

CSSE 374: Operational Behaviors – System Sequence Diagrams



Shawn Bohner

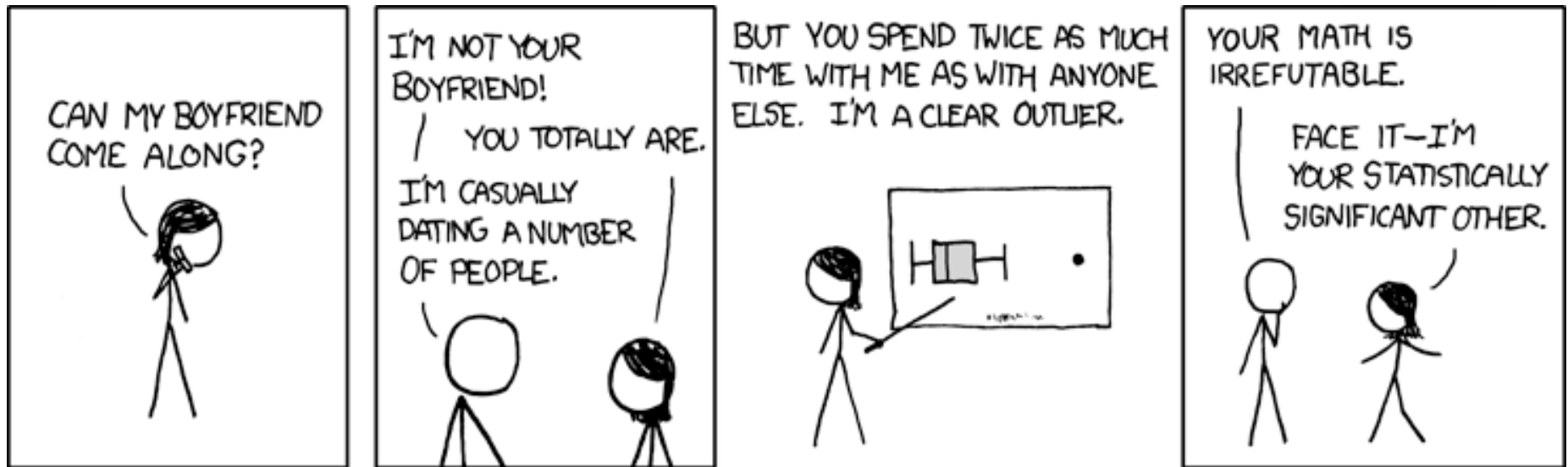
Office: Moench Room F212

Phone: (812) 877-8685

Email: bohner@rose-hulman.edu



Clear Misuse of Statistics



... So, is this a case of premature design decisions?

Learning Outcomes: O-O Design

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

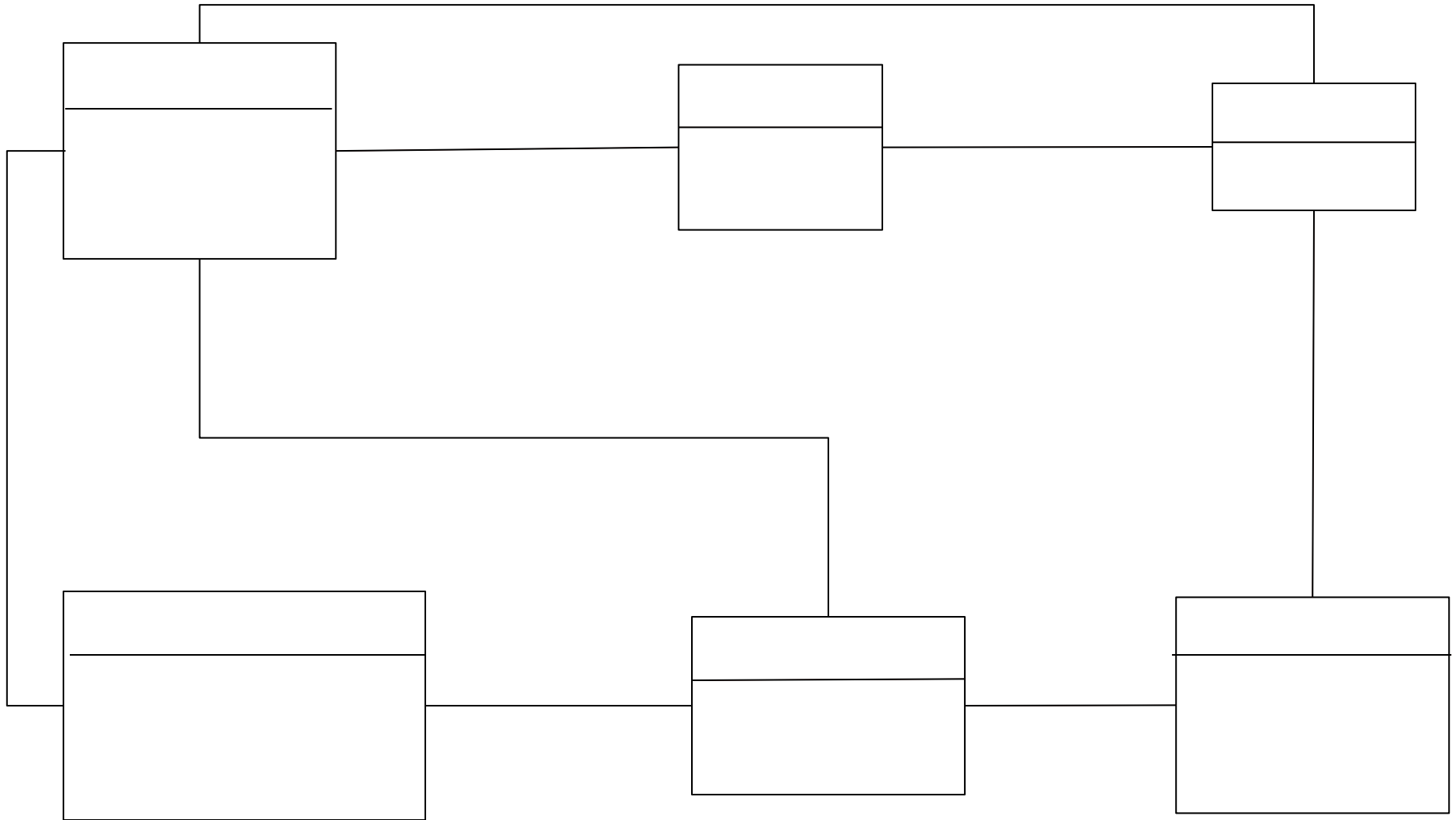
- **Tying loose ends on Domain Model Attributes**
- **Introduce Behavioral Modeling**
- **Examine System Sequence Diagrams (SSD)**
- **Do an SSD Exercise**



