3 Elaboration Iteration 1
Basics
First Week

庞雄文
Tel：18620638848
Wechat: augepang
QQ: 443121909

Contents

- Part2:
- 3.3 Other requirement
- Part3: Elaboration Iteration 1 Basics
- 4.1 Elaboration Iteration 1 and Case Studies
- 4.2 Domain Models
- 4.3 System Sequence Diagrams
- homework

3.3 Other requirement

- **Use cases aren't the whole story. There are a few other important UP requirement artifacts**
  - **The Supplementary Specification** captures and identifies other kinds of requirements, such as reports, documentation, packaging, supportability, licensing, and so forth**.**
  - **The Glossary** captures terms and definitions; it can also play the role of a data dictionary.
  - **The Vision** summarizes the "vision" of the project an executive summary. It serves to tersely communicate the big ideas**.**
  - **The Business Rules (or Domain Rules)** capture long-living rules or policies, such as tax laws, that transcend one particular application.
- **Question:**
  - **Should We Analyze These Thoroughly During Inception?**
    - It is useful to understanding the non-functional requirements (such as performance or reliability), as these have a significant impact on architectural choices

3.3 Other requirement

Supplementary Specification

- **Supplementary Specification captures** other requirements, information, and constraints not easily captured in the use cases or Glossary, including system-wide "URPS+" (usability, reliability, performance, supportability, and more) quality attributes or requirements
- **Elements of the Supplementary Specification include**
  - FURPS+ requirements functionality, usability, reliability, performance, and supportability
  - reports
  - hardware and software constraints (operating and networking systems, …)
  - development constraints (for example, process or development tools)
  - other design and implementation constraints
  - internationalization concerns (units, languages)
  - documentation (user, installation, administration) and help

3.3 Other requirement

- **Supplementary Specification**
  - **Elements of the Supplementary Specification include**
    - licensing and other legal concerns
    - packaging
    - standards (technical, safety, quality)
    - physical environment concerns (for example, heat or vibration)
    - operational concerns (for example, how do errors get handled, or how often should backups be done?)
    - application-specific domain rules
    - information in domains of interest (for example, what is the entire cycle of credit payment handling?)
  - Functionality in **Supplementary Specification?**
    - Some functions or features don't fit in a use case format. such as "add EJB Entity Bean 1.0 support."
    - UP allows this feature-oriented approach to requirements, in which case the feature list goes in the Supplementary **Specification**

3.3 Other requirement

- **Supplementary Specification**
  - **Quality Attributes**
    - system-wide quality are especially interesting because architectural analysis and design are largely concerned with the identification and resolution of the quality attributes in the context of the functional requirements.
  - **Application-Specific Domain (Business) Rules**
    - broad domain rules such as tax laws , and more narrow application-specific rules, such as how to calculate a line-item discount, can be recorded in the Supplementary Specification.

3.3 Other requirement

- **NextGen Example: (Partial) Supplementary Specification**
  - **Usability, Section 7.4**
    - **Human Factors**
      - The customer will be able to see a large-monitor display of the POS. Therefore:
        - » Text should be easily visible from 1 meter.
        - » Avoid colors associated with common forms of color blindness.
        - » Speed, ease, and error-free processing are paramount in sales processing, as the buyer wishes to leave quickly, or they perceive the purchasing experience (and seller) as less positive.
        - » The cashier is often looking at the customer or items, not the computer display. Therefore, signals and warnings should be conveyed with sound rather than only via graphics.

3.3 Other requirement

- Supplementary Specification in RUP

Collegiate Sports Paging System

Supplementary Specification

Version 1.0

Revision History

Date	Versi
October 10, 1999	1.0

Table of Contents

Introduction

Purpose

The purpose of this document is to
use-case model. The Supplementa

Scope

This Supplementary Specification

The Collegiate Sports Paging Syst

Table of Contents

3.3 Other requirement

Vision

- an executive summary that briefly describes the project, as a context for the major players to establish a common vision of the project
- **Elements of the vision include:**
  - **Introduction**
  - **Positioning:** Business Opportunity, Problem Statement, Product Position Statement, Alternatives and Competition..
  - **Stakeholder Descriptions:** Market Demographics, Stakeholder (Non-User) Summary, User Summary, Key High-Level Goals and Problems of the Stakeholders, User-Level Goals, User Environment…
  - **Product Overview**: Product Perspective, Summary of Benefits, Assumptions and Dependencies, Cost and Pricing, Licensing and Installation…
  - **Summary of System Features**
  - **Other Requirements and Constraints**

3.3 Other requirement

- **Commentary: Vision**
  - **The Key High-Level Goals and Problems of the Stakeholders**
    - summarizes the goals and problems at a high level often higher than specific use cases and reveals important non-functional and quality goals that may belong to one use case or span many, such as:
      - We need fault-tolerant sales processing.
      - We need the ability to customize the business rules
  - **Summary of System Features**
    - system features are high-level, terse statements summarizing system functions. Features are behavioral functions a system can do. They should pass this linguistic test:
      - The system does <feature X>.
      - For example: the system does payment authorization.
    - Guideline
      - less than 10 features is desirable. If more, consider grouping and abstracting the features.

3.3 Other requirement

- **NextGen Example: Vision**
  - **Stakeholder Descriptions**
    - **Key High-Level Goals and Problems of the Stakeholders**
      - one-day requirements workshop with subject matter experts and other stakeholders, and surveys at several retail outlets led to identification of the following key goals and problems:

High-Level Goal	Priority	Problems and Concerns	Current Solutions
Fast, robust, integrated sales processing	high	Reduced speed as load increases. Loss of sales processing capability if components fail. Lack of up-to-date and accurate information from accounting and other systems due to non-integration with existing accounting, inventory, and HR systems. Leads to difficulties in measuring and planning. Inability to customize business rules to unique business requirements. Difficulty in adding new terminal or user interface types (for example, mobile PDAs).	Existing POS products provide basic sales processing, but do not address these problems.
...	...	...	...

3.3 Other requirement

- Vision in RUP

Date	Version	
October 6, 1999	1.0	Initial vers

Table of Contents

Table of Contents

- Introduction
- Positioning
- Stakeholder and User Descriptions
- Product Overview
- Product Features
- Constraints
- Quality ranges
- Precedence and Priority
- Other Product Requirements
- Documentation Requirements

3.3 Other requirement

■ Glossary (Data Dictionary)

➢**list of noteworthy terms and their definitions.** During inception the glossary should be a simple document of terms and descriptions. During elaboration, it may expand into a data dictionary**.**

Term	Definition and Information	Format	Validation Rules	Aliases
item	A product or service for sale			
payment authorization	Validation by an external payment authorization service that they will make or guarantee the payment to the seller.			
payment authorization request	A composite of elements electronically sent to an authorization service, usually as a char array. Elements include: store ID, customer account number, amount, and timestamp.			
UPC	Numeric code that identifies a product. Usually symbolized with a bar code placed on products. See www.uc-council.org for details of format and validation.	12-digit code of several subparts.	Digit 12 i s a check digit.	Universal Product Code

3.3 Other requirement

- Glossary in RUP

Collegiate Sports Paging System

Glossary

Version 2.0

Revision History

Date	Version	Description	Author
October 12, 1999	1.0	Initial version	Context Integration
November 12, 1999	2.0	Update after Elaboration iteration	Context Integration

Table of Contents

Introduction

Purpose

The glossary contains the working definitions for all classes in the Collegiate Sports Paging System. This glossary will be expanded throughout the life of the project.

Scope

This glossary addresses all terms which have specific meanings for this project. Actors are not listed here as they are described more fully in the use case definitions.

References

None.

Definitions

>Advertiser Profile

Information about an advertiser and their contract with WebNewsOnLine. Includes advertiser name and address, balance due less than 30 days, balance due greater than 30 days, pricing information (maintained by WebNewsOnLine advertising department), and account status (active, inactive).

Category

Information about content. Includes major status (viewed, archived) and, if content is not archived, type of story (NCAA, PAC10, etc.). Types of stories are selected from a maintained list within the system, and are referenced by subscriber's Page-me-when profile.

Content

Content consists of all of the media by which a news story or sporting event story can be delivered to a user. This may include text, graphics, video, or sound.

Page

Sending of text information to a pager to inform subscriber of new content available on Web Site.

3.3 Other requirement

Business Rules (Domain Rules)

- Domain rules dictate how a domain or business may operate. They are **not requirements of any one application**. Company policies, physical laws and government laws are common domain rules.
- It's useful to identify and record domain rules in a separate application-independent artifact for share.

