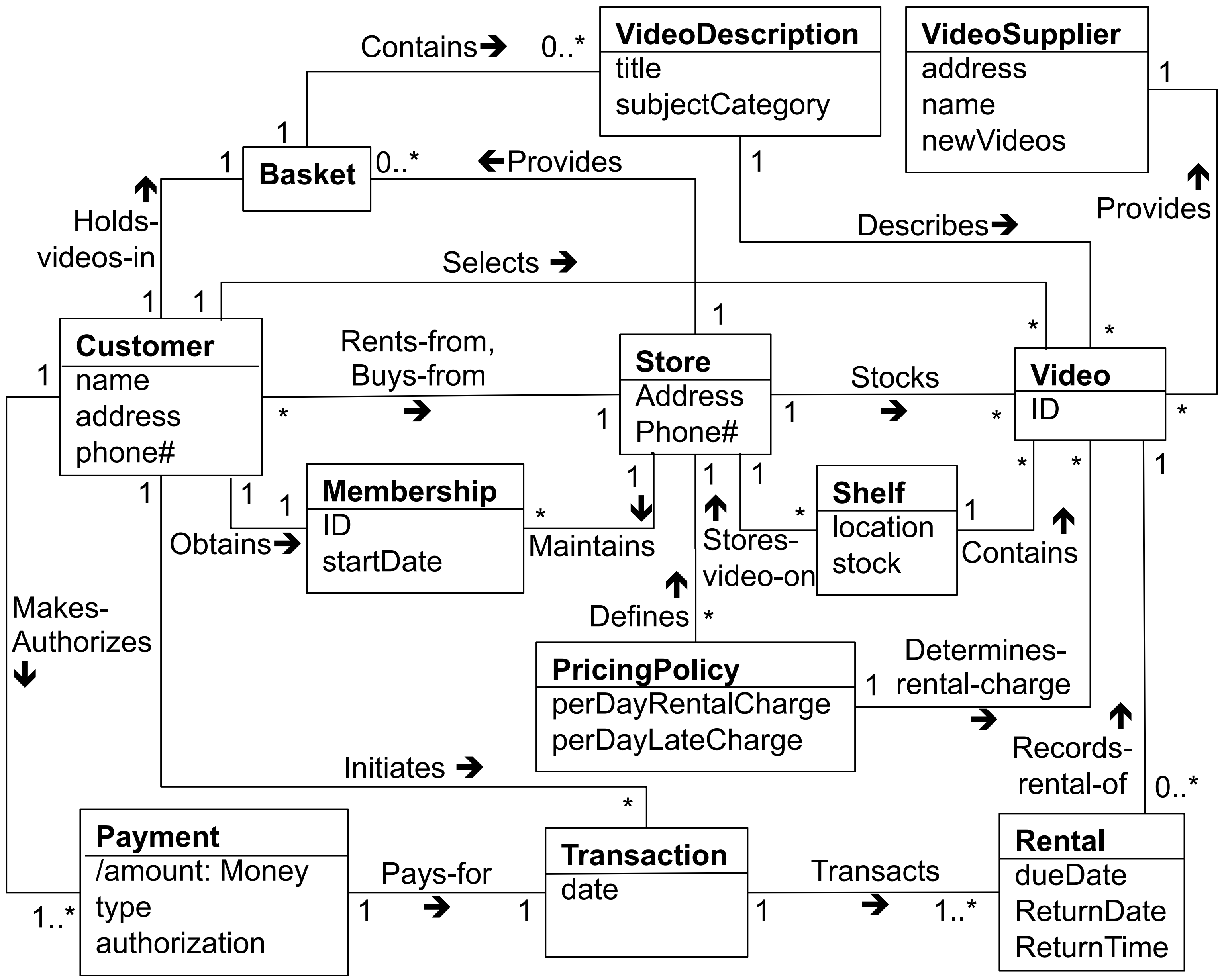
CSSE 374: Operations Contracts

Shawn Bohner

Office: Moench Room F212

Phone: (812) 877-8685

Email: bohner@rose-hulman.edu


VideoDescription
title
subjectCategory
VideoSupplier
address
name
newVideos
Contains →
0..*
1
Basket
1
0..*
← Provides
1
Holds-videos-in ↑
Describes →
Provides ↑
1
Selects →
1
1
1
Customer
name
address
phone#
Rents-from, Buys-from →
*
1
Store
Address
Phone#
1
Stocks →
1
*
*
*
Video
ID
*
1
1
1
1
Membership
ID
startDate
Obtains →
*
Maintains ↓
1
1
Stores-video-on ↑
1
*
Shelf
location
stock
1
Contains ↑
*
*
1
Makes-Authorizes ↓
Defines ↑
*
PricingPolicy
perDayRentalCharge
perDayLateCharge
Determines-rental-charge →
1
Records-rental-of ↑
0..*
Initiates →
*
Payment
/amount: Money
type
authorization
1..*
Pays-for →
1
1
Transaction
date
Transacts →
1
1..*
Rental
dueDate
ReturnDate
ReturnTime

