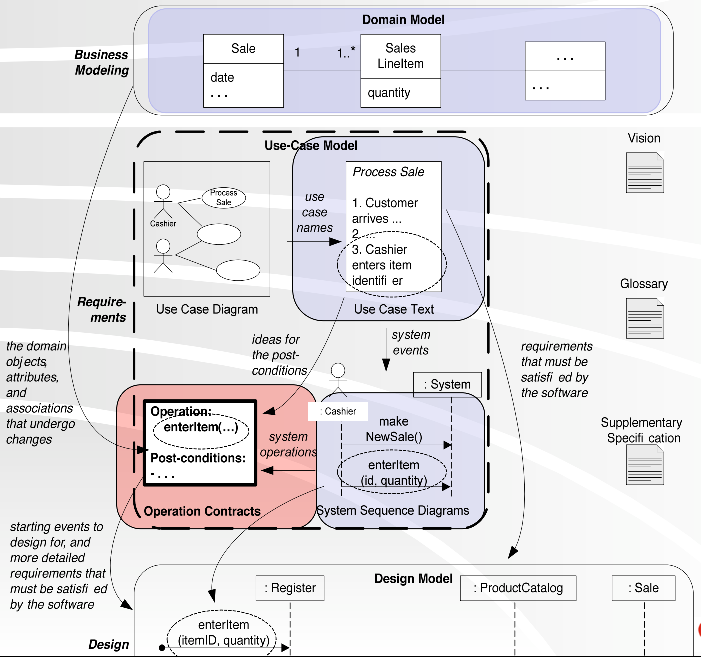
# Operations Contracts and Preliminaries on Design