ID	Rule	Changeability	Source
RULE1	Signature required for credit payments.	Buyer "signature" will continue to be required, but within 2 years most of our customers want signature capture on a digital capture device, and within 5 years we expect there to be demand for support of the new unique digital code "signature" now supported by USA law.	The policy of virtually all credit authorization companies.

3.3 Other requirement

Evolutionary Requirements in Iterative Methods

Inception

- the Vision summarizes the project idea in a form to help decision makers determine if it is worth continuing, and where to start
- the Supplementary Specification should be only lightly developed during inception, highlighting noteworthy quality attributes that expose major risks and challenges (for example, the NextGen POS must have recoverability when external services fail)

Elaboration

- Complete use cases, Supplementary Specification, and a Vision to reflects the stabilized major features and other requirements

3.3 Other requirement

- Sample UP artifacts and timing. s - start; r - refine

Discipline	Artifact	Incep.	Elab.	Const.	Trans.
	Iteration	I1	E1..En	C1..Cn	T1..T2
Business Modeling	Domain Model		s		
Requirements	Use-Case Model	s	r		
	Vision	s	r		
	Supplementary Specification	s	r		
	Glossary	s	r		
	Business Rules	s	r		
Design	Design Model		s	r	
	SW Architecture Document		s		
	Data Model		s	r	

3.3 Other requirement

- Risk list ---example

Collegiate Sports Paging System

Risk List

Version 1.0

Revision History

Date	Version	Description	Author
October 4, 1999	1.0	Initial version	Context Integration

Topics

- Introduction
- Risks

Introduction

Purpose

This document describes the risks known to the Collegiate Spor

Scope

This risk lists addresses the entire Collegiate Sports Paging Sys

Definitions, Acronyms and Abbreviations

See Glossary document.

References

None.

Overview

The risks known at the publication date of this document are list

Risks

Technical Risk : Capacity and Capability

Technical Risk : Capacity and Capability

Risk Magnitude: Most Damaging

Description

Areas of risk include the inability to deliver a solution that meets capacity requirements or to issue a page to a paging devic

Impacts

System not functional, probably resulting in loss of subscribers.

Indicators

Failed or delayed delivery of messages within established time frame of 5 minutes.

Mitigation Strategy

Context has provided similar pager capability for other projects, therefore this area of technical risk is relatively low. Contex
workloads, which are currently 200,000 to 500,000 subscribers. Context Integration will develop a scalable system. will pro
service levels within the desired specifications.

Contingency Plan

Attempt to locate a service that can, at peak processing time, accept and send up to 500,000 page requests.

3.3 Other requirement

■ Iterative Plan

Date	Version	Description	Author
October 6, 1999	1.0	Initial version	Context Integration

Table of Contents

Introduction

Purpose

This Iteration Plan describes the detailed plans for the Preliminary Iteration of the Project. During this iteration, the re
the business case for the system and will result in a decision on whether the project will proceed.

Scope

The Preliminary Iteration Plan applies to the project being developed by Context Integration for WebNewsOnLine. Th

Definitions, Acronyms and Abbreviations

See Glossary document.

References

1. CSPS Vision 1.0
2. CSPS Requirements Management Plan 1.0

Plan

The Preliminary Iteration will develop the product requirements and establish the business case for the Collegiate Sp
with the project based upon the business case.

Iteration Tasks

The following table illustrates the tasks with their planned start and end dates.

Task	Start	End

Collegiate Sports Paging Service

Iteration Plan

Version 1.0

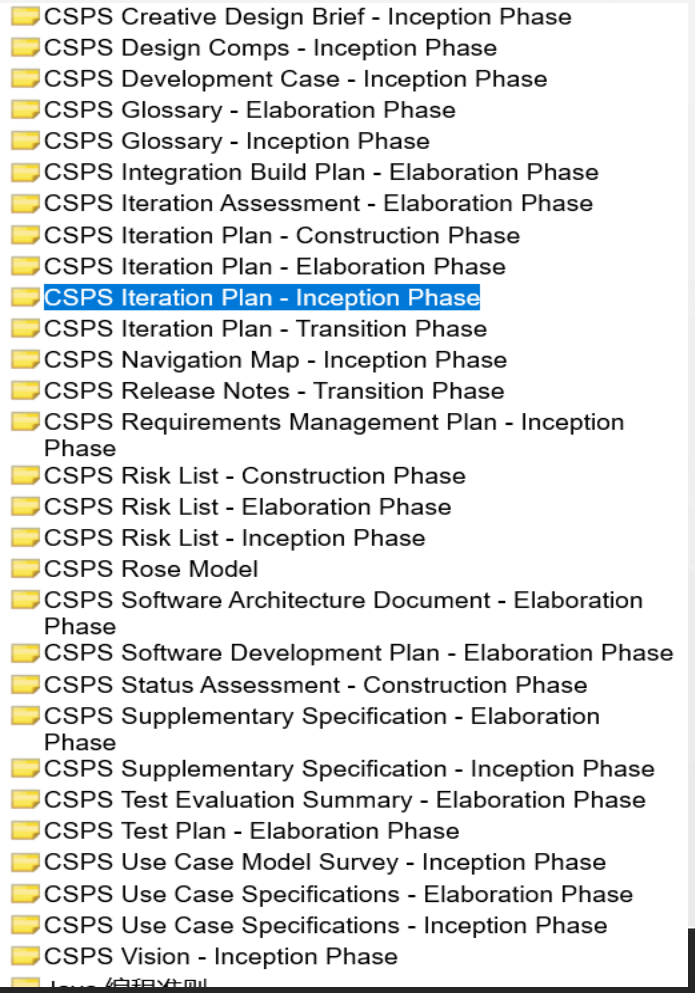
Iteration Tasks

The following table illustrates the tasks with their planned start and end dates.

Task	Start	End
INCEPTION	Fri 10/1/99	Mon 10/25/99
Begin Inception	Fri 10/1/99	Fri 10/1/99
Inception Kick-off	Mon 10/4/99	Wed 10/6/99
Add tasks to project plan for specific project technology using ContextWISE cartridges	Mon 10/4/99	Mon 10/4/99
Assemble Change Control Board	Mon 10/4/99	Tue 10/5/99
Create and baseline Change Control Plan	Tue 10/5/99	Tue 10/5/99
Obtain Sign-off	Tue 10/5/99	Tue 10/5/99
Inception Kick-off Meeting	Tue 10/5/99	Wed 10/6/99
Prepare for inception kick-off meeting	Tue 10/5/99	Wed 10/6/99
Hold Inception Kick-off Meeting	Wed 10/6/99	Wed 10/6/99
Inception Kick-off Completed	Wed 10/6/99	Wed 10/6/99
Inception Deliverables	Wed 10/6/99	Thu 10/14/99

3.3 Other requirement

- **Two** Completed cases in RUP
  - Collegiate Sports Paging System (CSPS)
  - 课程注册系统



How to write SRS?

Photo Album Editing	Version: 1.0
Software Requirements Specification	Date: 03/12/2006
BLLC-001	

Table of Contents

Part3: Elaboration Iteration 1 Basics

- **What we have in inception phase:**
  - most actors, goals, and use cases named
  - most use cases written in brief format; 10-20% of the use cases are written in fully dressed detail to improve understanding of the scope and complexity
  - most influential and risky quality requirements identified
  - version one of the Vision and Supplementary Specification written
  - Risk list
  - technical proof-of-concept prototypes and other investigations to explore the technical feasibility of special requirements ("Does Java Swing work properly on touch-screen displays?")
  - user interface-oriented prototypes to clarify the vision of functional requirements

4.1 Elaboration Iteration 1 and Case Studies

What should wo do in the elaboration phase?