Learning Outcomes: O-O Design

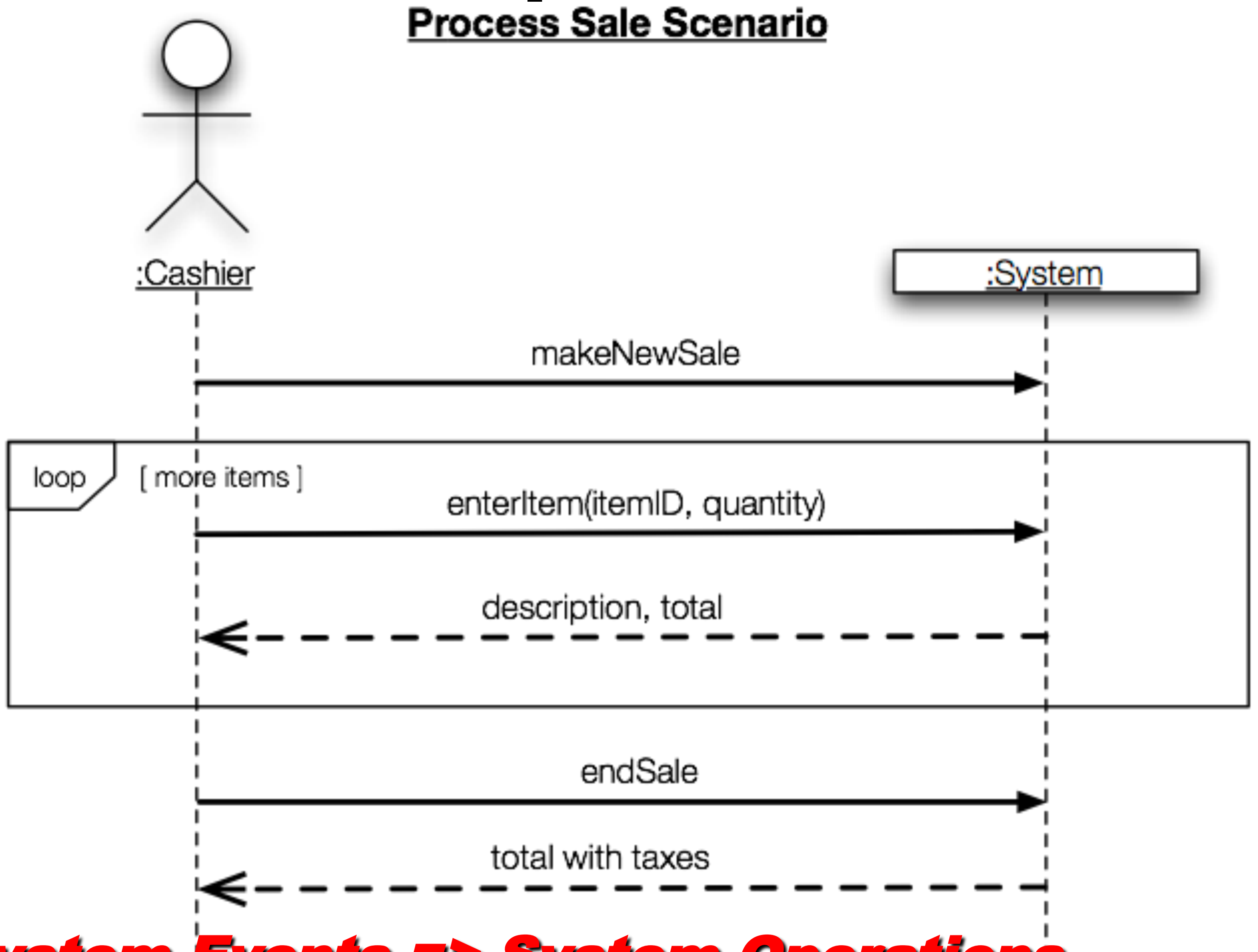
Demonstrate object-oriented design basics like domain models, class diagrams, and interaction (sequence and communication) diagrams.

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

- Introduce Operations Contracts (OCs)
- Do an Operations Contracts Exercise
- Transitioning from Requirements to Design
- Introduce Logical Architecture

ROSE-HULMAN INSTITUTE OF TECHNOLOGY

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.

Example OC:

(At most) one OC per System Operation

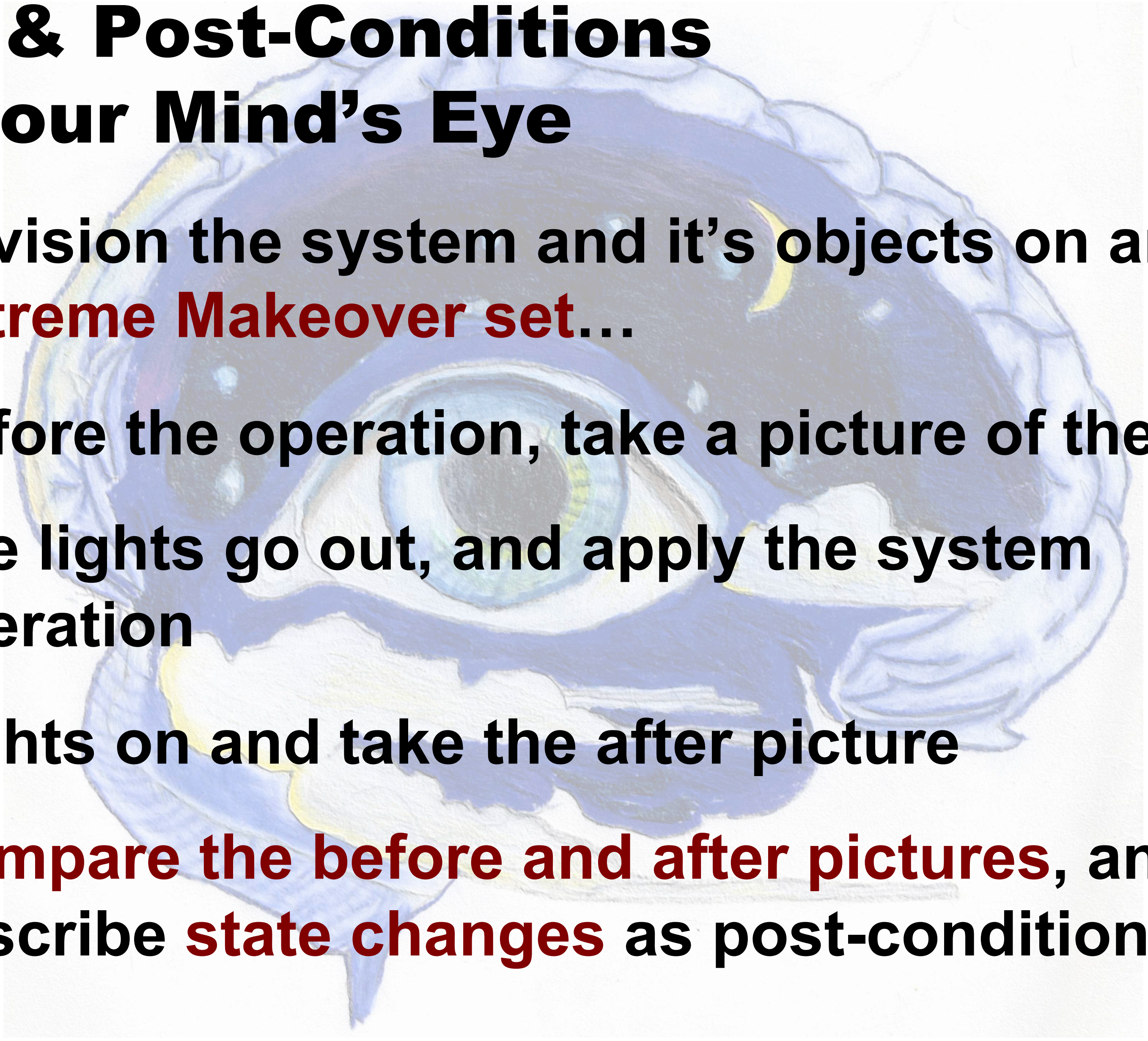
Contract CO2: enterItem

Any uses cases where this operation appears

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 ❖ sli was associated with the current Sale ❖ sli.quantity became quantity (attribute modification) ❖ sli was associated with ProductDescription based on itemID match

Noteworthy assumptions

Most important section



Pre & Post-Conditions in Your Mind's 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.

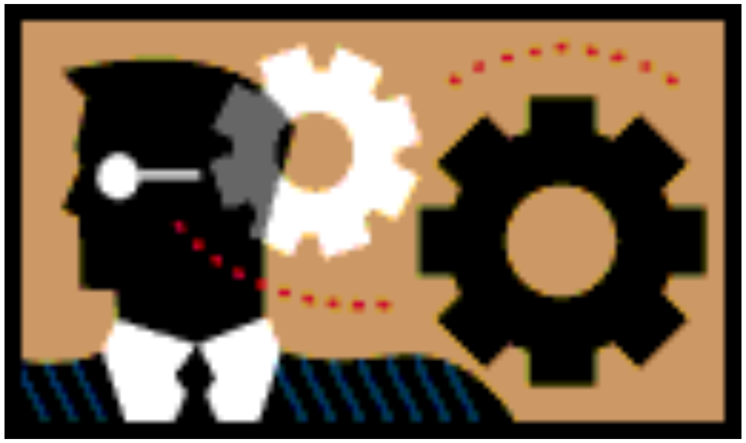
Postconditions (continued)

- 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

Why OC Post-Conditions?

- Domain model
  =>objects attributes and associations
- 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

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

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

Creating Operation Contracts

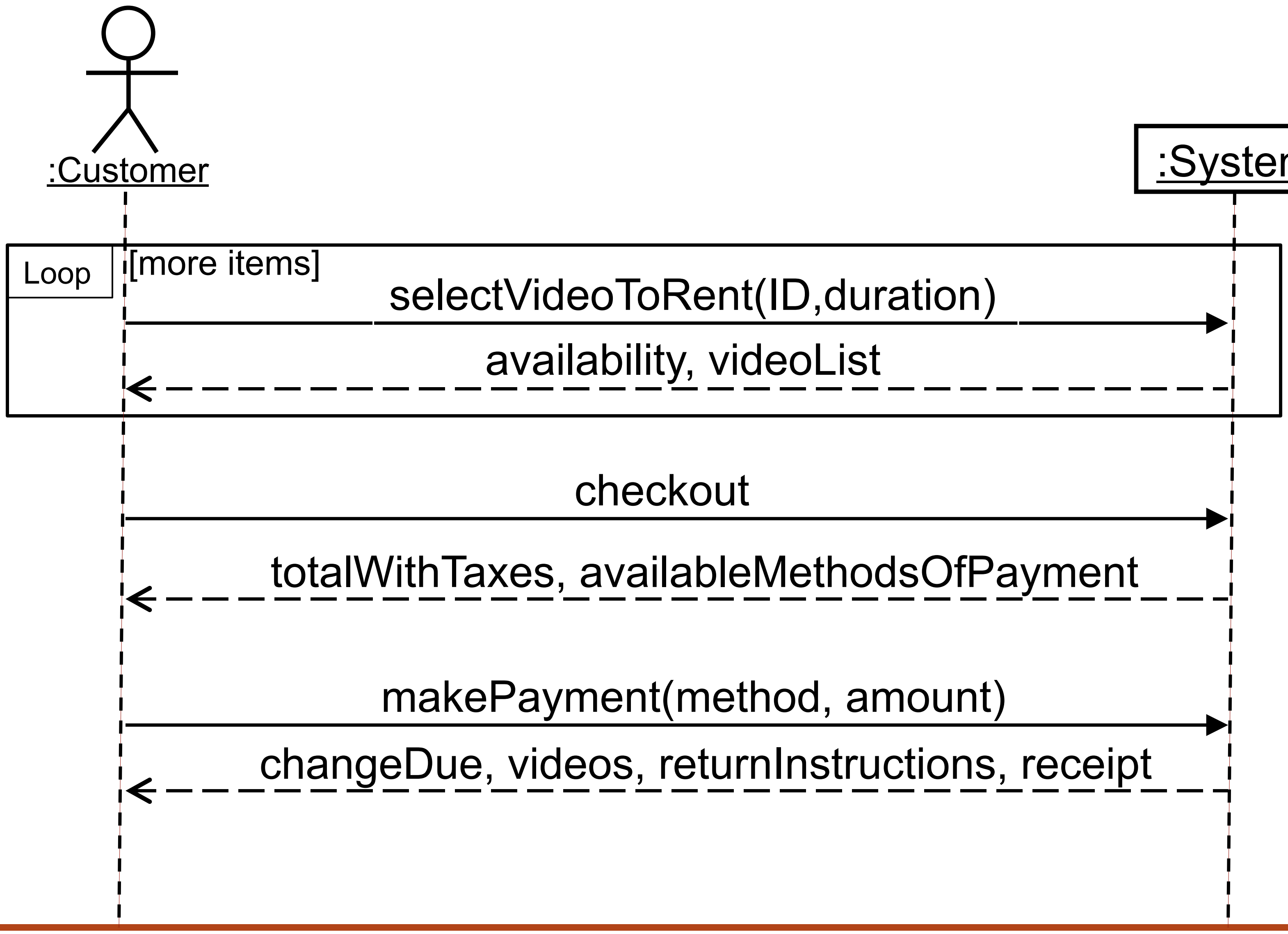
- Identify System Operations from SSDs
- Make contracts for System Operations that are:
  - Complex and perhaps subtle in their own results
  - Not clear in the use case
- Again, in describing post-conditions use:
  - Instance creation and deletion
  - Attribute modification
  - Associations formed and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