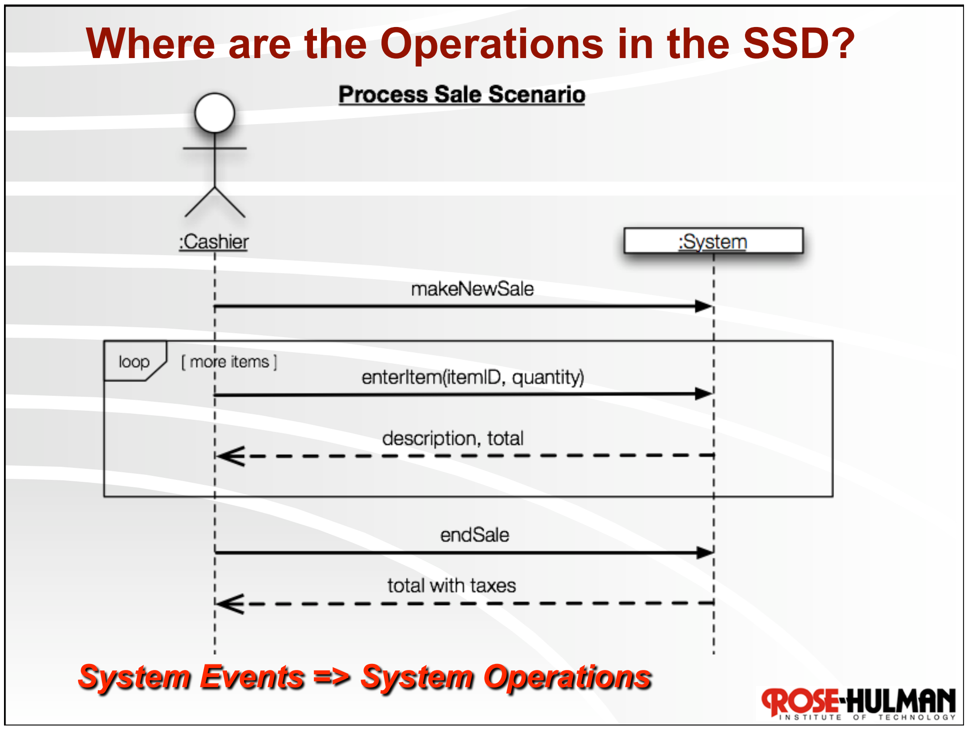
## *CSSE 374: Session 7*

**Shawn Bohner**
**Office: Moench Room F212**
**Phone: (812) 877-8685**
**Email: bohner@rose-hulman.edu**

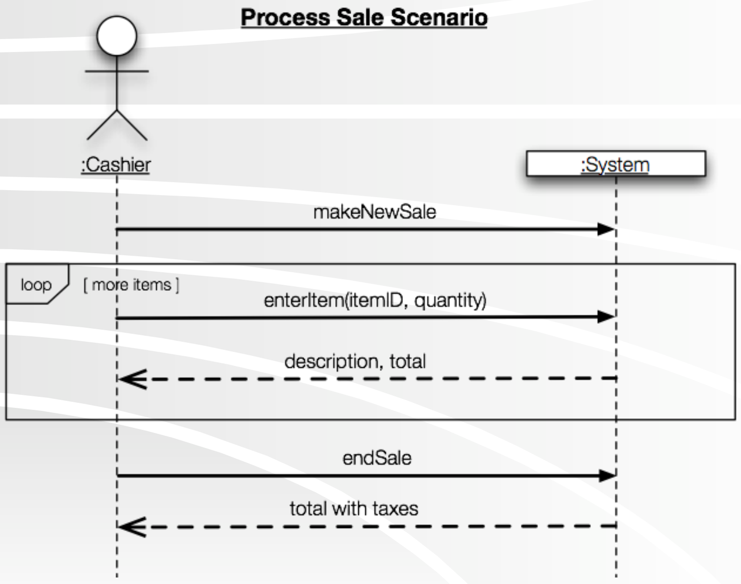
© 2009 Shawn A. Bohner

## We are at System Operation Contracts

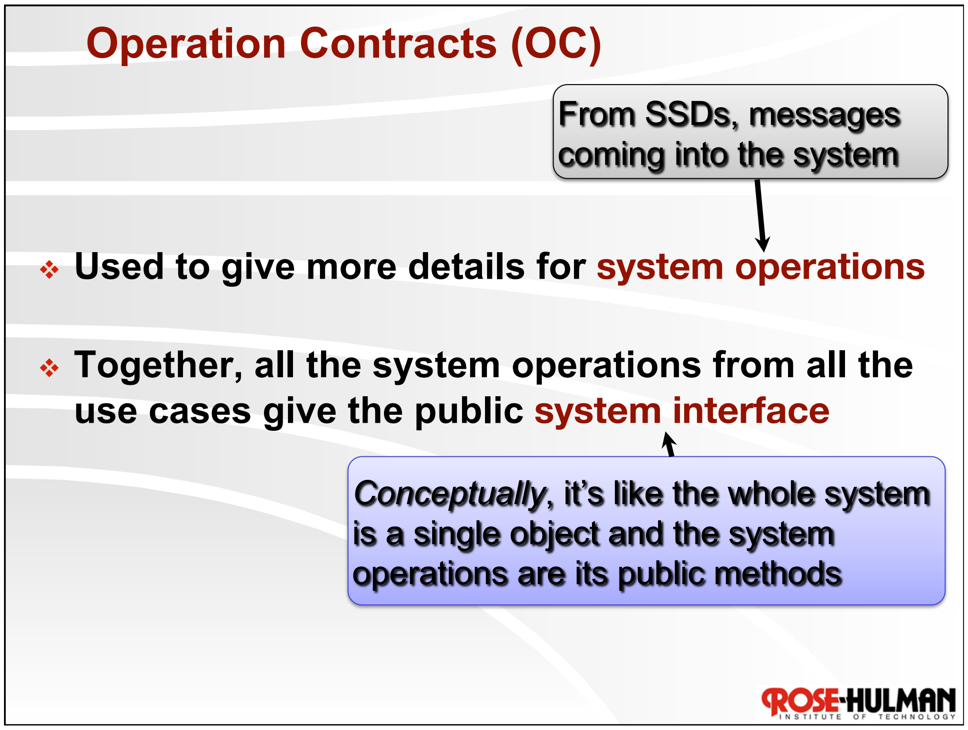




## Where are the Operations in the SSD?



***System Events => System Operations***

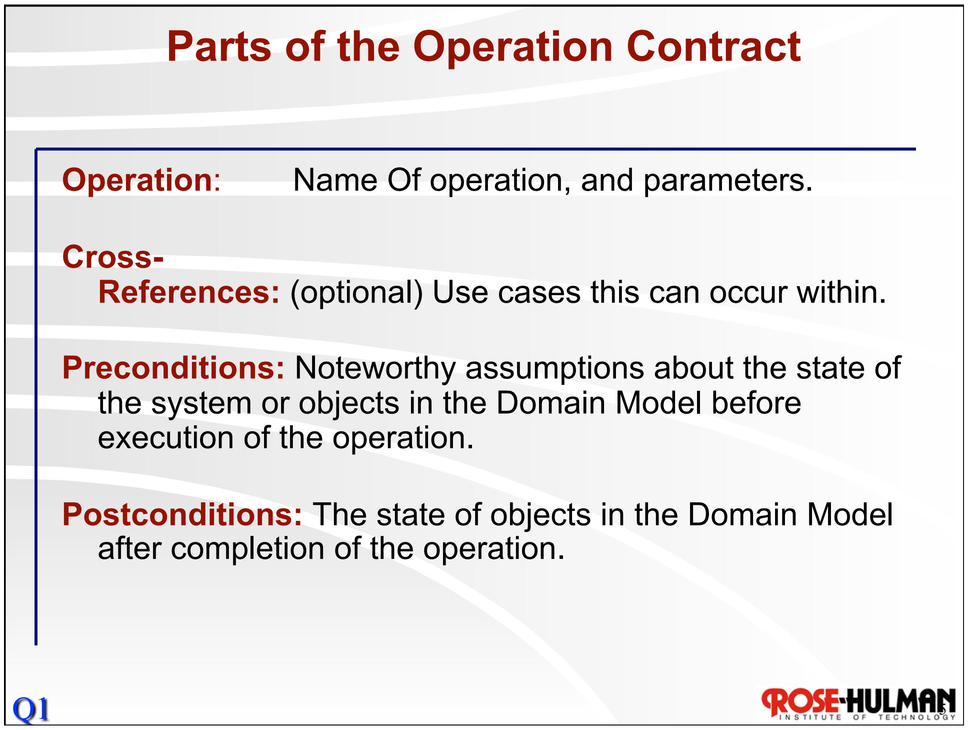


## Operation Contracts (OC)

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



## Parts of the Operation Contract

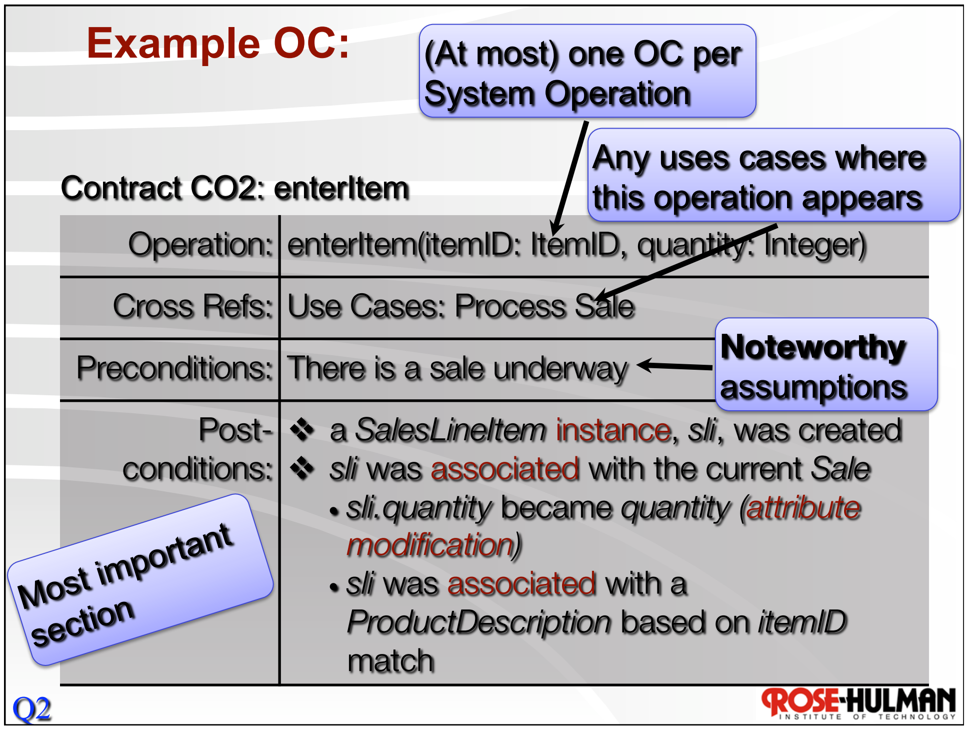
**Operation:** Name Of operation, and parameters.

**Cross-References:** (optional) Use cases this can occur within.

**Preconditions:** Noteworthy assumptions about the state of the system or objects in the Domain Model before execution of the operation.

**Postconditions:** The state of objects in the Domain Model after completion of the operation.

Q1



## Example OC:

(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 |
| Post-conditions: | ❖ a *SalesLineItem* instance, *sli*, was created<br>❖ *sli* was associated with the current *Sale*<br>• *sli.quantity* became *quantity* (*attribute modification*)<br>• *sli* was associated with a *ProductDescription* based on *itemID* match |