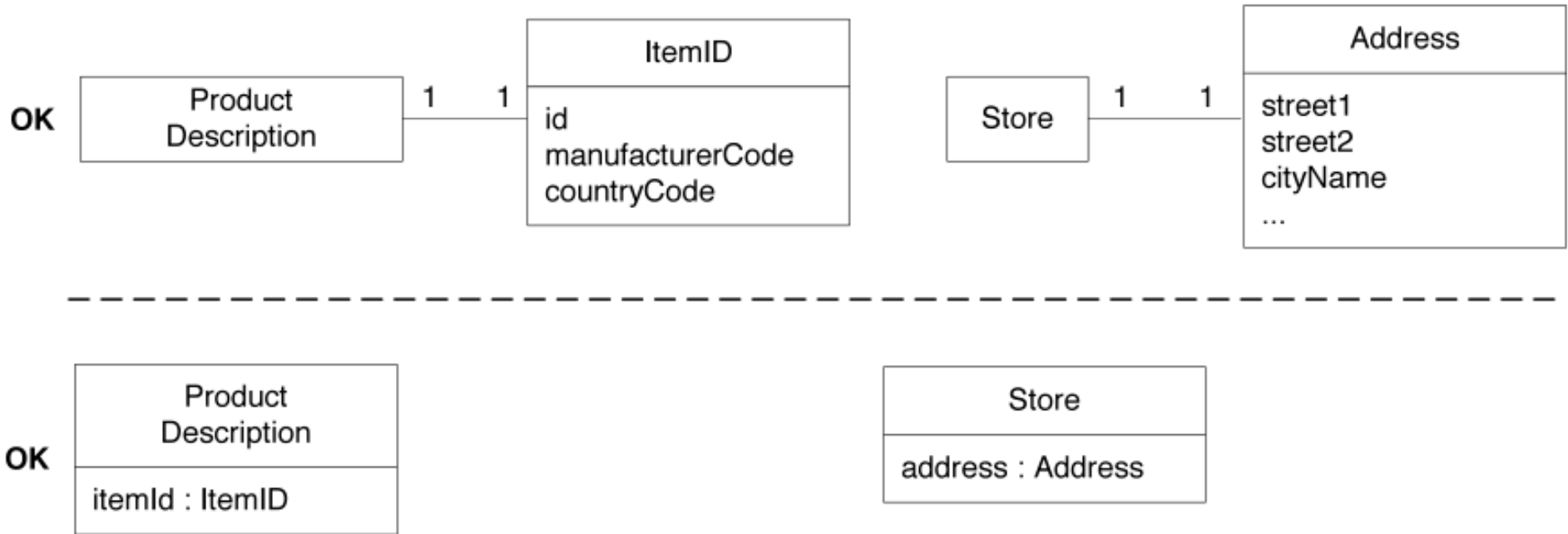
<http://enterprisegeeks.com/blog/2009/07/>



Recall: DM Exercise from Yesterday

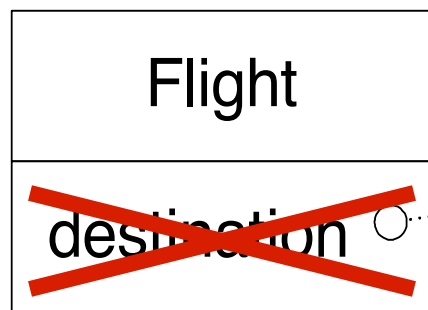


Showing Data Type Attributes



Choose the representation that best communicates with the stakeholders

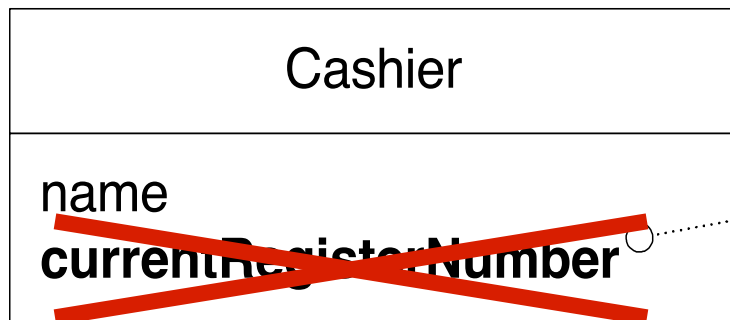
Show **Complex Concepts** with Associations (not attributes)



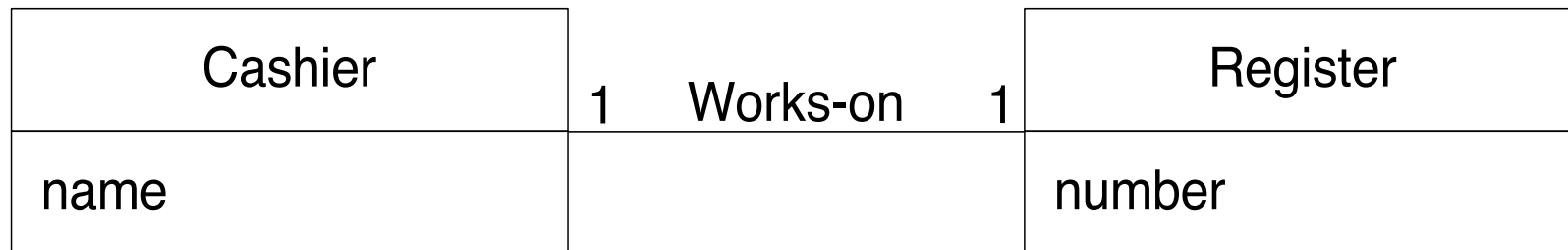
destination is a complex concept



Avoid Using Data Type Attributes as Foreign Keys



a "simple" attribute, but being used as a foreign key to relate to another object