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

Class Exercise on Domain Modeling

- Break up into your project teams
- Look over the SSD from Tuesday looking for system operations and Read the Use Case again referring to the Domain Model
- Write an Operations Contract for MakePayment (method, amount)

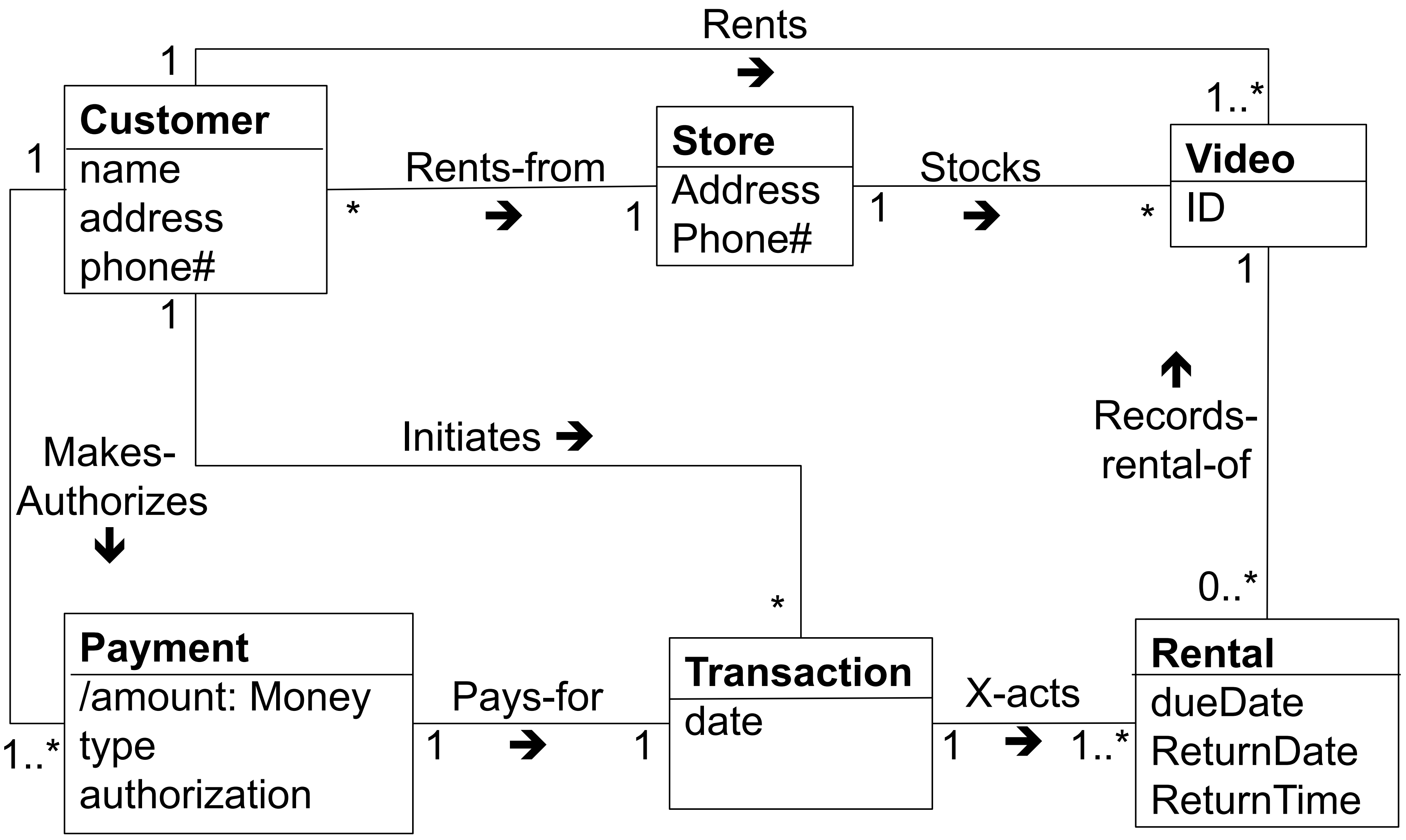
SSD for Use Case 1


:Customer
:System
Loop
[more items]
selectVideoToRent(ID,duration)
availability, videoList
checkout
totalWithTaxes, availableMethodsOfPayment
makePayment(method, amount)
changeDue, videos, returnInstructions, receipt