**Noteworthy assumptions**

Most important section

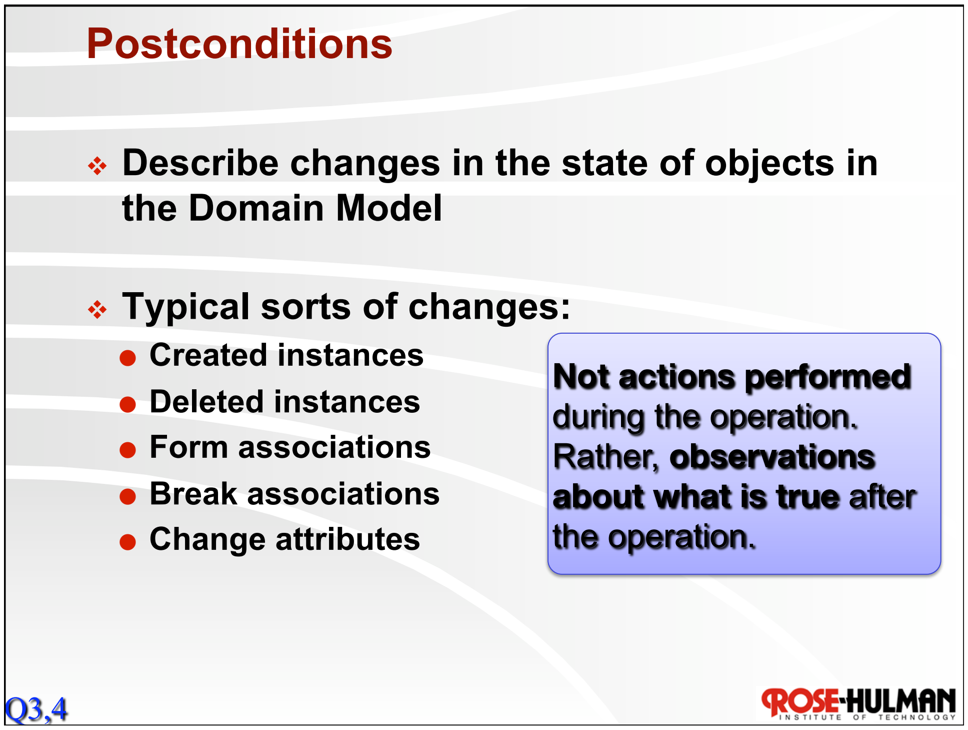
Q2

## Pre & Post-Conditions in Your Minds Eye

- Envision the system and it’s objects on an Extreme Makeover set…
- Before the operation, take a picture of the set
- The lights go out, and apply the system operation
- Lights on and take the after picture
- Compare the before and after pictures, and describe state changes as post-conditions

## Pre- and Post-Conditions

- Pre-Conditions are what must be in place to invoke the operation
- Post-conditions are declarations about the Domain Model objects that are true when the operation has finished



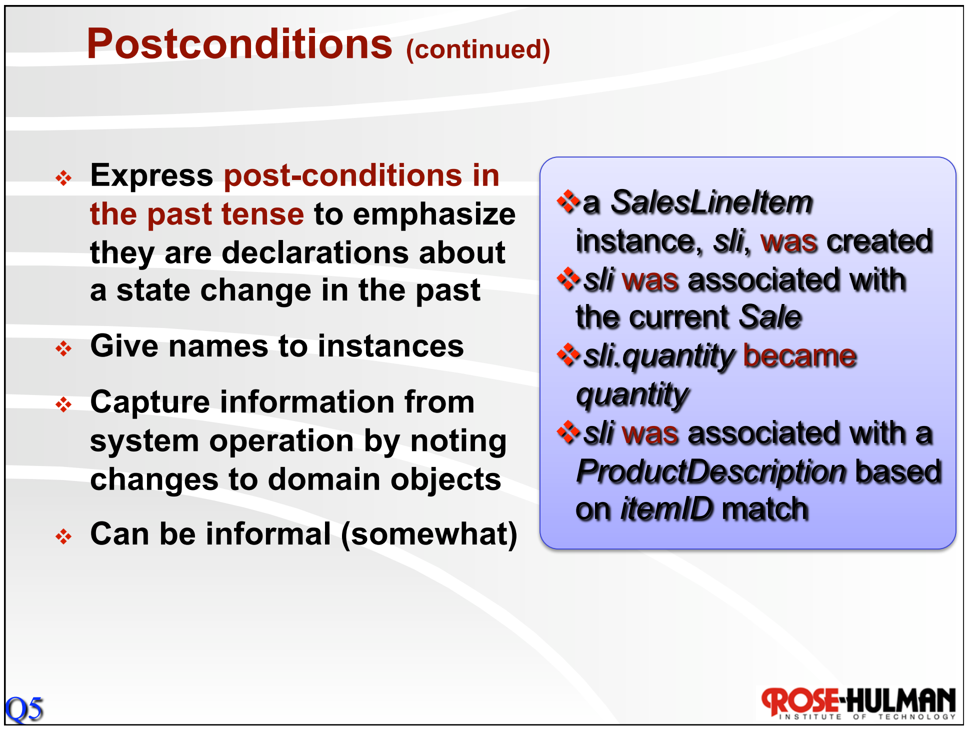
## Postconditions

- Describe changes in the state of objects in the Domain Model
- Typical sorts of changes:
  - Created instances
  - Deleted instances
  - Form associations
  - Break associations
  - Change attributes

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

Q3,4

ROSE-HULMAN INSTITUTE OF TECHNOLOGY



## Postconditions (continued)

- Express post-conditions in the past tense to emphasize they are declarations about a state change in the past
- Give names to instances
- Capture information from system operation by noting changes to domain objects
- Can be informal (somewhat)

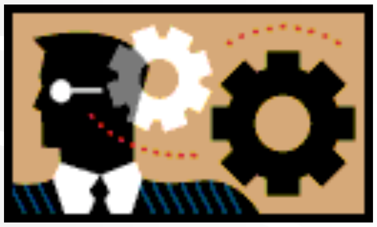
- a *SalesLineItem* instance, *sli*, was created
- *sli* was associated with the current *Sale*
- *sli.quantity* became *quantity*
- *sli* was associated with a *ProductDescription* based on *itemID* match

Q5

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

## Why Operation Contract Post-Conditions?

- Domain model
  =>objects attributes and associations
- The OC links a system operation to specific objects in the domain model