Domain Model Guidelines: Summary

- Use terms from the application domain
- Classes first, then associations and attributes
- Use existing models, category lists, noun phrases
- Include *report* objects, (e.g., Receipt), if part of the business rules
- Don't send an attribute to do a conceptual class's job
- Use description classes to remember information independent of instances and to reduce redundancy
- Use association for relationship that must be remembered
- Be "parsimonious" with associations
- Name associations with verb phrases, not "has" or "uses"
- Use common association lists
- Use attributes for information that must be remembered
- Use data type attributes
- Define new data types for complex data
- Communicate with stakeholders

Domain Model Classes show static information. How can we show dynamic behaviors like event sequencing found in our Use Cases?

- Again, think for 15 seconds...
- Turn to a neighbor and discuss it for a minute

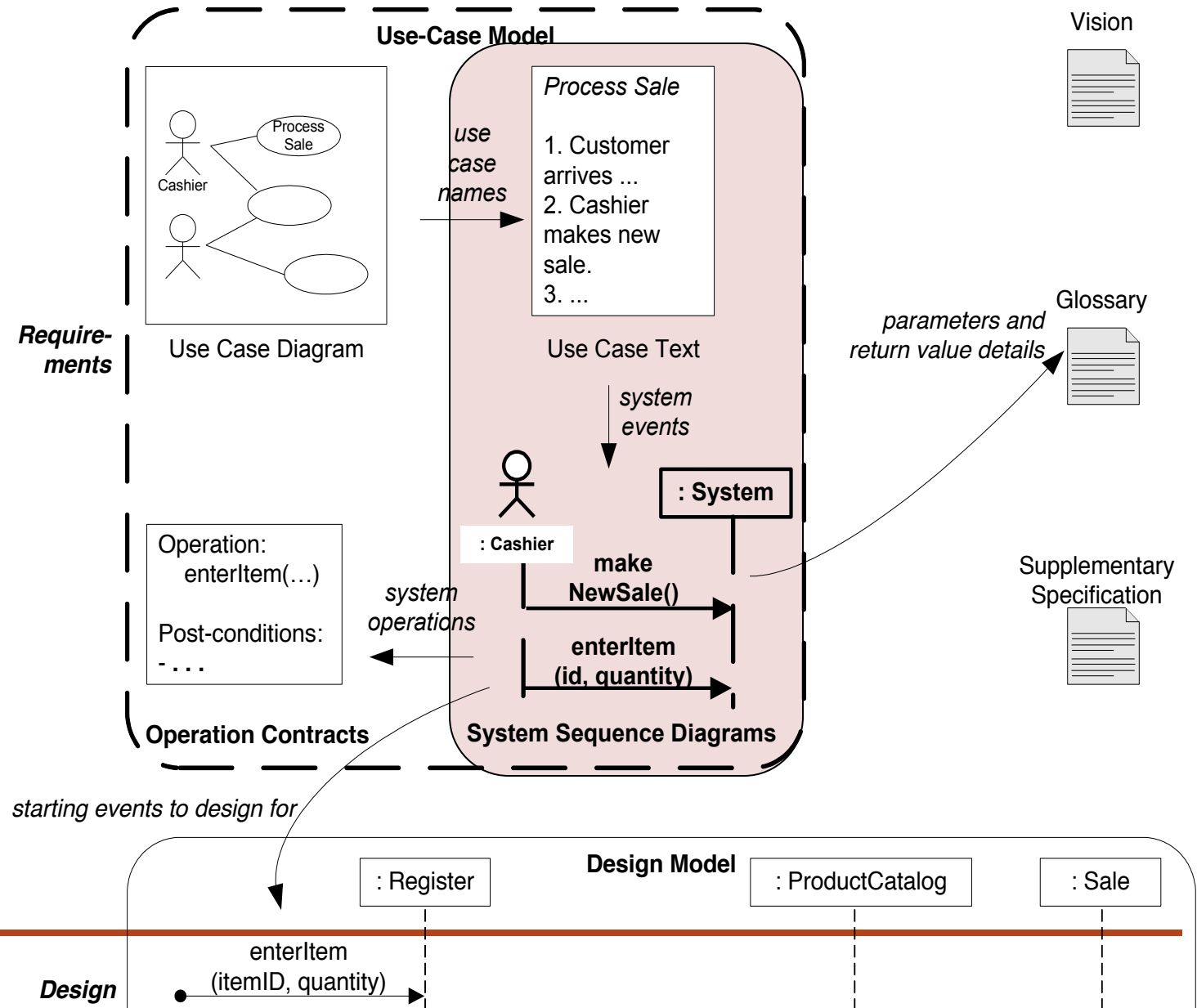


System Sequence Diagrams (SSD)

- A specialization of “sequence diagrams” that describe system behaviors
- SSDs show interactions between the system and external actors
- SSDs typically modeled for:
 - Main use case scenario
 - Frequent and alternative scenarios