Homework 1: Basic Use Case 1/2

- **UC1: Customer rents videos**
  - **Preconditions**: Customer has a membership, has selected videos they want, and made system aware of their choices.
- **Main flow:**
  1. Actor indicates to rent first item (e.g., clicking "rent" on a networked device, or scanning it physically in a store)
  2. System verifies immediate availability, and waits to make next option
  3. Actor indicates they are done selecting
  4. System shows total, prompts for payment
  5. Actor selects method of payment, entering additional data if needed (e.g., credit card number)
  6. System verifies the payment has gone through, schedules the goods for rental (e.g., sets up a window to click on to view the video remotely, or tells the store clerk where to find the DVD)…
- **Postcondition:** Rental transaction is complete

Concise DM For Video Store



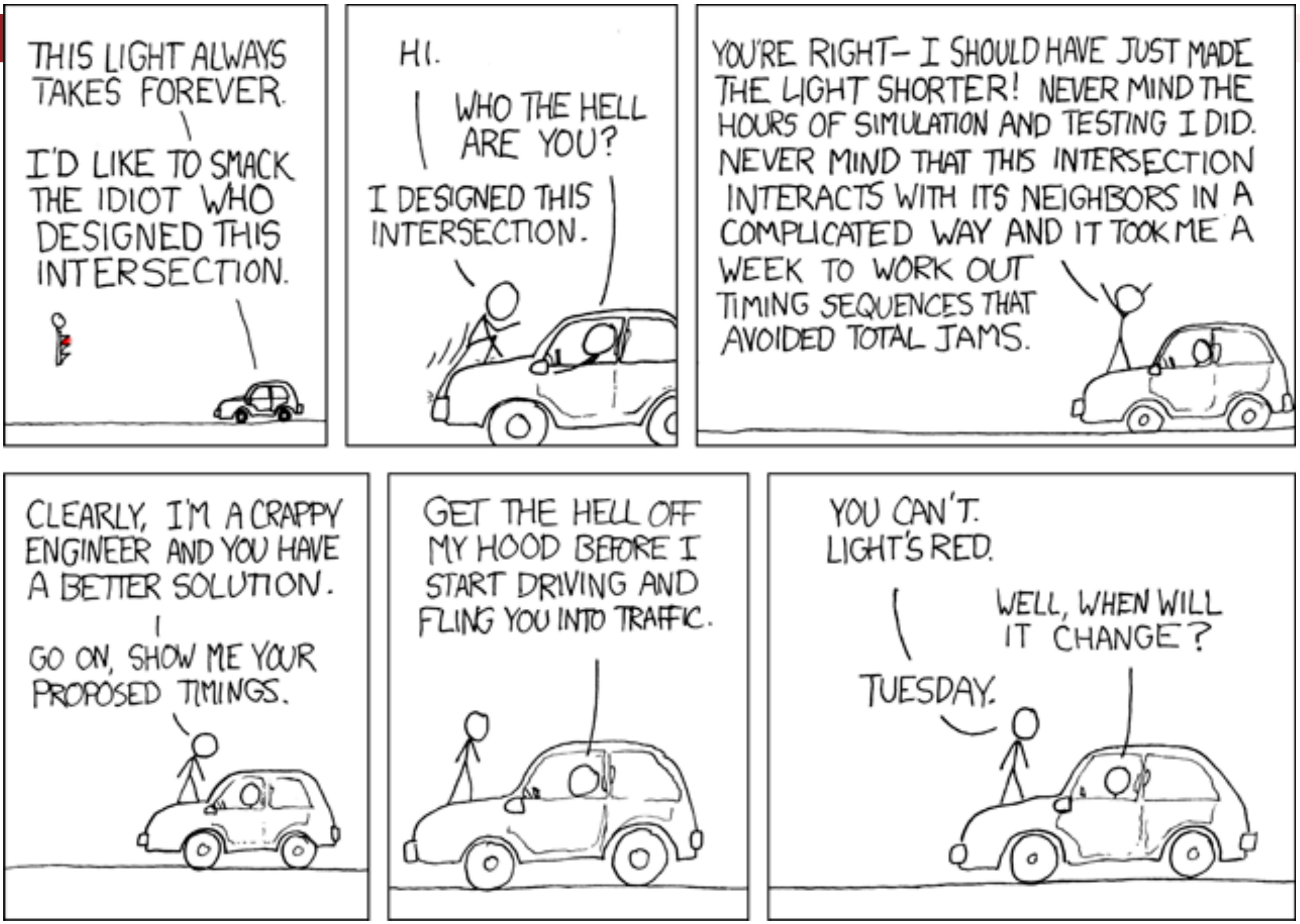
Exercise: Complete the OC

Operation: makePayment(method, amount)

Cross references:

Preconditions:

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.