- Indicates which objects are affected by the operation
- Will help with assignment of responsibilities

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

## Contracts Lead to Domain Model Updates

New Domain Model classes, attributes, and associations are often discovered while writing contracts

Elaborate Domain Model as you think through the operation contracts

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

## Use Operation Contracts 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)

OCs help to validate the domain model

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

## Creating Operation Contracts

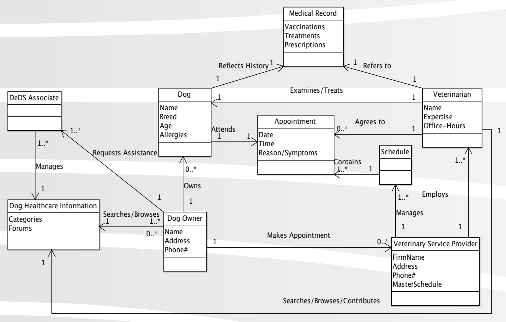
- Identify System Operations from SSDs
- Make contracts for System Operations that are:
  - Complex and perhaps subtle in their own results
  - Not clear in the use case
- Again, in describing post-conditions use:
  - Instance creation and deletion
  - Attribute modification
  - Associations formed and broken

Most frequent mistake in creating contracts:
Forgetting to include forming of associations

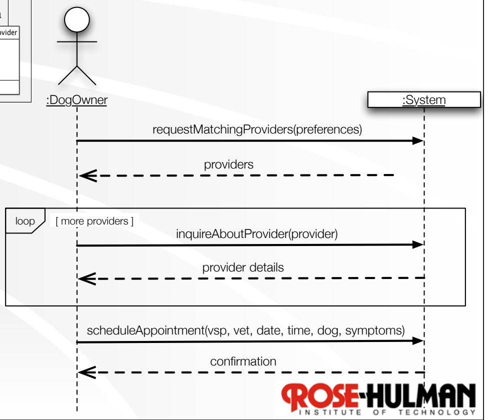
Q6

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

## Let's do an Example…



SEARCH, SERVICE INQUIRY, AND SCHEDULE APPOINTMENT


:DogOwner
:System
requestMatchingProviders(preferences)
providers
loop
[ more providers ]
inquireAboutProvider(provider)
provider details
scheduleAppointment(vsp, vet, date, time, dog, symptoms)
confirmation