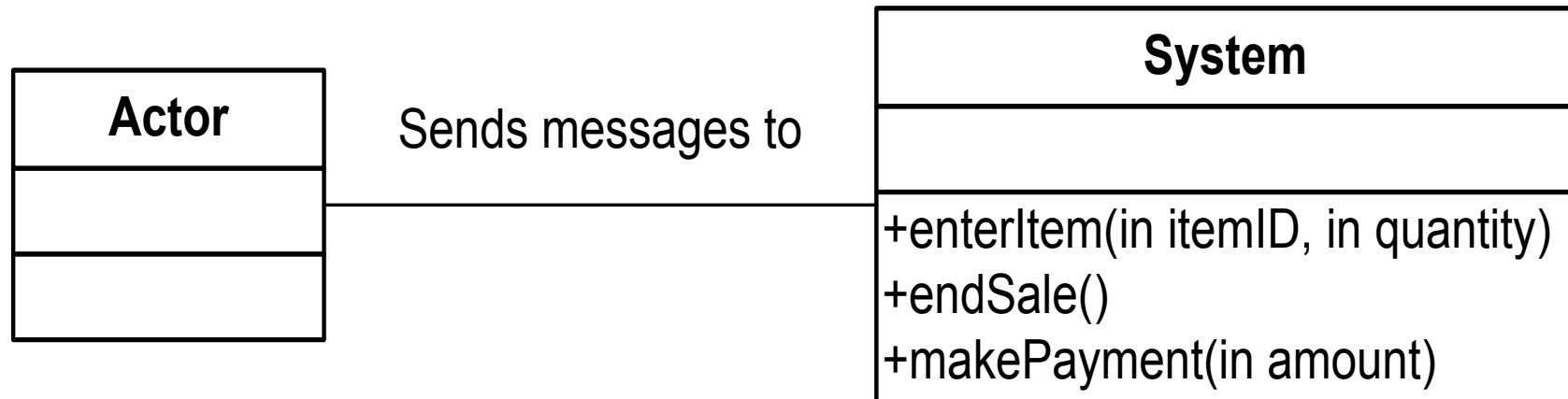


Where are SSDs in the Analysis Model?



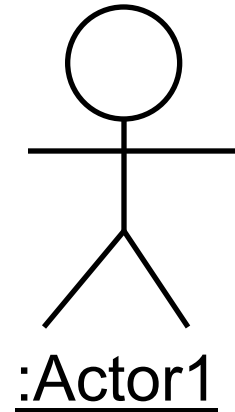
Modeling Behavior from a System Perspective

A *Use Case Scenario* is an ordered series of operations (functions) that Actors invoke on the System

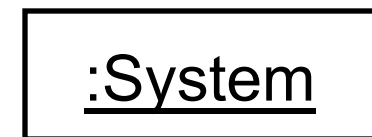


SSD Notation

- **Actor:** An Actor is modeled using the ubiquitous stick figure symbol



- **Object:** is represented as a rectangle which contains the name of the object underlined

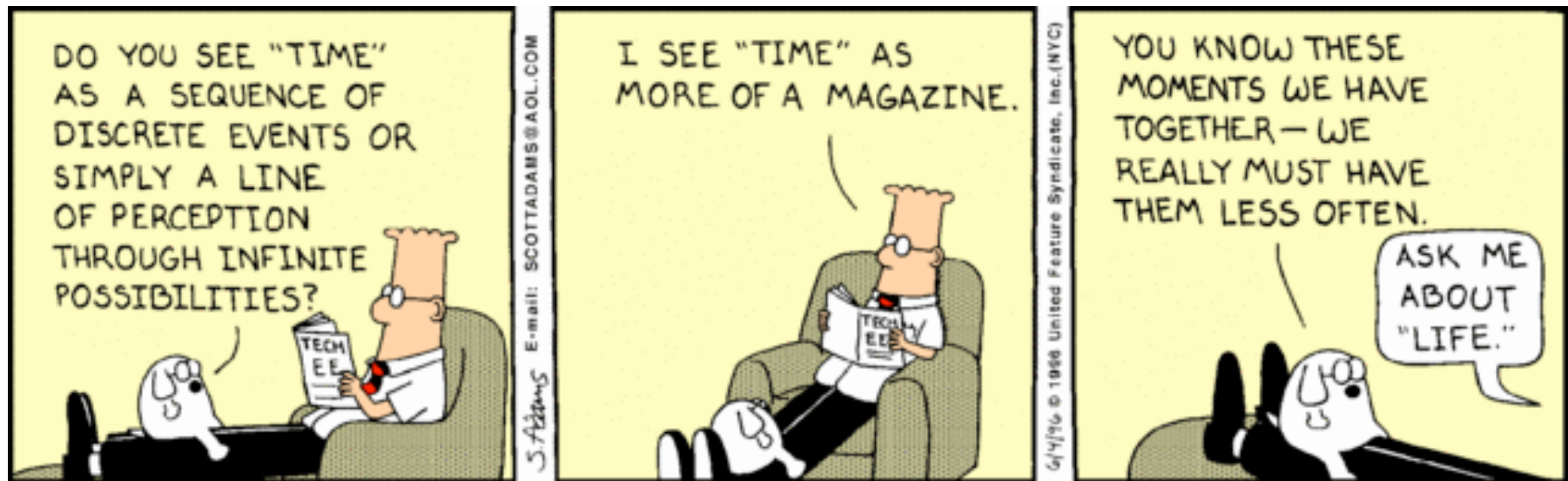


SSD Notation (continued)