- **The works:**
  - the core, risky software architecture is programmed and tested
  - the majority of requirements are discovered and stabilized
  - the major risks are mitigated or retired
- **Elaboration phase**
  - is the initial series of iterations, often consists of two or more iterations
  - is not a design phase or a phase when the models are fully developed in preparation for implementation in the construction step that would be an example of superimposing waterfall ideas on iterative development and the UP.
  - not creating throw-away prototypes; rather, the code and design are production-quality portions of the final system
- **In Summary:**
  - **Build the core architecture, resolve the high-risk elements, define most requirements,** and estimate the overall schedule and resources

4.1 Elaboration Iteration 1 and Case Studies

- **What should wo do in the elaboration phase?**
  - **Best Practice:**
    - do short time-boxed risk-driven iterations
    - start programming early
    - adaptively design, implement, and test the core and risky parts of the architecture
    - test early, often, realistically
    - adapt based on feedback from tests, users, developers
    - write most of the use cases and other requirements in detail, through a series of workshops, once per elaboration iteration

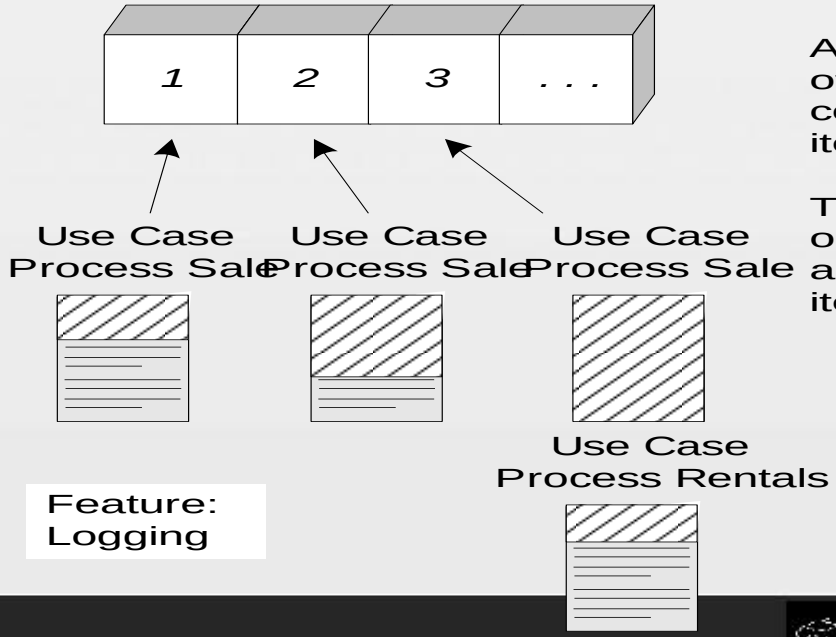
4.1 Elaboration Iteration 1 and Case Studies

- **What should wo do in the elaboration phase?**
  - **Artifacts in Elaboration**

Artifact	Comment
Domain Model	This is a visualization of the domain concepts; it is similar to a static information model of the domain entities.
Design Model	This is the set of diagrams that describes the logical design. This includes software class diagrams, object interaction diagrams, package diagrams, and so forth.
Software Architecture Document	A learning aid that summarizes the key architectural issues and their resolution in the design. It is a summary of the outstanding design ideas and their motivation in the system.
Data Model	This includes the database schemas, and the mapping strategies between object and non-object representations.
Use-Case Storyboards, UI Prototypes	A description of the user interface, paths of navigation, usability models, and so forth.

4.1 Elaboration Iteration 1 and Case Studies

- **In Iterative Development We Don't Implement All the Requirements at Once**
  - **Of course not.** we haven't done all the requirements analysis, only have 10-15% full dressed use cases.
  - **Incremental Development for the Same Use Case Across Iterations**
    - It is common to work on varying scenarios of the same use case over several iterations and gradually extend the system to ultimately handle all the functionality required



A use case or feature is often too complex to complete in one short iteration.

Therefore, different part or scenarios must be allocated to different iterations.

4.1 Elaboration Iteration 1 and Case Studies

Iteration 1 Requirements and Emphasis

NextGen PS

- Implement a basic, key scenario of the Process Sale use case: entering items and receiving a cash payment.
- Implement a Start Up use case as necessary to support the initialization needs of the iteration.
- Nothing fancy or complex is handled, just a simple happy path scenario, and the design and implementation to support it.
- There is no collaboration with external services, such as a tax calculator or product database.
- No complex pricing rules are applied.
- The design and implementation of the supporting UI, database, and so forth, would also be done, but is not covered in any detail.

4.1 Elaboration Iteration 1 and Case Studies

Iteration 1 Requirements and Emphasis

Monoplay Game

- Implement a basic, key scenario of the Game use case: players moving around the squares of the board.
- Implement a Start Up use case as necessary to support the initialization needs of the iteration.
- Two to eight players can play.
- A game is played as a series of rounds.
- Play the game for only 20 rounds.
- In iteration-1 there is no money, no winner or loser, no properties to buy or rent to pay, and no special squares of any kind.
- Each square has a name. Every player begins the game with their piece located on the square named "Go." The square names will be Go, Square 1, Square 2, … Square 39
- Run the game as a simulation requiring no user input, other than the number of players.

4.1 Elaboration Iteration 1 and Case Studies

Iteration 1 Requirements and Emphasis

- iteration-1 is not architecture-centric or risk-driven
- helping people learn fundamental OOA/D and UML

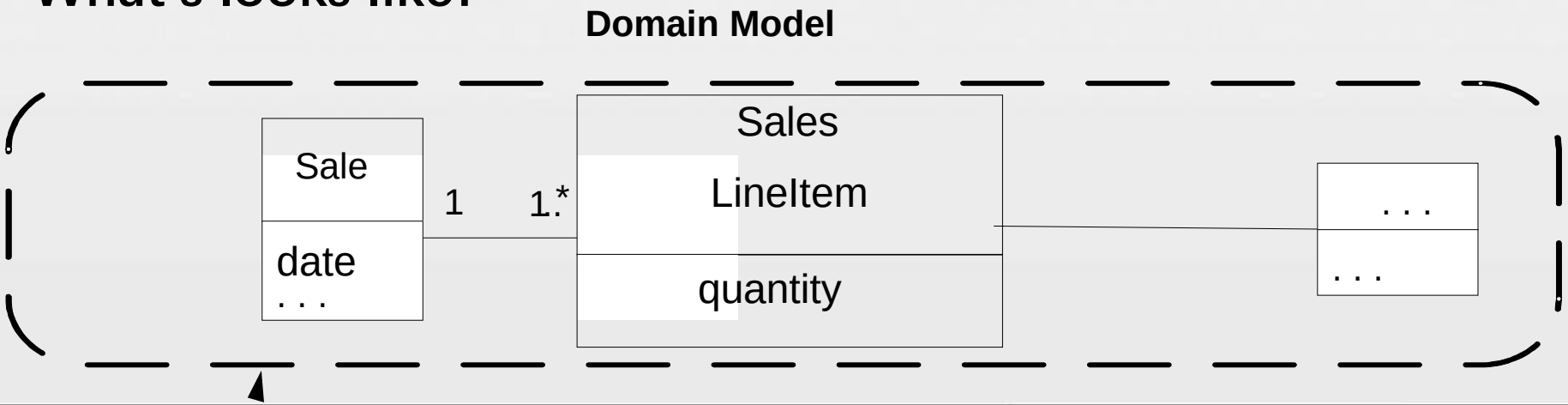
4.1 Elaboration Iteration 1 and Case Studies

You Know You Didn't Understand Elaboration When…

- It is more than "a few" months long for most projects.
- It only has one iteration (with rare exceptions for well-understood problems).
- Most requirements were defined before elaboration.
- The risky elements and core architecture are not being tackled.
- It does not result in an executable architecture; there is no production-code programming.
- It is considered primarily a requirements or design phase, preceding an implementation phase in construction.
- There is an attempt to do a full and careful design before programming.
- There is minimal feedback and adaptation; users are not continually engaged in evaluation and feedback.
- There is no early and realistic testing.
- The architecture is speculatively finalized before programming.
- It is considered a step to do the proof-of-concept programming, rather than programming the production core executable architecture