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

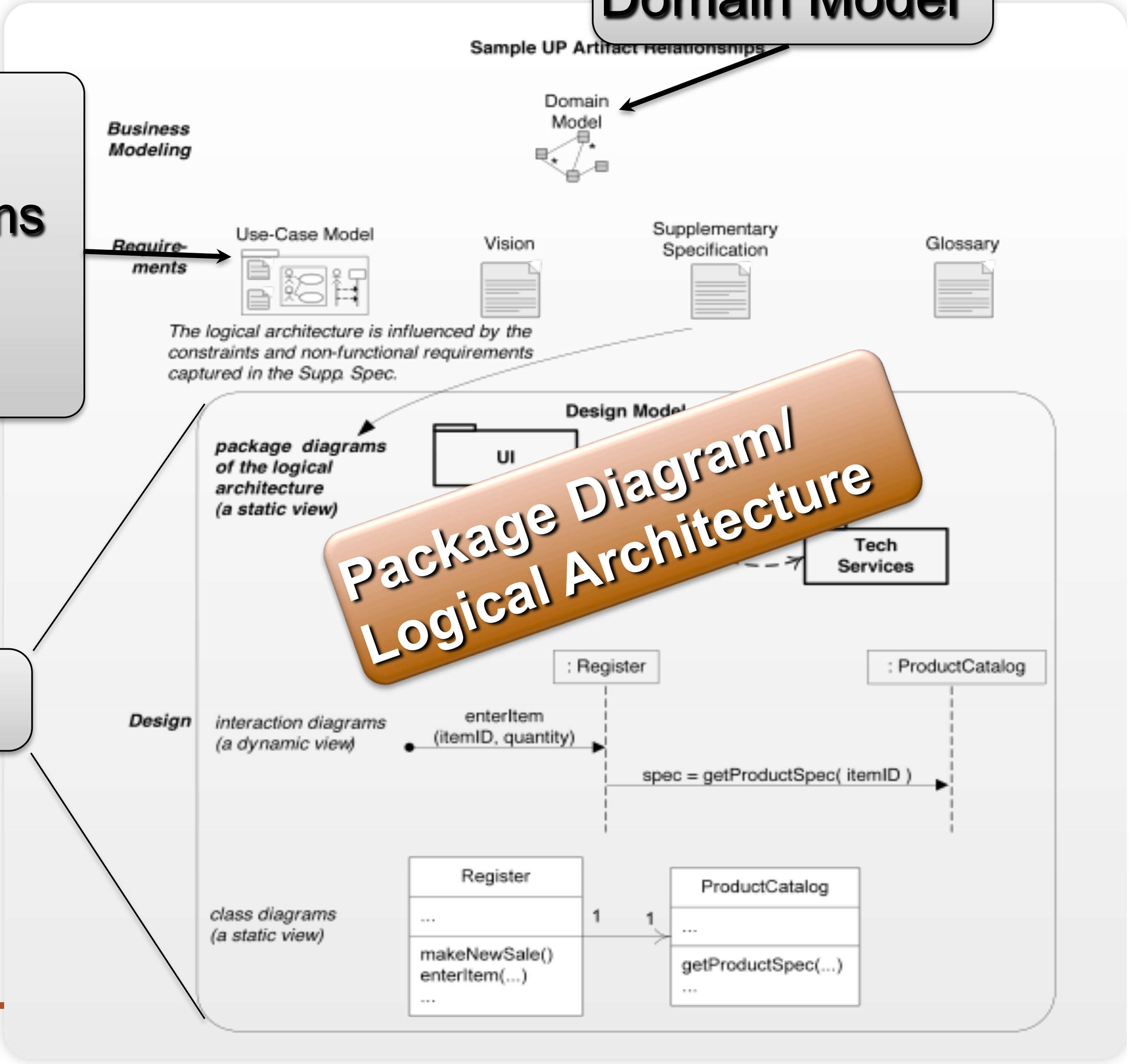
Where Are We?

Domain Model

Use Case Model including System Sequence Diagrams and Operation Contracts

Design Model

Package Diagram/ Logical Architecture



Sample UP Artifact Relationships
Business Modeling
Domain Model
Requirements
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 Model
package diagrams of the logical architecture (a static view)
UI
Tech Services
Design
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