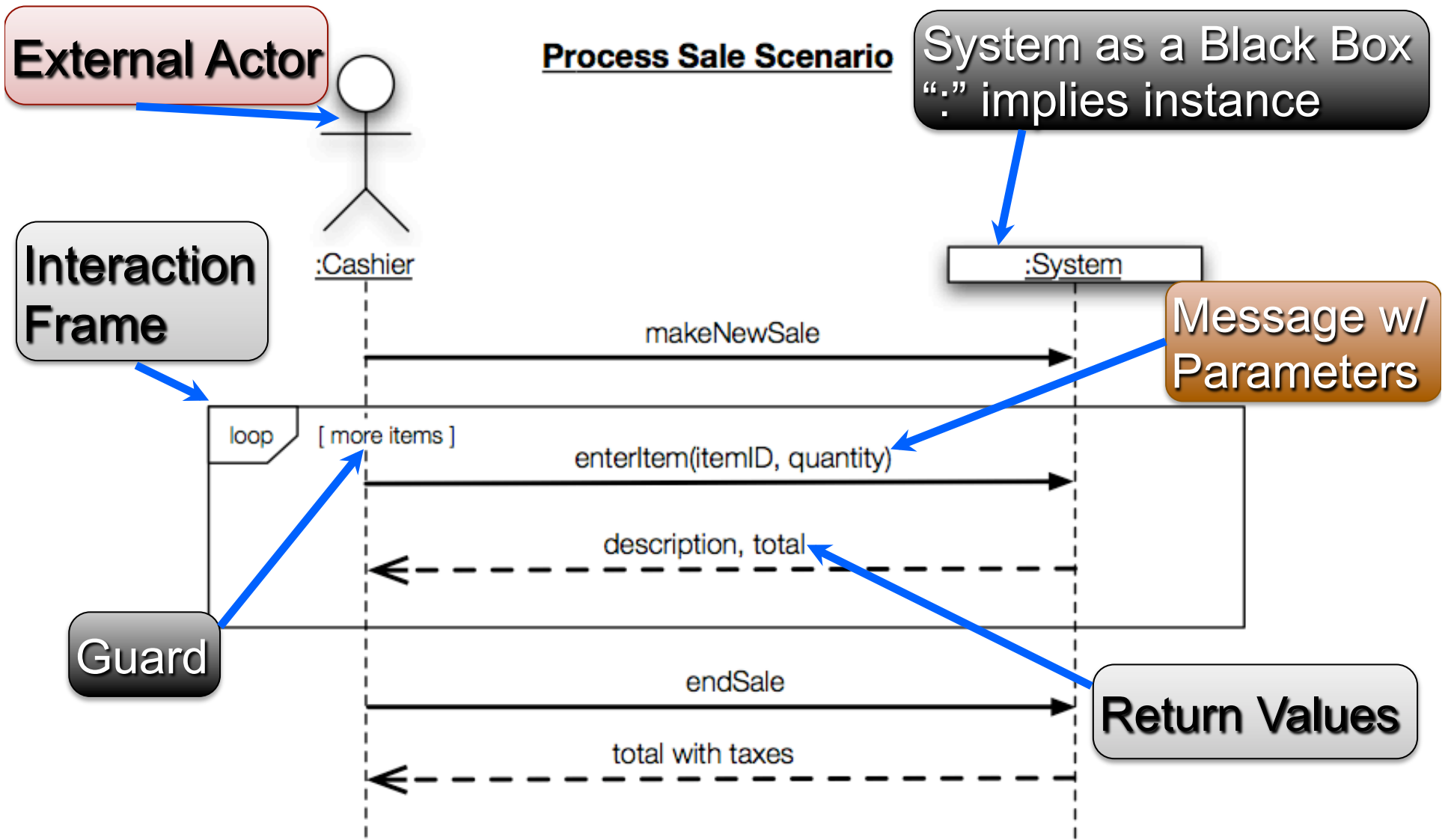
- **Lifeline:** is depicted as a vertical dotted line extending from an object that identifies the existence of the object over time
- **Message:** modeled as horizontal arrows between activations, indicate the communications between objects