Q7

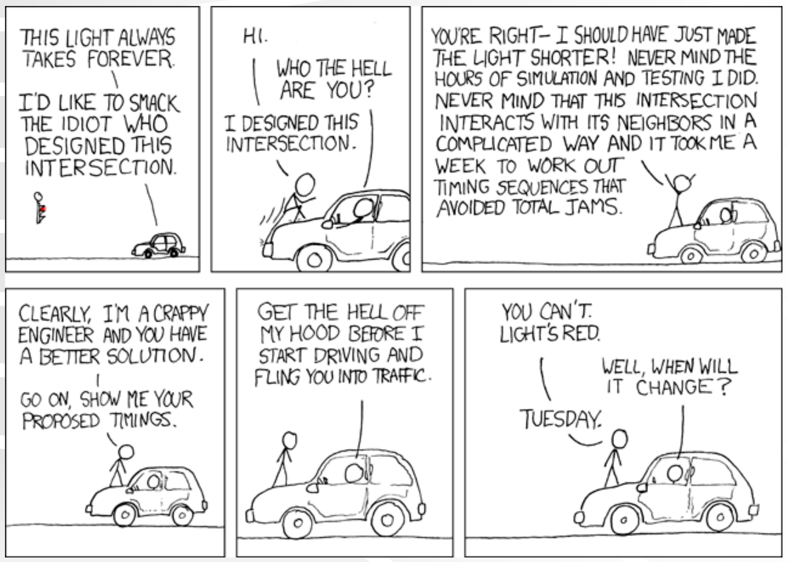
## Exercise: Complete this OC

**Operation: scheduleAppointment(vsp, vet, date, time, dog, symptoms)**

**Cross references: Use Cases: SEARCH, SERVICE INQUIRY, AND SCHEDULE APPOINTMENT**

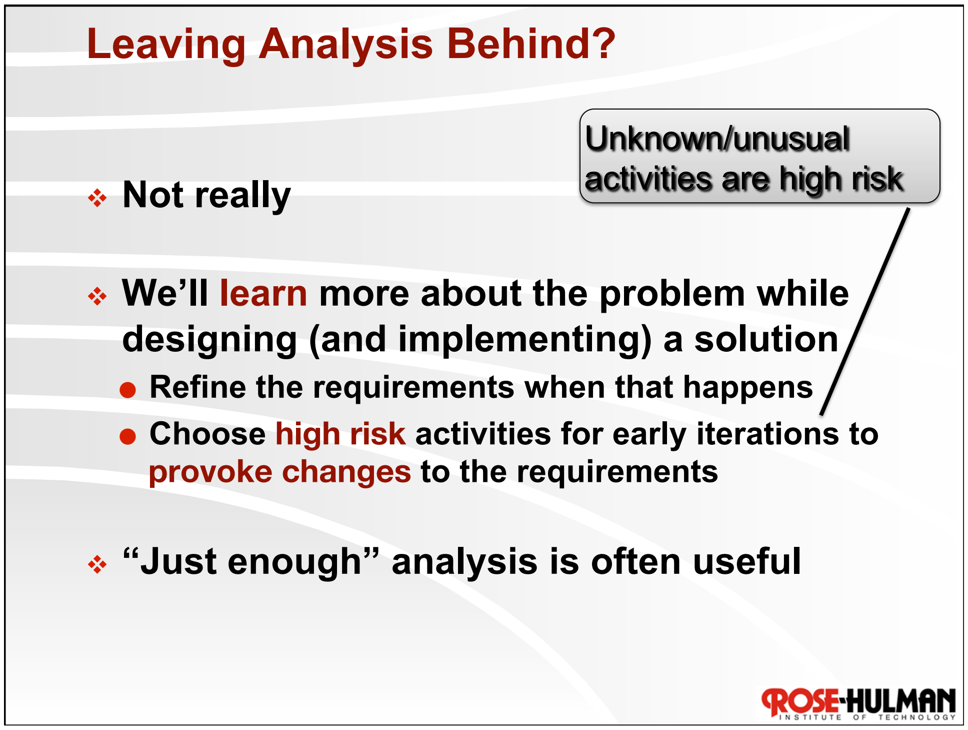
**Preconditions: dog owner, dog, veterinarian, and VSP all are registered with the system**

**Postconditions:**


THIS LIGHT ALWAYS TAKES FOREVER.
I'D LIKE TO SMACK THE IDIOT WHO DESIGNED THIS INTERSECTION.
HI.
WHO THE HELL ARE YOU?
I DESIGNED THIS INTERSECTION.
YOU'RE RIGHT– I SHOULD HAVE JUST MADE THE LIGHT SHORTER! NEVER MIND THE HOURS OF SIMULATION AND TESTING I DID. NEVER MIND THAT THIS INTERSECTION INTERACTS WITH ITS NEIGHBORS IN A COMPLICATED WAY AND IT TOOK ME A WEEK TO WORK OUT TIMING SEQUENCES THAT AVOIDED TOTAL JAMS.
CLEARLY, I'M A CRAPPY ENGINEER AND YOU HAVE A BETTER SOLUTION.
GO ON, SHOW ME YOUR PROPOSED TIMINGS.
GET THE HELL OFF MY HOOD BEFORE I START DRIVING AND FLING YOU INTO TRAFFIC.
YOU CAN'T. LIGHT'S RED.
WELL, WHEN WILL IT CHANGE?
TUESDAY.

