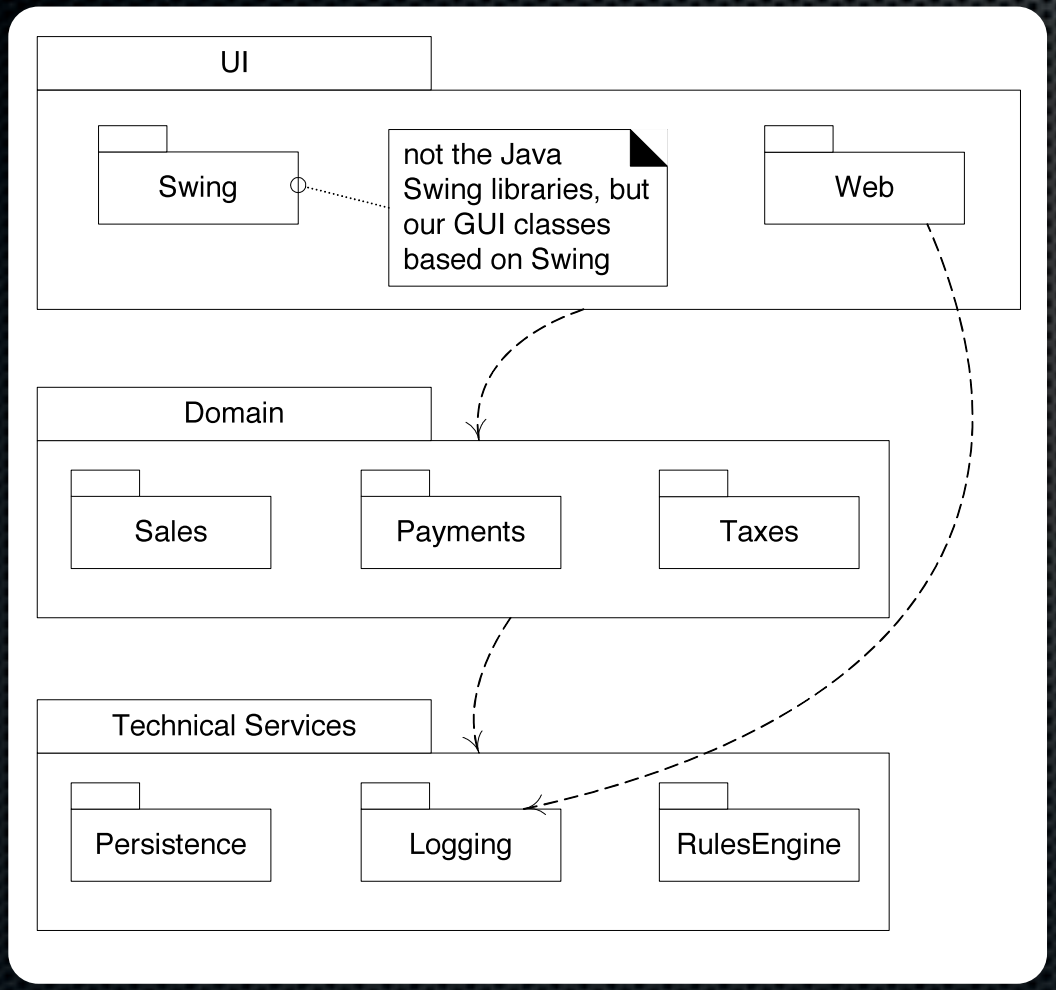
Three Typical Layers

- User Interface
- Application Domain Layer
- Technical Services:
  - Persistence
  - Logging
  - Rules Engine

Heavily influenced by domain model

Reusable across systems

Strict vs. Relaxed Layered Architectures



- Strict: only calls next layer down
- Relaxed: can call any layer below

Q10