Dogbert's take on Time



SSD Notation: An Example

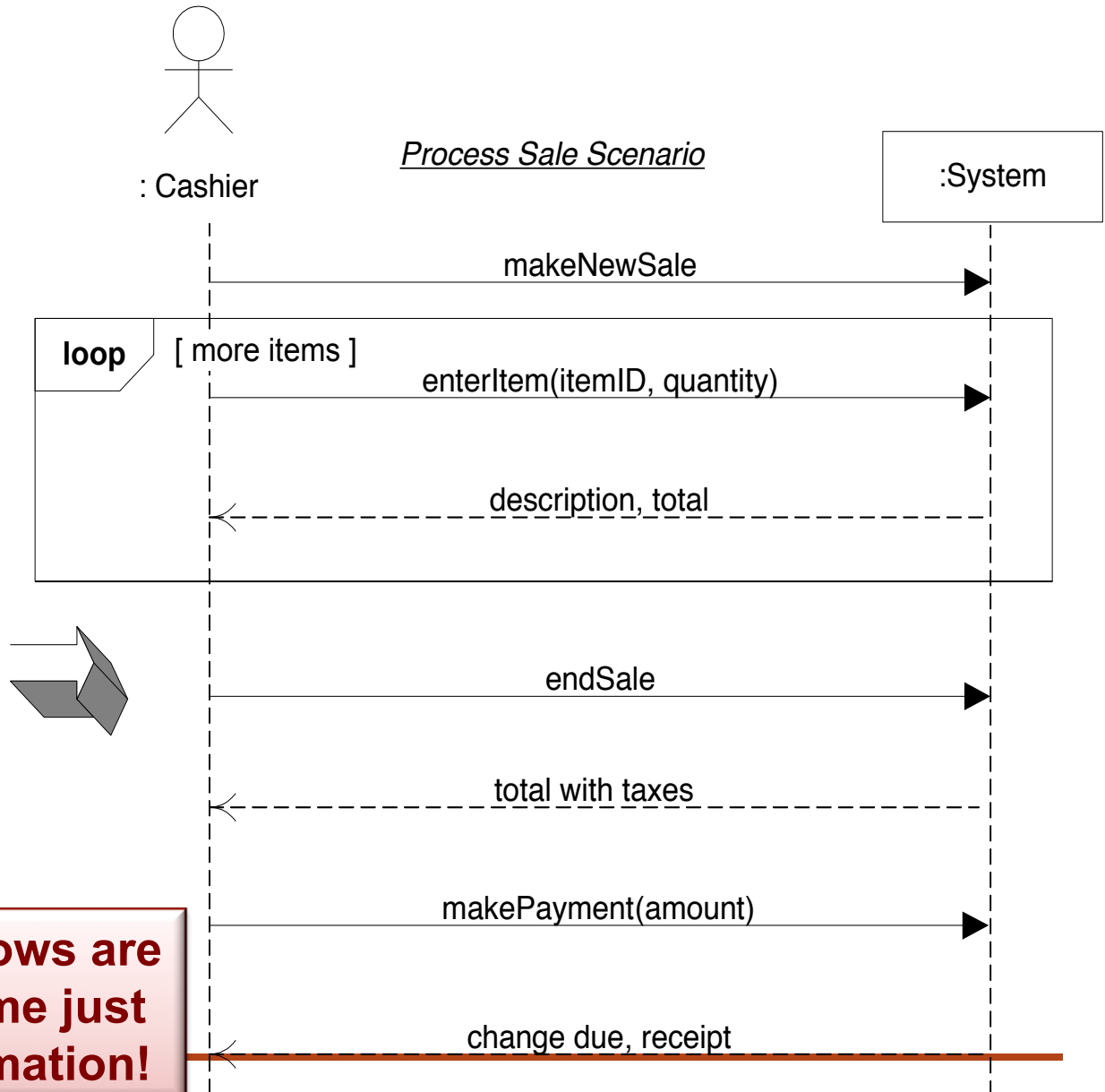


Relating UC and SSD

Simple cash-only *Process Sale* scenario:

1. Customer arrives at a POS checkout with goods and/or services to purchase.
 2. Cashier starts a new sale.
 3. Cashier enters item identifier.
 4. System records sale line item and presents item description, price, and running total.
- Cashier repeats steps 3-4 until indicates done.
5. System presents total with taxes calculated.
 6. Cashier tells Customer the total, and asks for payment.
 7. Customer pays and System handles payment.
- ...

Note not all arrows are functions... some just events or information!





From Use Case to SSD

- Use cases describe how *external actors* will interact with our system
- Actors generate *system events* requesting some *system operation*
- For a *single scenario* of a use case, SSD shows *system events and their order*
- All systems treated as black boxes; only show events that *cross system boundaries*



Also inter-system events



Why Draw an SSD?

- Software systems react to three things:
 1. External input events (a.k.a., **system events**) from actors
 2. Timer events
 3. Faults or exceptions

- SSD captures **System Behavior**: a description of what a system does, **NOT** how it does it



How To “Tips” on Creating SSDs

- Show **one scenario** of a use case
- Show events as **intentions**, not physical implementation
 - e.g., *enterItem* (not *scanItem*)
 - e.g., *presentCredentials*, (not *enterPassword*)
- Start system event names with **verbs**
- Can model collaborations between systems
- Give **details in the Glossary**

Key Idea: SSDs are a Bridge

- **Challenge:** To transition the functional UCs into OO System Model
 - Without losing any requirements
 - Delivering a correct, robust system

- **System Sequence Diagram is the key**
 - Links UCs with OO models (e.g., class & sequence)
 - Supported by operation contracts
 - Provides traceability of requirements into OO models



Class Exercise on Domain Modeling

- Break up into your project teams
- Read the Use Case again and determine the key events
- Draw a SSD for Use Case 1 (without alternatives)



Homework 1: Basic Use Case 1/2

■ UC1: Customer rents videos

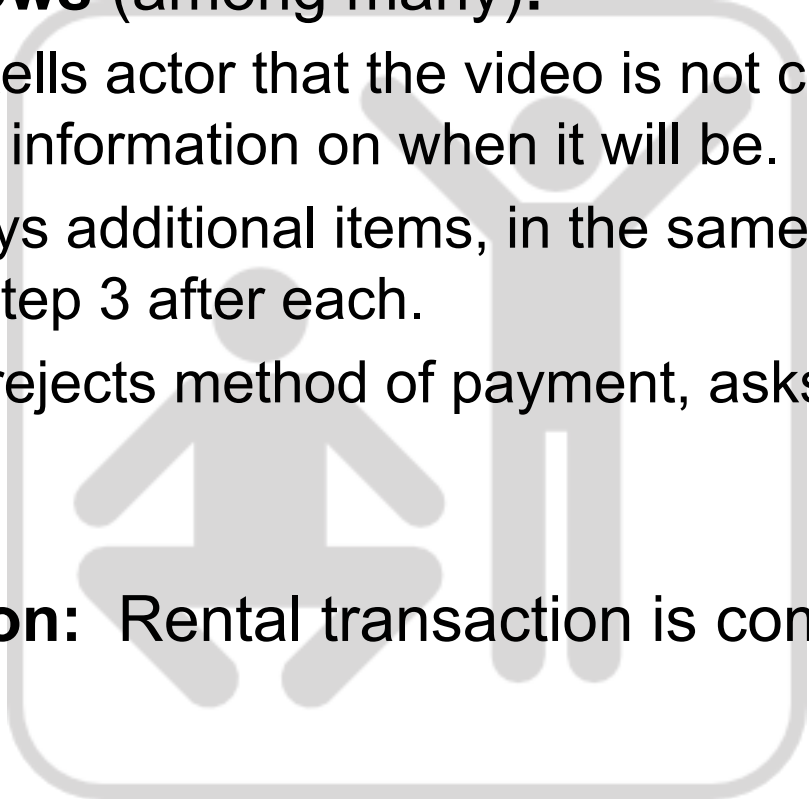
- **Preconditions:** Customer has a membership, has selected videos they want, and made system aware of their choices.
- **Actor:** Customer (self-service/remote), or store associate (in store)

■ 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)