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

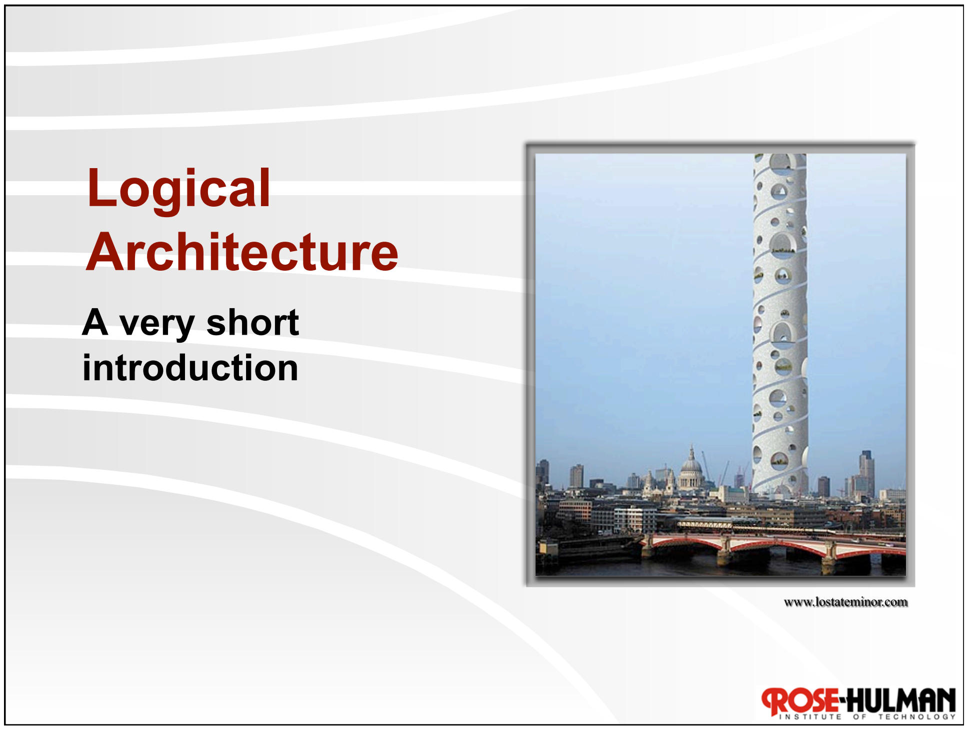


## Leaving Analysis Behind?

- 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

Unknown/unusual activities are high risk

ROSE-HULMAN INSTITUTE OF TECHNOLOGY



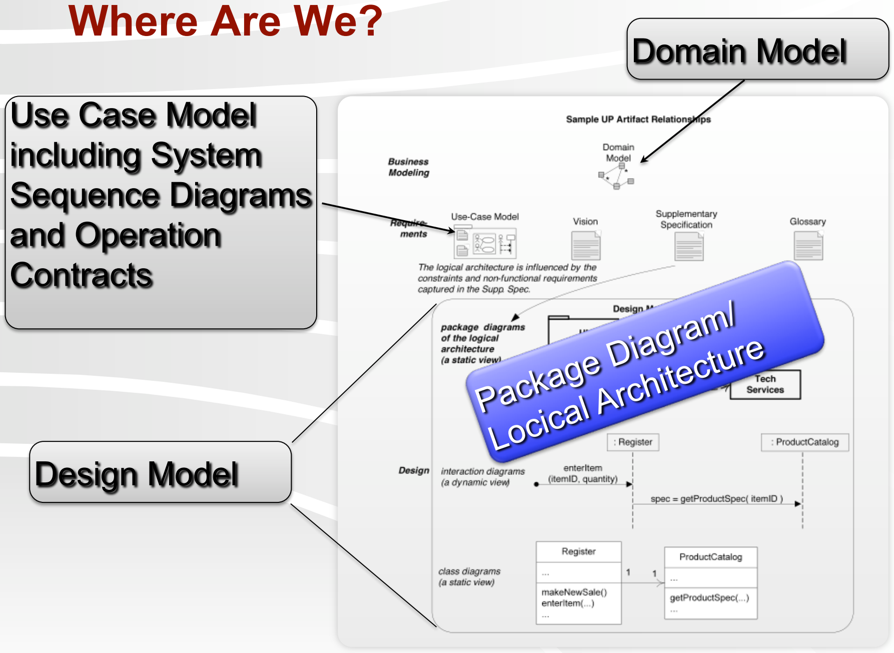
## Logical Architecture

A very short introduction

www.lostateminor.com

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

## Where Are We?


Domain Model
Use Case Model including System Sequence Diagrams and Operation Contracts
Design Model
Package Diagram/ Locical Architecture
Sample UP Artifact Relationships
Business Modeling
Domain Model
Require-ments
Use-Case Model
Vision
Supplementary Specification
Glossary
The logical architecture is influenced by the constraints and non-functional requirements captured in the Supp. Spec.
Design
package diagrams of the logical architecture (a static view)
Tech Services
interaction diagrams (a dynamic view)
: Register
: ProductCatalog
enterItem (itemID, quantity)
spec = getProductSpec( itemID )
class diagrams (a static view)
Register
makeNewSale()
enterItem(...)
ProductCatalog
getProductSpec(...)
1
1