4.2 Domain Models

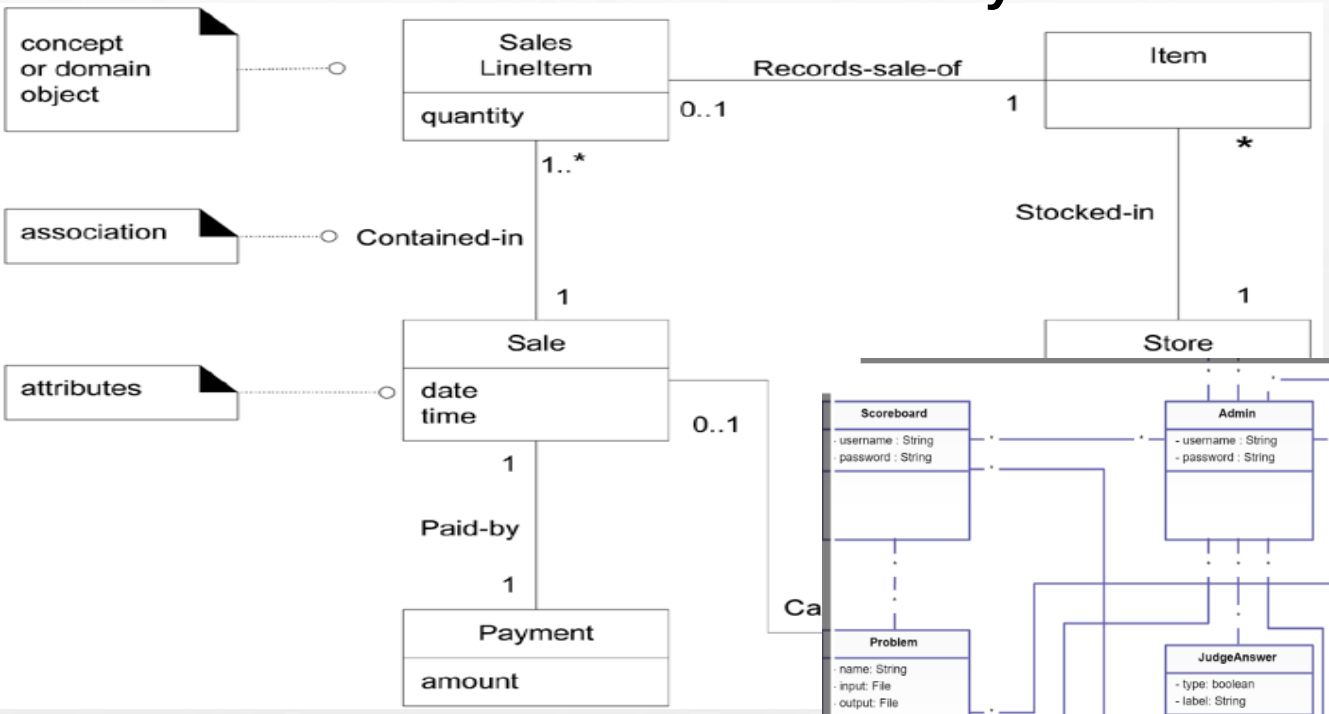
- **domain model is** the most important and classic model in OO analysis, It illustrates noteworthy **concepts** in a domain. It can act as a source of inspiration for designing some software objects
- **Goal of domain model**
  - **Identify conceptual classes related to the current iteration.**
  - **Create an initial domain model.**
  - **Model appropriate attributes and associations.**
- **What's looks like?**

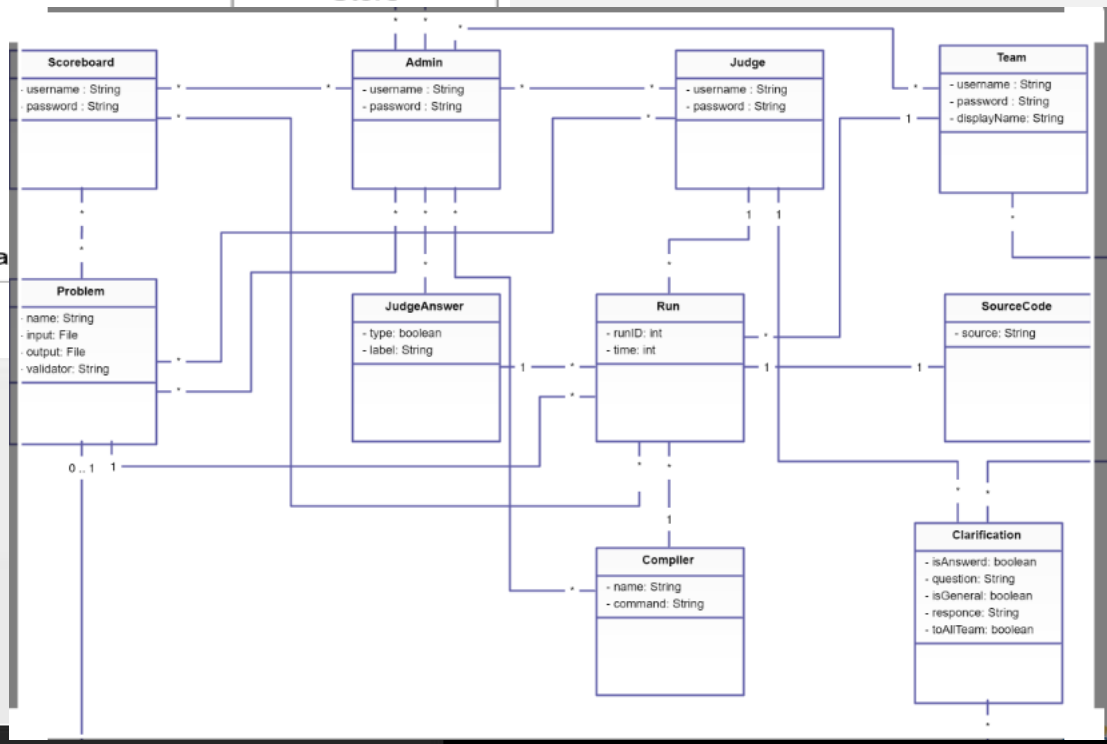


4.2 Domain Models

- Example

Identifying a rich set of conceptual classes is at the heart of OO analysis





4.2 Domain Models

What is a Domain Model?

- **A domain model** is a visual representation of conceptual classes or real-situation objects in a. it also been **called conceptual models, domain object models, and analysis object models**
- **"Domain Model" means a representation of real-situation conceptual classes, not of software objects.**

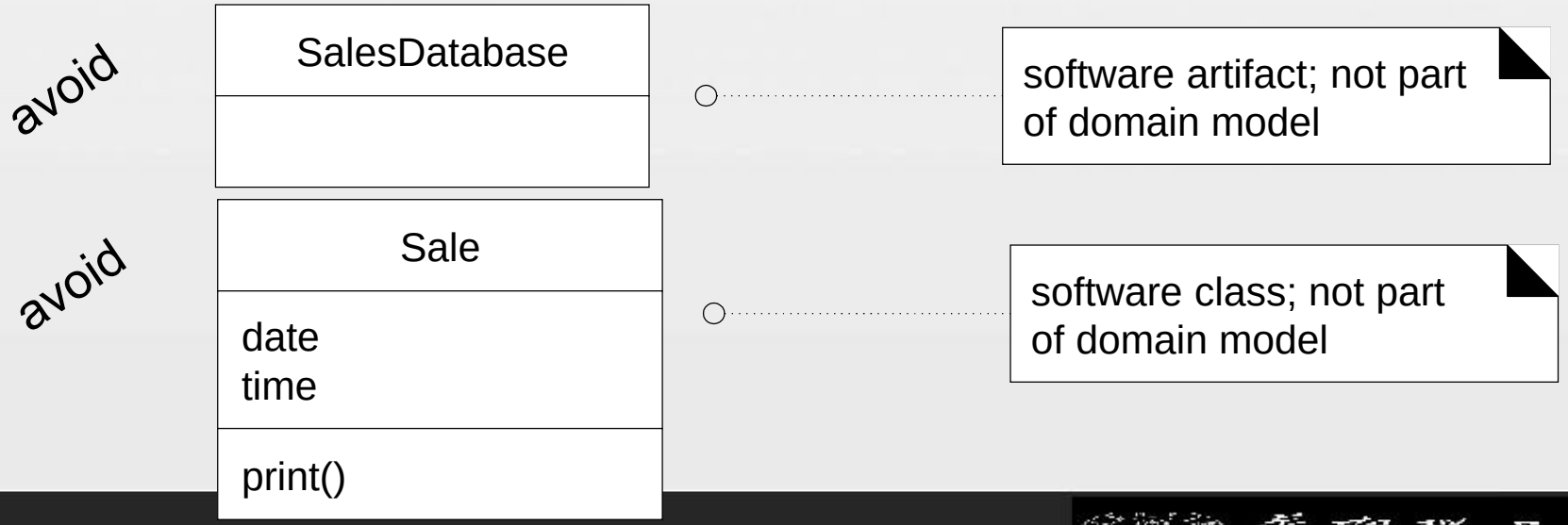
- Applying UML notation, a domain model is illustrated with a set of class diagrams in which no operations (method signatures) are defined. It show:
  - domain objects or conceptual classes
  - associations between conceptual classes
  - attributes of conceptual classes

4.2 Domain Models

What is a Domain Model?

Domain Model is not the Picture of Software Business Objects

- Domain Model is a visualization of real-situation domain , not of software objects such as Java or C# classes, or software objects with responsibilities **. Following are not :**
  - Software artifacts, such as a window or a database, unless the domain being modeled is of software concepts, such as a model of graphical user interfaces.
  - Responsibilities or methods.