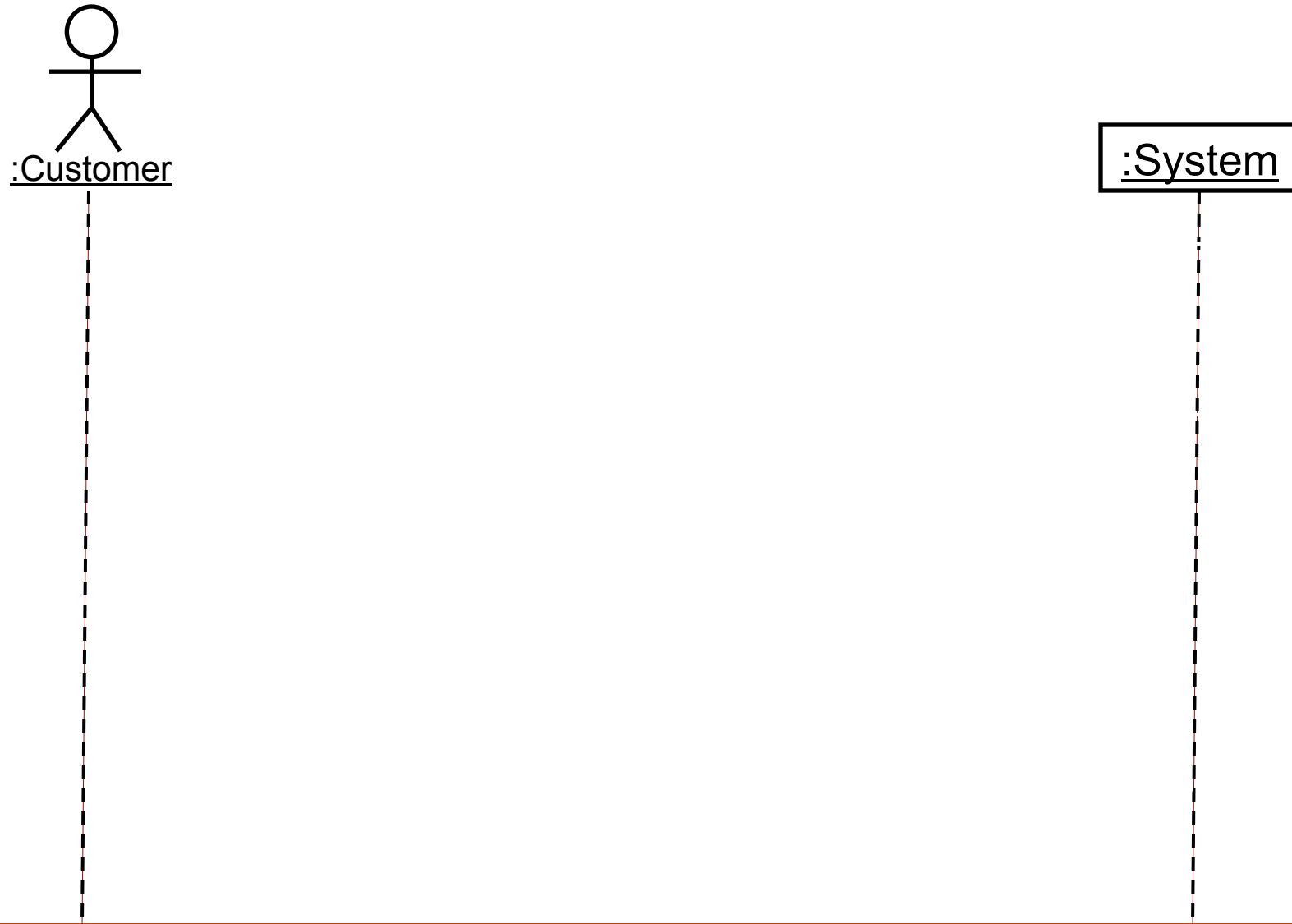


Homework 1: Basic Use Case **2/2**

- **Alternate flows** (among many):
 - 2a.** System tells actor that the video is not currently available, and provides information on when it will be.
 - 3a.** Actor buys additional items, in the same way, if desired, returning to step 3 after each.
 - 6a.** System rejects method of payment, asks actor for alternative.
 - **Postcondition:** Rental transaction is complete.
- 



SSD for Use Case 1





Homework and Milestone Reminders

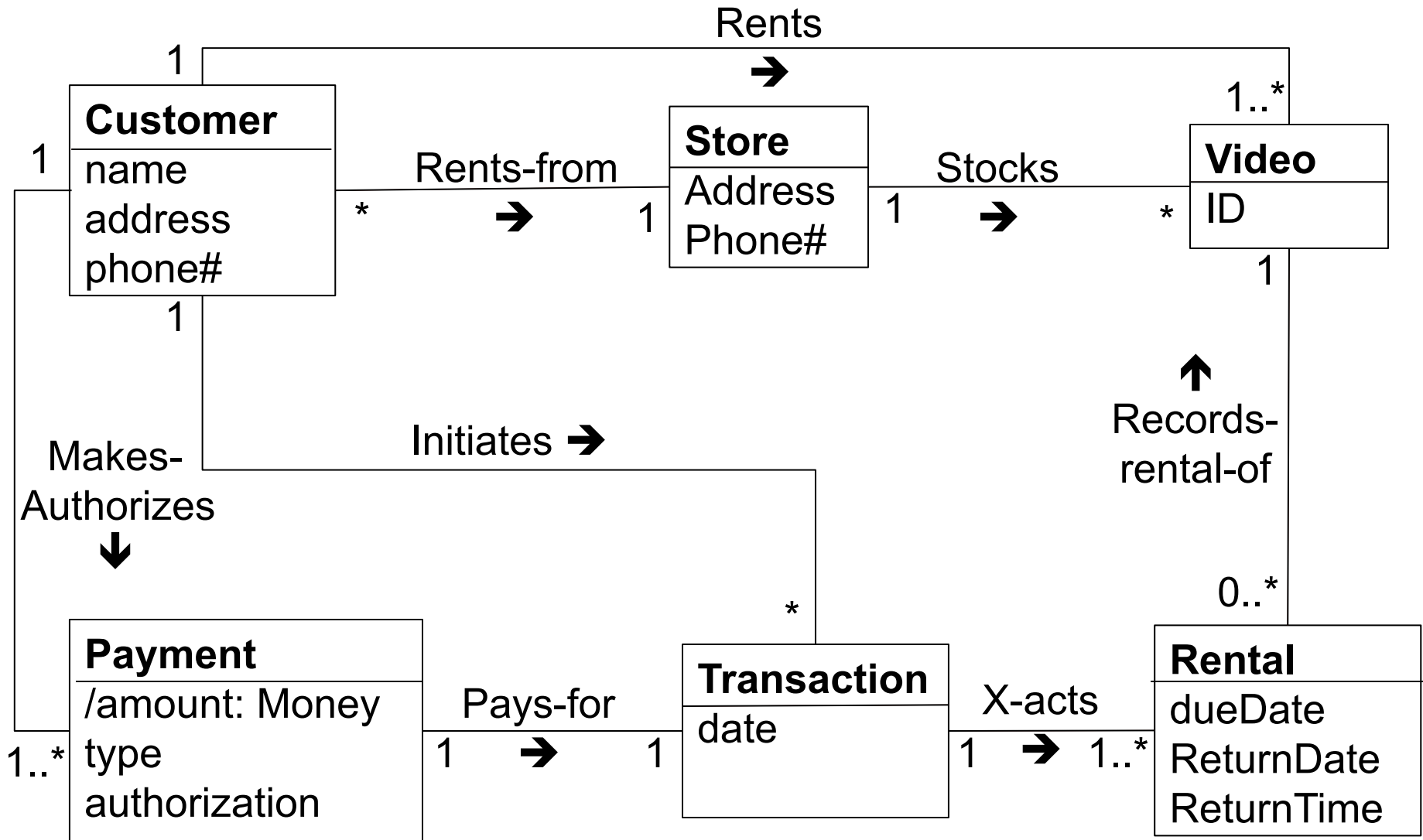
- Read Chapter 11 on Operations Contracts