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

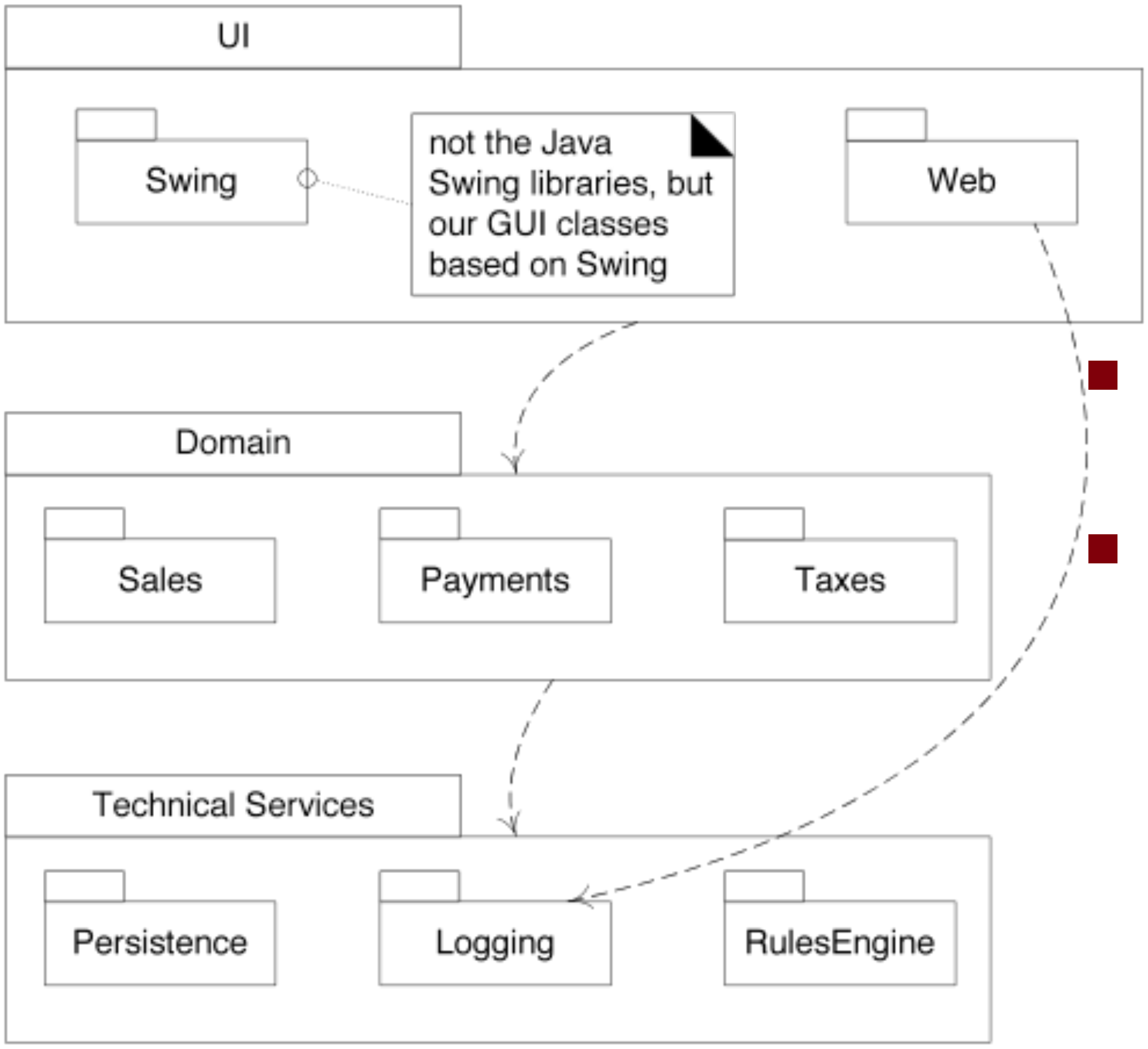
Three Typical Architectural Layers

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

Heavily influenced by domain model

Reusable across systems

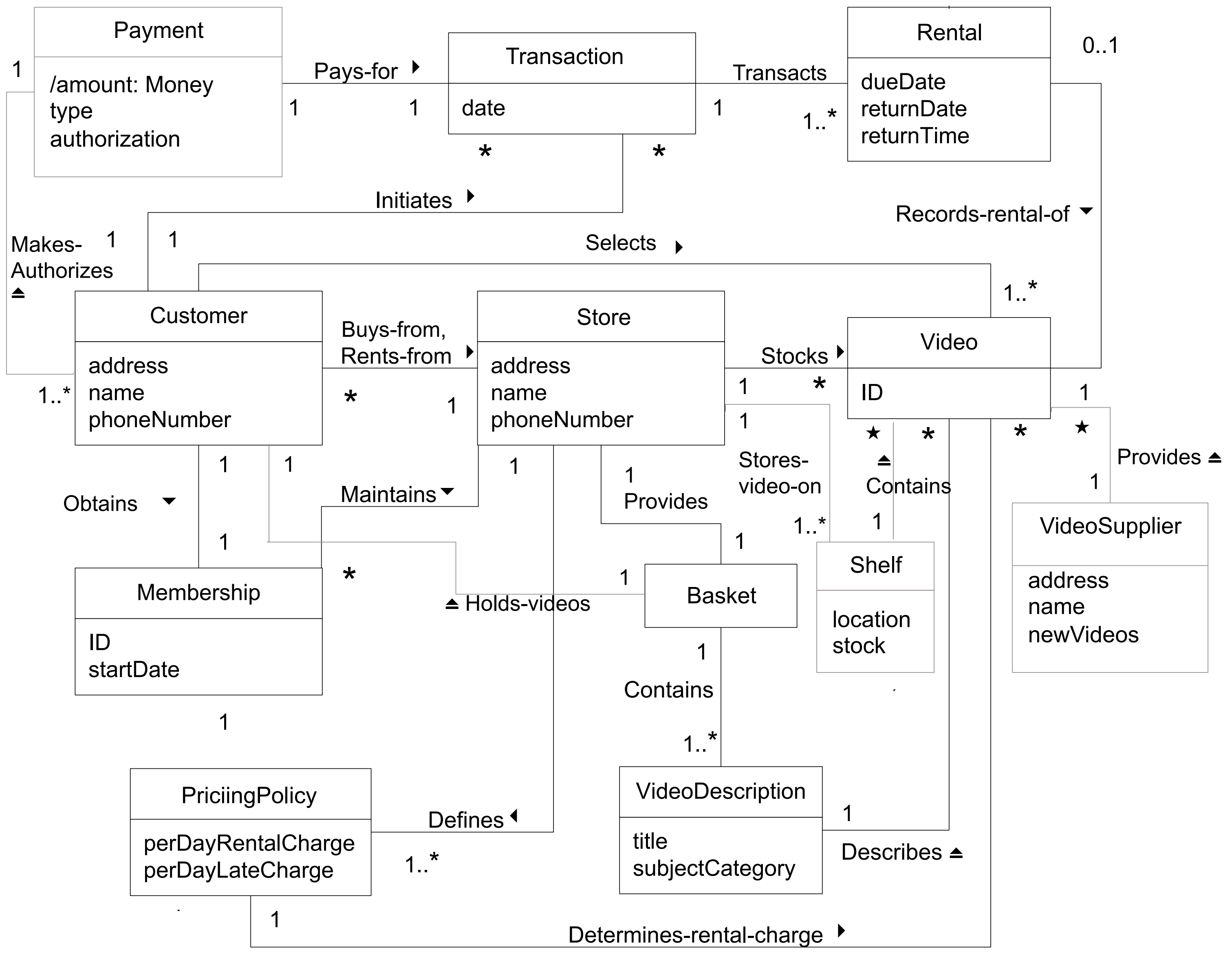
Strict vs. Relaxed Layered Architectures