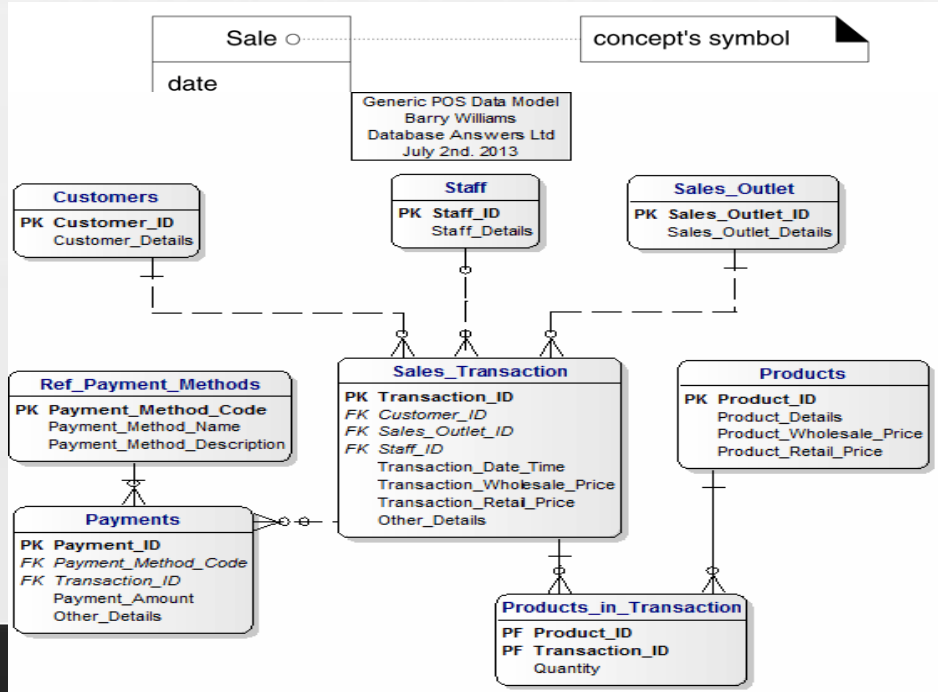
4.2 Domain Models

What are Conceptual Classes?

- a conceptual class is an idea, thing, or object. More formally, a conceptual class may be considered in terms of its symbol, intension, and extension
  - Symbol words or images representing a conceptual class.
  - Intension the definition of a conceptual class.
  - Extension the set of examples to which the conceptual class applies.

domain model is not data model, maybe include:

- requirements don't indicate any obvious need to remember information about it
- attributeless conceptual classes
- conceptual classes that have a purely behavioral role



4.2 Domain Models

Why Create a Domain Model?

- **domain model is helpful to understand the key concepts and vocabulary in the real situation.**
- If use the similar name between the domain model and the domain layer supported a lower gap between the software representation and our mental model of the domain.

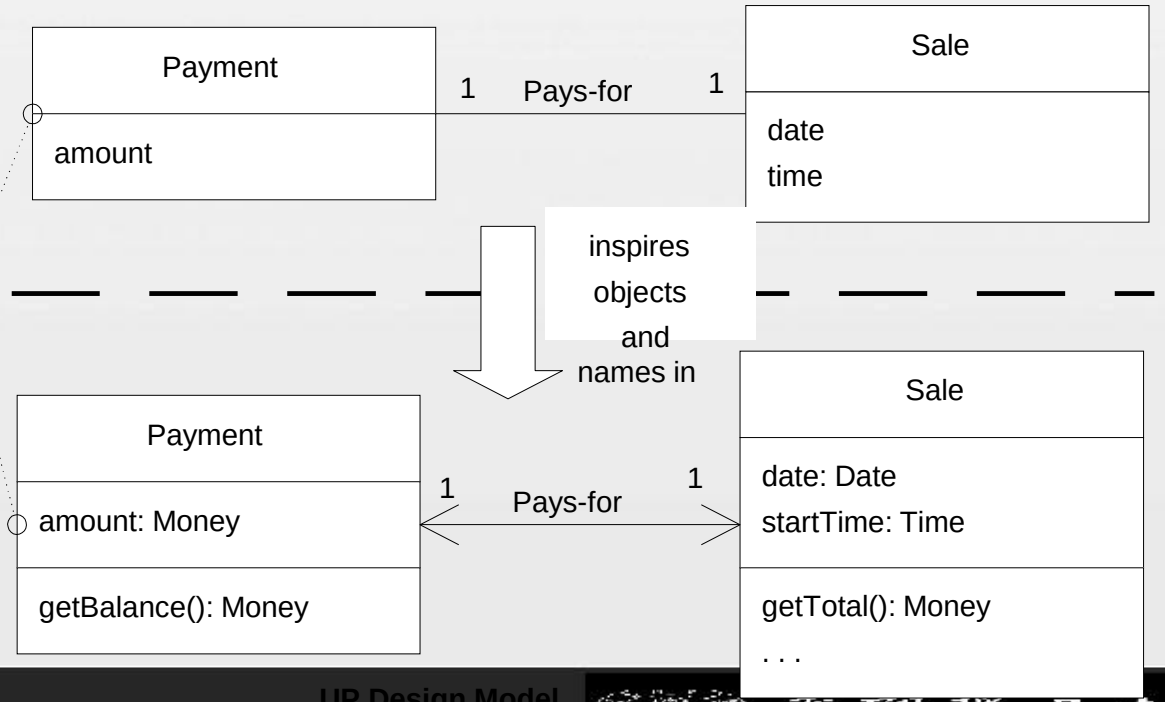
A Payment in the Domain Model is a concept, but a Payment in the Design Model is a software class. They are not the same thing, but the former *inspired* the naming and definition of the latter.

This reduces the representational gap.

This is one of the big ideas in object technology.

UP Domain Model

Stakeholder's view of the noteworthy concepts in the domain.



4.2 Domain Models

How to Create a Domain Model?

- Guideline:
  - **Find the conceptual classes (see a following guideline).**
  - **Draw them as classes in a UML class diagram.**
  - **Add associations and attributes.**

How to Find Conceptual Classes?

- Reuse or modify existing models. There are published, well-crafted domain models and data models (which can be modified into domain models) for many common domains, such as inventory, finance, health, and so forth.
- Use a category list
- Identify noun phrases

4.2 Domain Models

■ Find Conceptual Classes, Method 2: Use a Category List

Conceptual Class Category	Examples
business transactions Guideline: These are critical (they involve money), so start with transactions.	Sale, Payment Reservation
transaction line items Guideline: Transactions often come with related line items, so consider these next.	Sales LineItem
product or service related to a transaction or transaction line item Guideline: Transactions are for something (a product or service). Consider these next.	Item Flight, Seat, Meal
where is the transaction recorded? Guideline: Important.	Register, Ledger FlightManifest
roles of people or organizations related to the transaction; actors in the use case Guideline: We usually need to know about the parties involved in a transaction.	Cashier, Customer, Store Monopoly Player Passenger, Airline
place of transaction; place of service	Store, Airport, Plane, Seat
noteworthy events, often with a time or place we need to remember	Sale, Payment MonopolyGame

4.2 Domain Models

Find Conceptual Classes, Method 2: ~~Use a Category List~~

physical objects Guideline: This is especially relevant when creating device-control software, or simulations.	Item, Register Board, Piece, Die Airplane
descriptions of things Guideline: See p. 147 for discussion.	ProductDescription FlightDescription
catalogs Guideline: Descriptions are often in a catalog.	ProductCatalog FlightCatalog
containers of things (physical or information)	Store, Bin Board, Airplane
things in a container	Item Square (in a Board) Passenger
other collaborating systems	CreditAuthorizationSystem AirTrafficControl
records of finance, work, contracts, legal matters	Receipt, Ledger MaintenanceLog
financial instruments	Cash, Check, LineOfCredit TicketCredit
schedules, manuals, documents that are regularly referred to in order to perform work	DailyPriceChangeList RepairSchedule