- Homework 1 – Video Store Domain Model
 - Due by 5:00pm today, Tuesday, December 7th, 2010

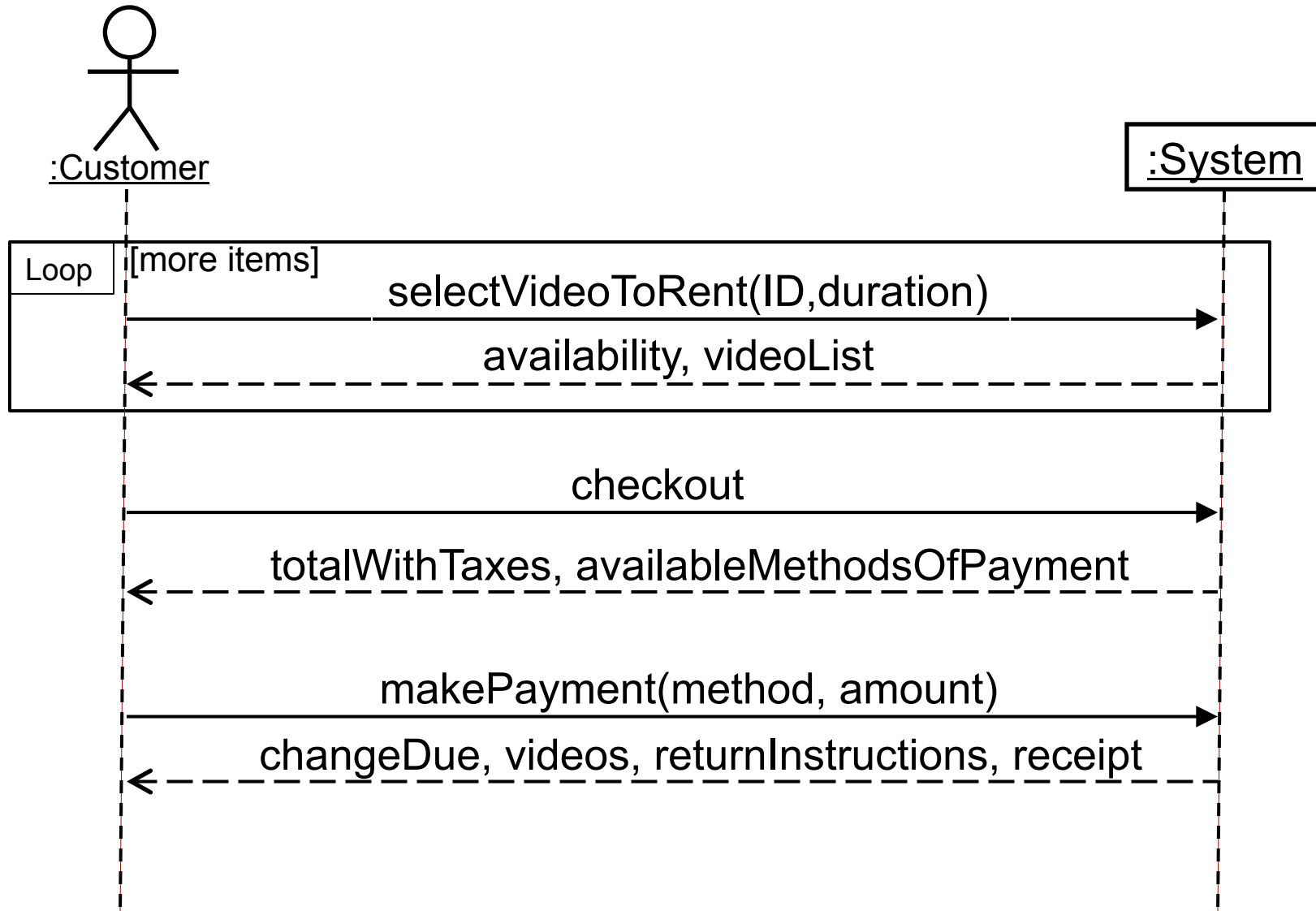
- Milestone 2 – Junior Project Domain Model
 - Due by 11:55pm on Friday, December 10th, 2010

- Homework 2 – Video Store SSDs and Operations Contracts
 - Due by 5:00pm on Tuesday, December 14th, 2010

Recall: DM Exercise from Yesterday



SSD for Use Case 1





Revisit: Using Data Type Attributes

- Primitive data types: Boolean, Real, Integer, ...
- Compound types more complex, but not domain specific: Address, Phone Number, ...
- If it's domain specific, use a class and association

Intuition from code: a “data type” is a primitive type, or a complex type where for instances a and b , $a.equals(b)$ doesn't imply $a == b$