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

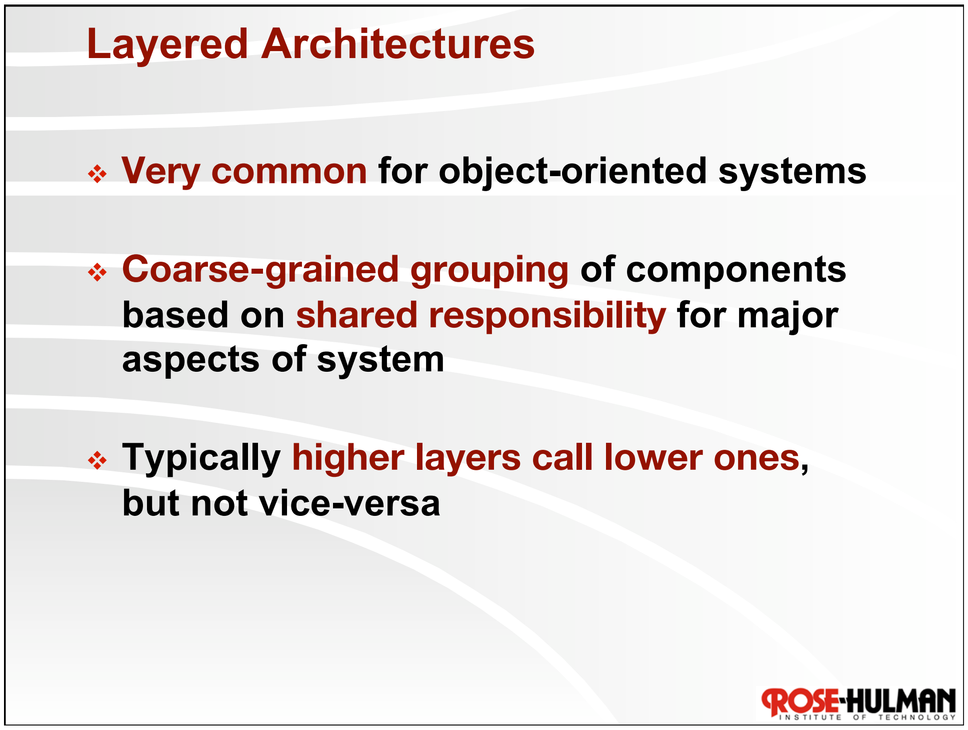
## Logical Architecture

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

***Why is an architecture necessary?***

Q8

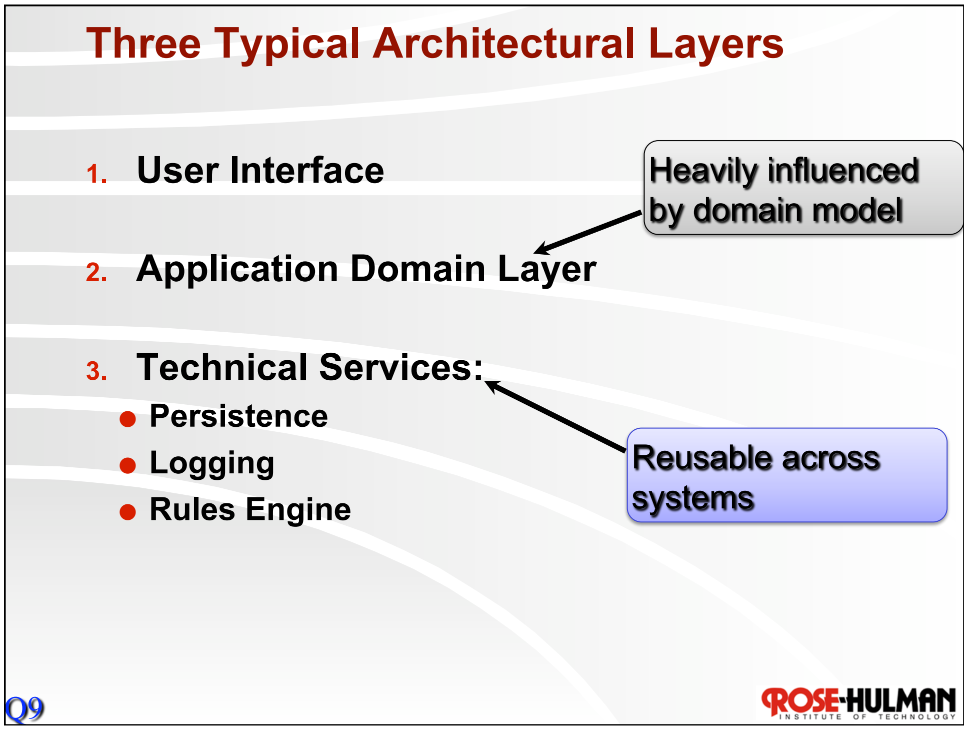
ROSE-HULMAN INSTITUTE OF TECHNOLOGY



## 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

ROSE-HULMAN INSTITUTE OF TECHNOLOGY



## Three Typical Architectural Layers

1. User Interface
2. Application Domain Layer
3. Technical Services:
   - Persistence
   - Logging
   - Rules Engine

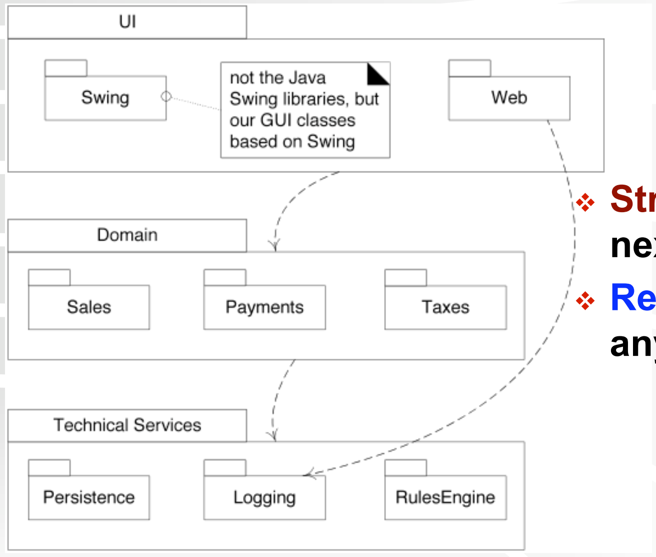
Heavily influenced by domain model

Reusable across systems

Q9

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

## Strict vs. Relaxed Layered Architectures



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

Q10

## Homework and Milestone Reminders

- **Read Chapters 12, 13, and 14 on Early Design**
- **Milestone 2 – Junior Project Domain Model**
  - **Due by 11:59pm on Friday, December 11th, 2009**
- **Homework 3 – Dog-eDoctor SSDs and Operations Contracts**
  - **Due by 5:00pm on Tuesday, December 15th, 2009**
- **Milestone 3 – Junior Project SSDs, OCs, and Logical Architecture – Coming!**
  - **Due by 11:59pm on Friday, January 8th, 2009**