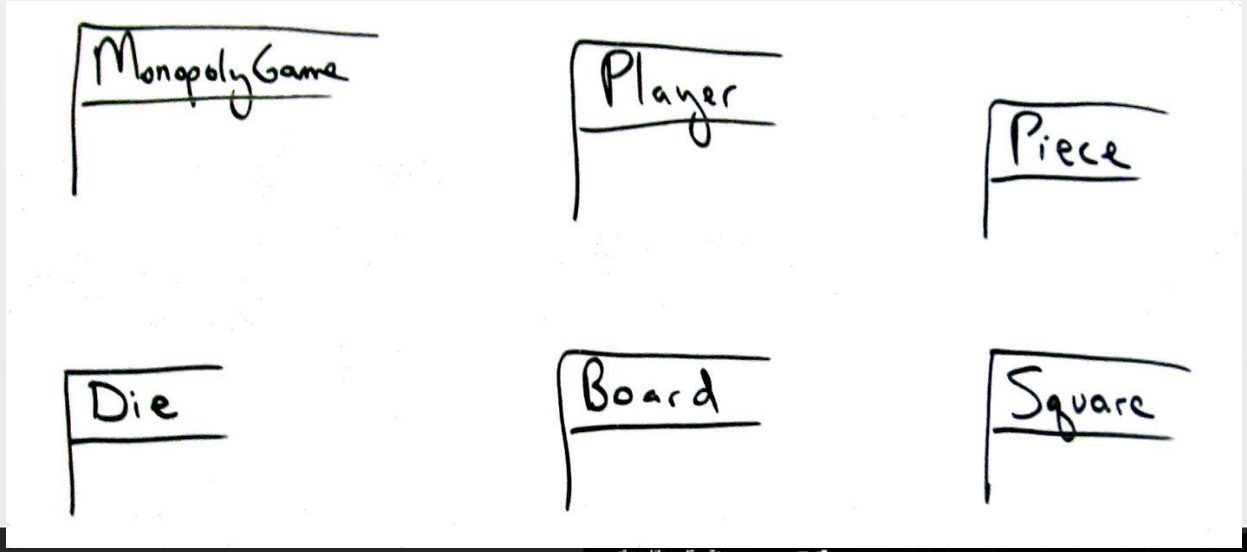
4.2 Domain Models

- **Find Conceptual Classes, Method 3: Noun Phrase Identification**
  - **Identify the nouns and noun phrases in textual descriptions of a domain and consider them as candidate conceptual classes or attributes**
  - **Example. Main Success Scenario (or Basic Flow):**
    - **Customer** arrives at a **POS checkout** with **goods** and/or **services** to purchase.
    - **Cashier** starts a new **sale**.
    - **Cashier** enters **item identifier**.
    - System records **sale line item** and presents **item description**, price, an running total. Price calculated from a set of price rules.
    - **Cashier** repeats the above steps until indicates done**.**
    - System presents **total** with **taxes** calculated.
    - **Cashier** tells **Customer** the total, and asks for **payment**.
    - **Custome**r pays and System handles payment.
    - ...

4.2 Domain Models

- **Example: Find and Draw Conceptual Classes**

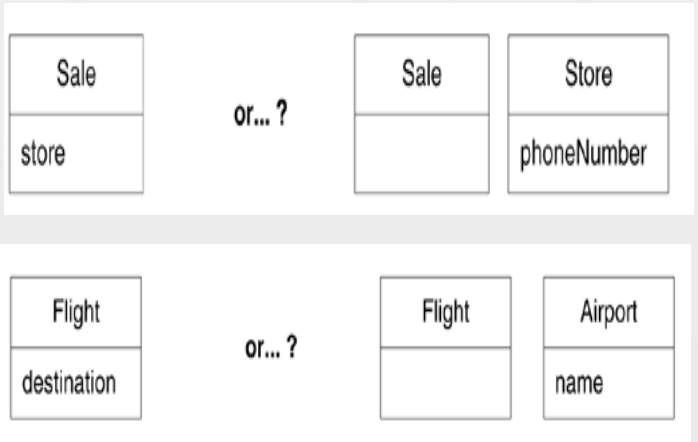
Register	Item	Store	Sale
Sales LineItem	Cashier	Customer	Ledger
Cash Payment	Product Catalog	Product Description	



4.2 Domain Models

- **Some guidelines for domain models:**
  - Agile Modeling: Sketching a Class Diagram
  - Agile Modeling: Maintain the Model in a Tool?
  - Report Objects Include 'Receipt' in the Model?
  - Think Like a Mapmaker; Use Domain Terms

- **Attributes vs. Classes?**
  - If we do not think of some conceptual class X as a number or text in the real world, X is probably a conceptual class, not an attribute.
  - should store be an attribute of Sale, or a separate conceptual class Store?
  - Should destination be an attribute of Flight, or a separate conceptual class Airport?



4.2 Domain Models

When to Model with 'Description' Classes?

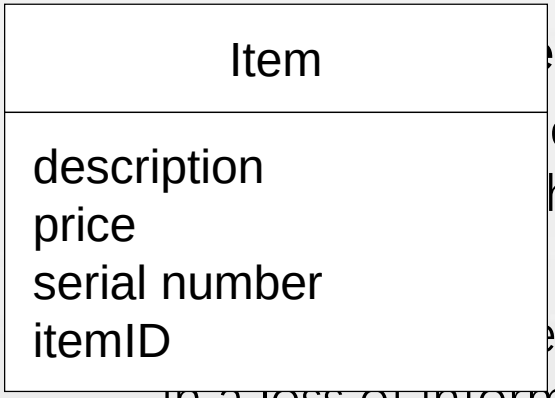
- description class contains information that describes something else. For example, a ProductDescription that records the price, picture, and text description of an Item

- **Motivation: Why Use 'Description' Classes**
  - Assume the following
    - An Item instance represents a physical item in a store; as such, it may even have a serial number.
    - An Item has a description, price, and itemID, which are not recorded anywhere else.
    - Every time a real physical item is sold, a corresponding software instance of Item is deleted from "software land."
  - With these assumptions, If someone asks, "How much do ObjectBurgers cost?", no one can answer

4.2 Domain Models

When to Model with 'Description' Classes?

- The problem illustrates the need for objects that are descriptions (sometimes called specifications) of other things.
- need for description classes is common in sales, product, and service domains. It is also common in manufacturing, which requires a description of a manufactured thing that is distinct from the thing itself



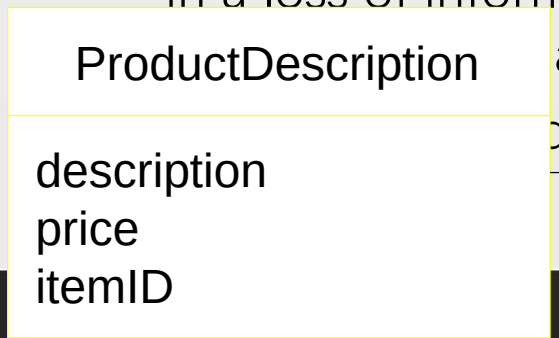
escription class when:

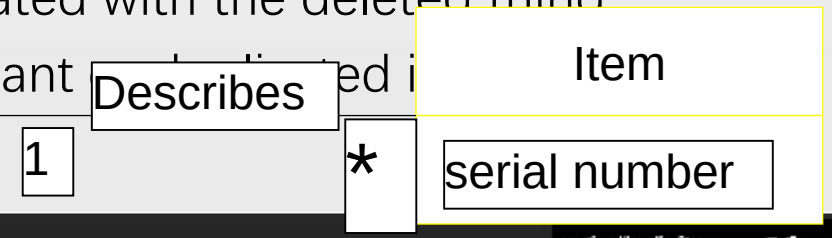
e a description about an item or service,
he current existence of any examples of those items

Worse

es of things they describe (for example, Item) results
in a loss of information that needs to be maintained, but was
ated with the deleted thing

dant





Better

4.2 Domain Models

Associations

- Concepts are not isolated, there are various relationships between them
- **An association is a relationship between classes (more precisely, instances of those classes)** that indicates some meaningful and interesting connection.
  - In the UML, associations are defined as "**the semantic relationship between two or more classifiers that involve connections among their instances.**"
- **When to Show an Association?**
  - Associations for which knowledge of the relationship needs to be preserved for some duration ("need-to-remember" associations).
  - Associations derived from the **Common Associations List**.
  - BUT
    - **Why Should We Avoid Adding Many Associations?**