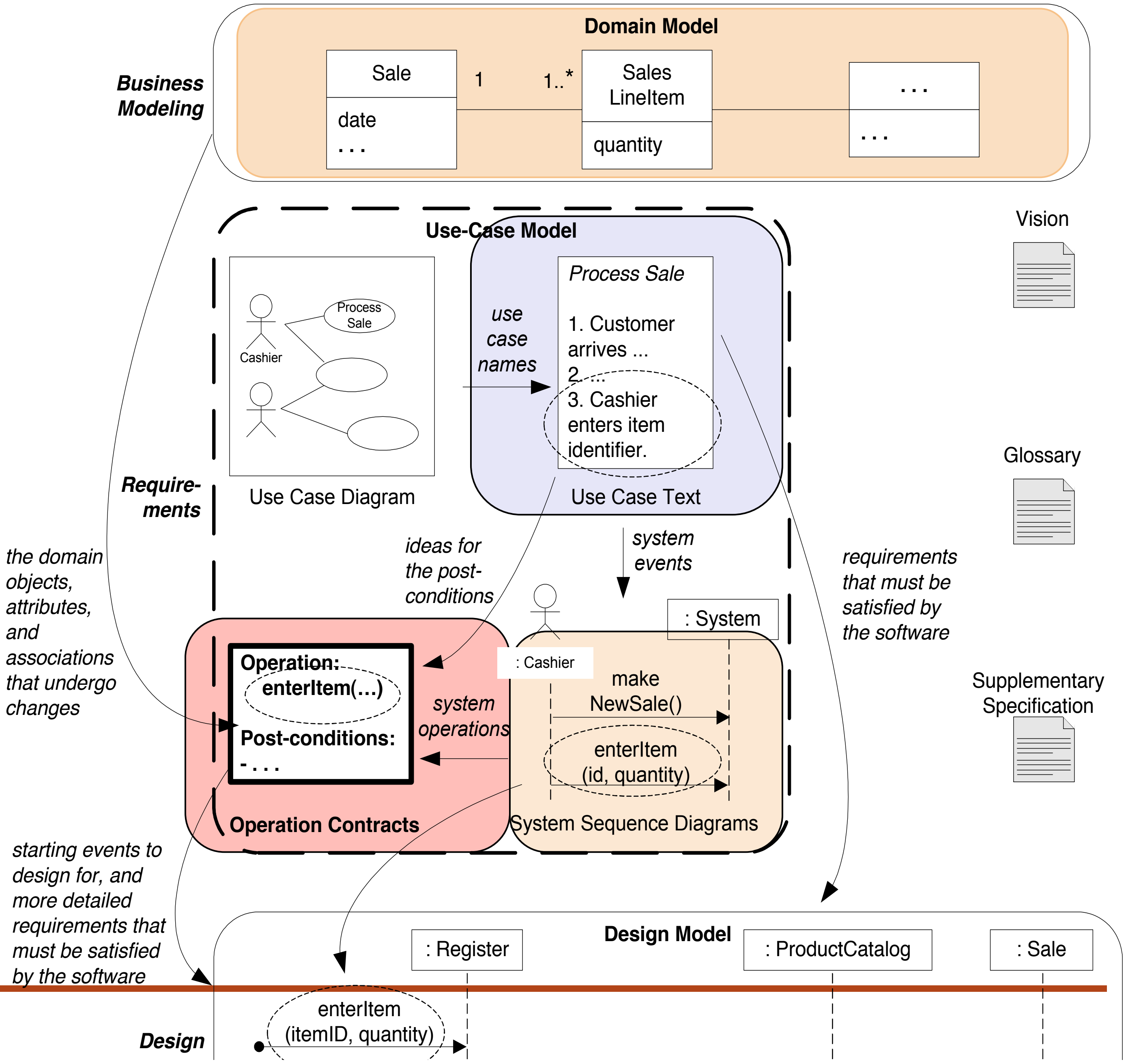
- **Strict**: only calls next layer down
- **Relaxed**: can call any layer below

Homework and Milestone Reminders

- Read Chapters 12, 13, and 14 on Early Design
- Milestone 2 – Junior Project Domain Model
  - Due by 11:55pm on Friday, December 10th, 2010
- 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 – Coming!
  - Due by 11:59pm on Friday, January 7th, 2010


Payment
/amount: Money
type
authorization
Transaction
date
Rental
dueDate
returnDate
returnTime
Pays-for
Transacts
0..1
Initiates
Records-rental-of
Makes-Authorizes
Selects
Customer
address
name
phoneNumber
Buys-from, Rents-from
Store
address
name
phoneNumber
Stocks
Video
ID
Obtains
Maintains
Provides
Stores-video-on
Contains
Provides
VideoSupplier
address
name
newVideos
Shelf
location
stock
Membership
ID
startDate
Holds-videos
Basket
Contains
PriciingPolicy
perDayRentalCharge
perDayLateCharge
Defines
VideoDescription
title
subjectCategory
Describes
Determines-rental-charge
1
1..*
*

System Operation Contracts


Domain Model
Sale
date
. . .
1
1..*
Sales LineItem
quantity
. . .
. . .
Business Modeling
Use-Case Model
Process Sale
Cashier
Use Case Diagram
use case names
Process Sale
1. Customer arrives ...
2. ...
3. Cashier enters item identifier.
Use Case Text
Require-ments
Vision
Glossary
Supplementary Specification
the domain objects, attributes, and associations that undergo changes
ideas for the post-conditions
system events
requirements that must be satisfied by the software
: System
: Cashier
make NewSale()
enterItem (id, quantity)
System Sequence Diagrams
Operation: enterItem(…)
Post-conditions:
- . . .
system operations
Operation Contracts
starting events to design for, and more detailed requirements that must be satisfied by the software
Design Model
: Register
: ProductCatalog
: Sale
enterItem (itemID, quantity)
Design