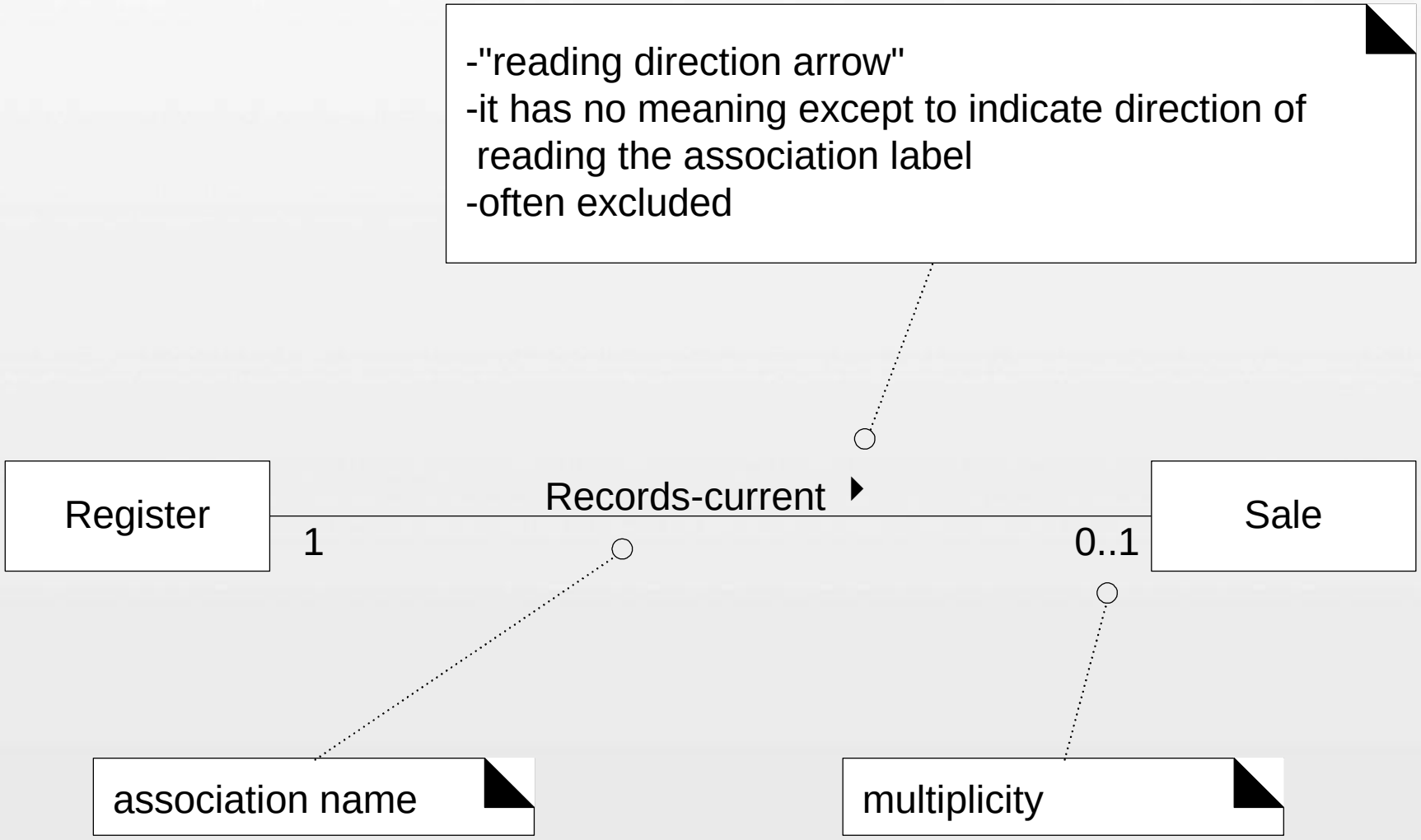
4.2 Domain Models

- **Associations**
  - **Partial Common Associations List**

Category	Examples
A is a transaction related to another transaction B	CashPaymentSale CancellationReservation
A is a line item of a transaction B	SalesLineItemSale
A is a product or service for a transaction (or line item) B	ItemSalesLineItem (or Sale) FlightReservation
A is a role related to a transaction B	CustomerPayment PassengerTicket
A is a physical or logical part of B	DrawerRegister SquareBoard SeatAirplane
A is physically or logically contained in/on B	RegisterStore, ItemShelf SquareBoard PassengerAirplane
……	

4.2 Domain Models

Applying UML: Association Notation



4.2 Domain Models

Applying UML: Association Notation

How to Name an Association?

- Name an association based on a ClassName-VerbPhrase-ClassName format where the verb phrase creates a sequence that is readable and meaningful.

Roles

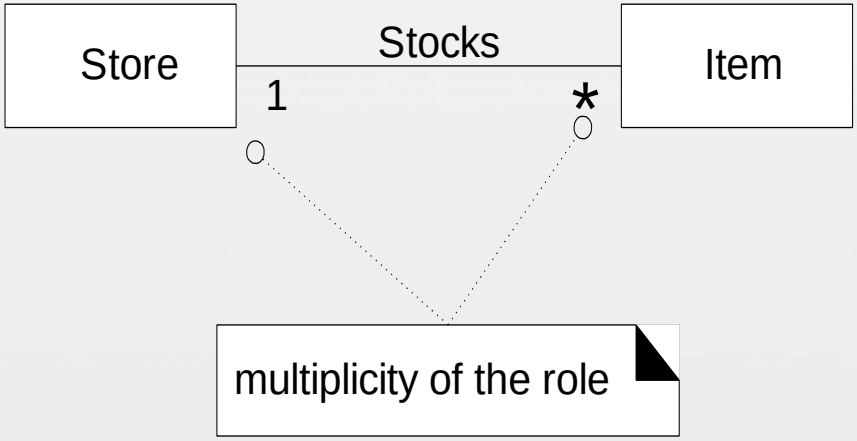
- **Each end of an association is called a role. Roles may optionally have:**
  - multiplicity expression
  - name
  - navigability

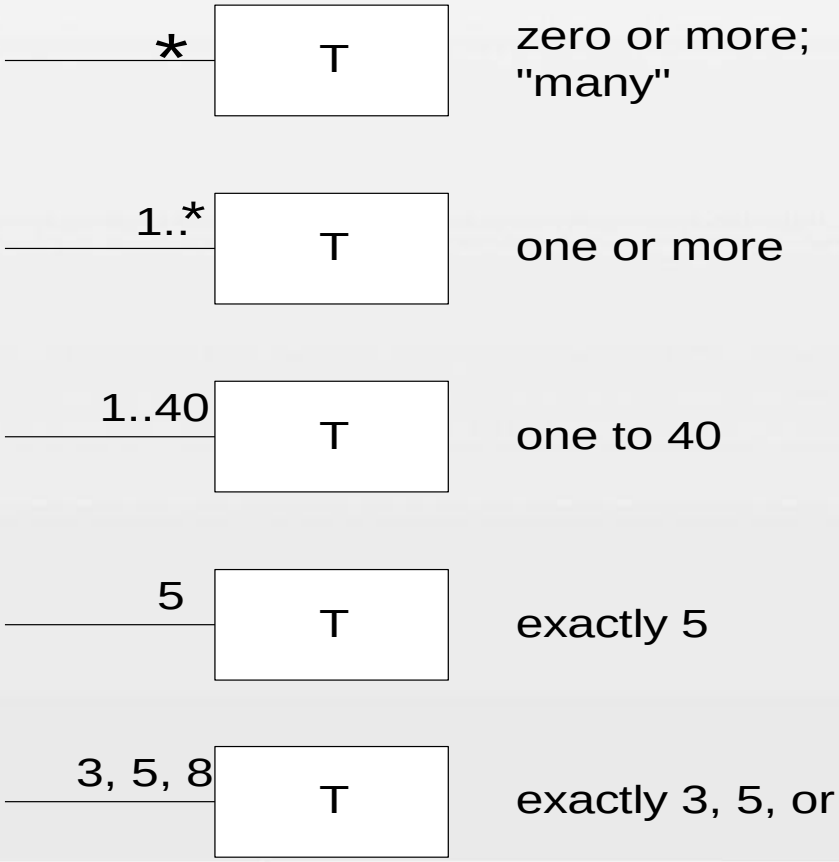
4.2 Domain Models

Applying UML: Association Notation

Multiplicity

- Multiplicity defines how many instances of a class A can be associated with one instance of a class B

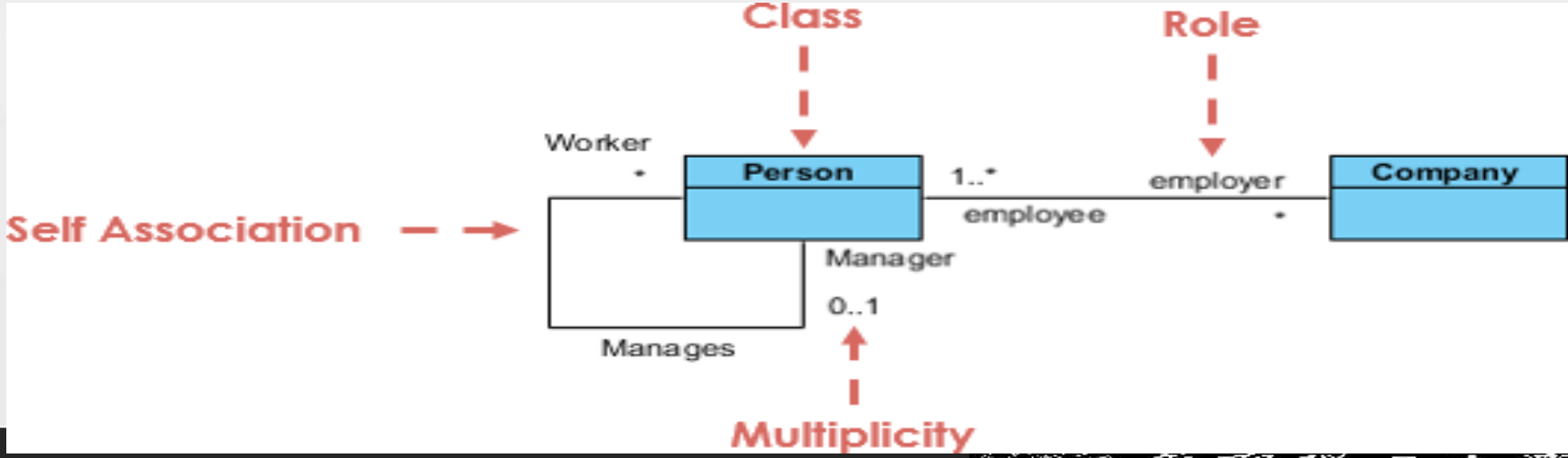




4.2 Domain Models

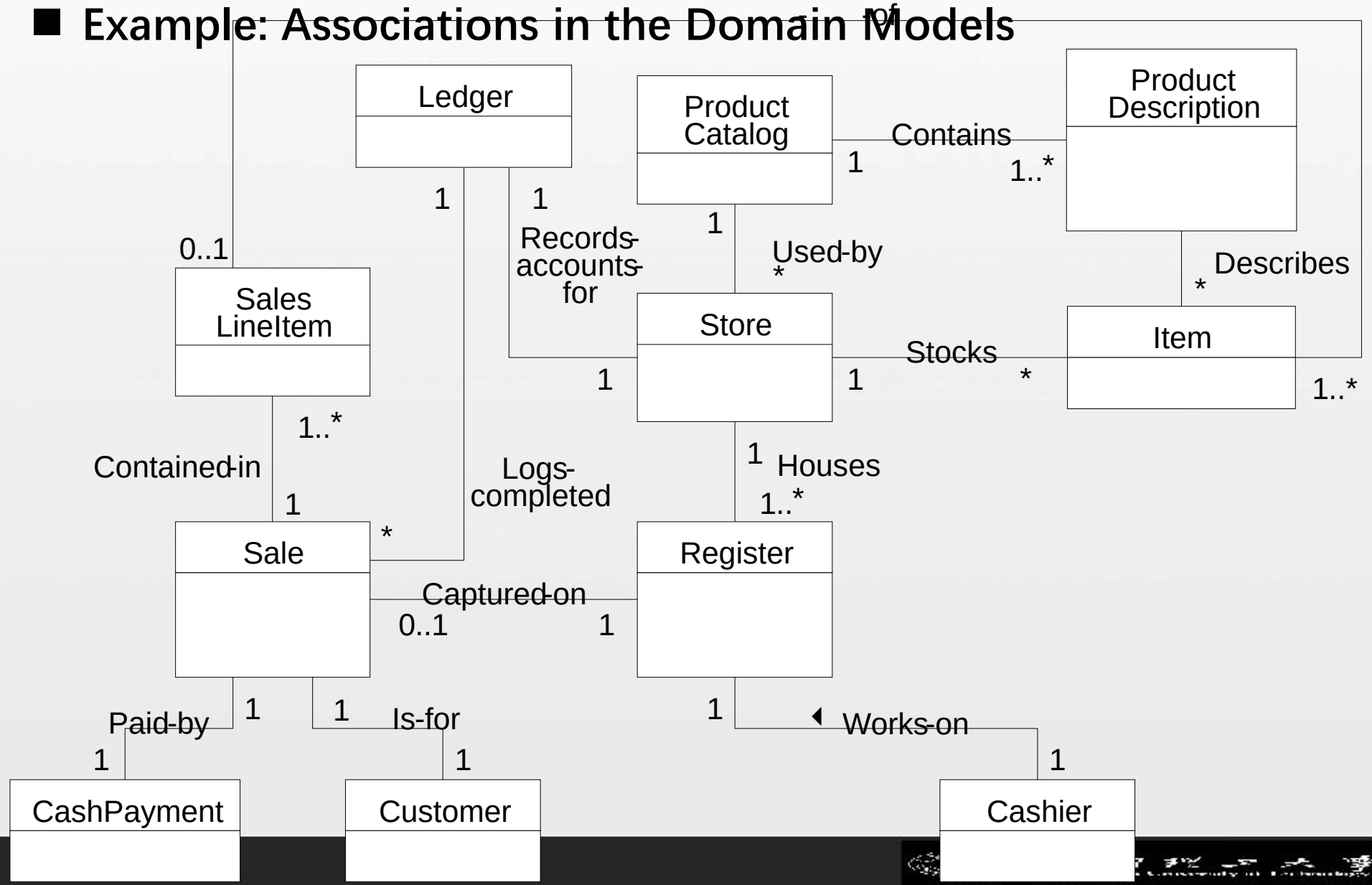
- Applying UML: Association Notation
  - Multiple Associations Between Two Classes

  - Self-association of one class



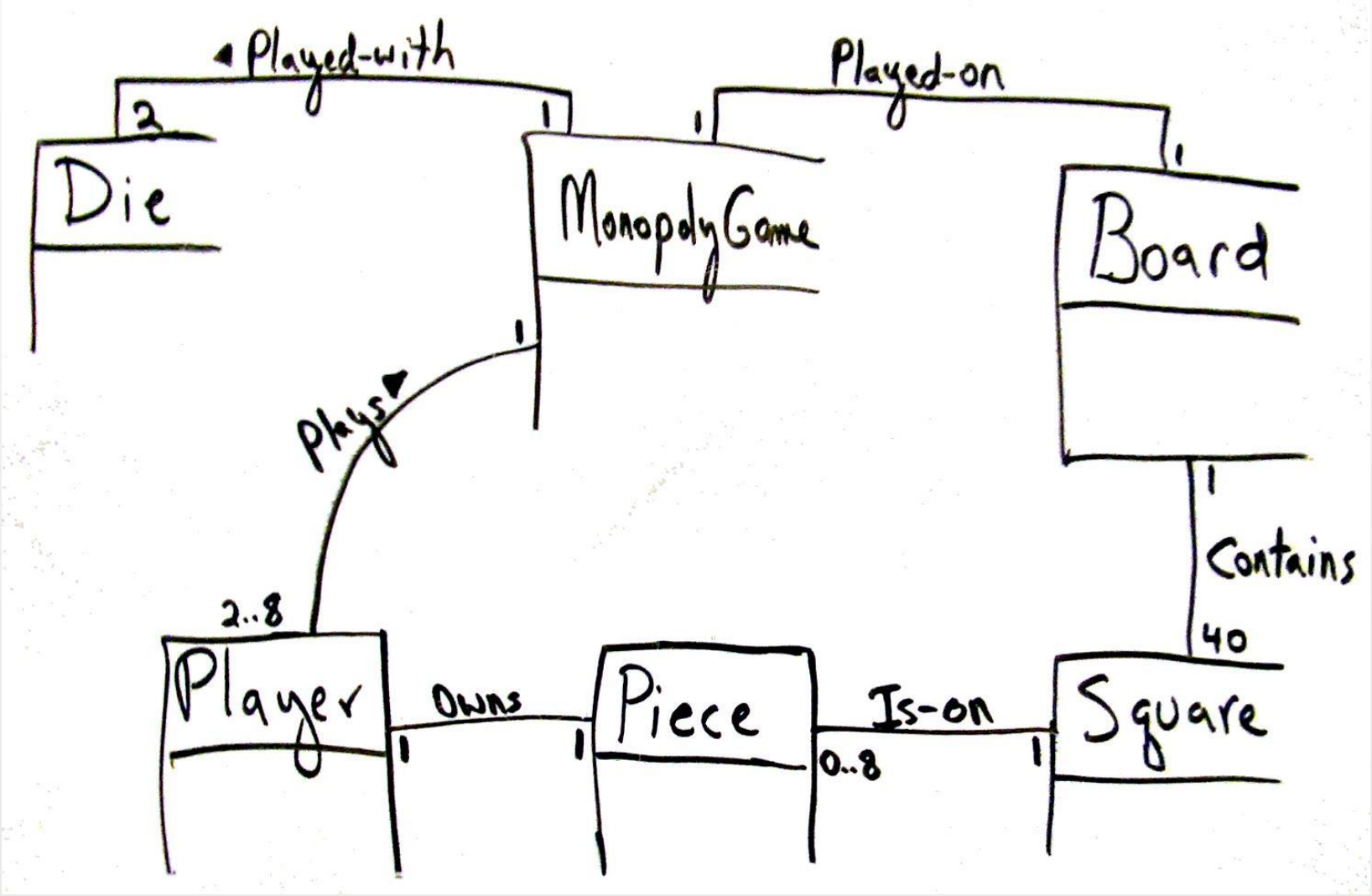
4.2 Domain Models

- Example: Associations in the Domain Models



4.2 Domain Models

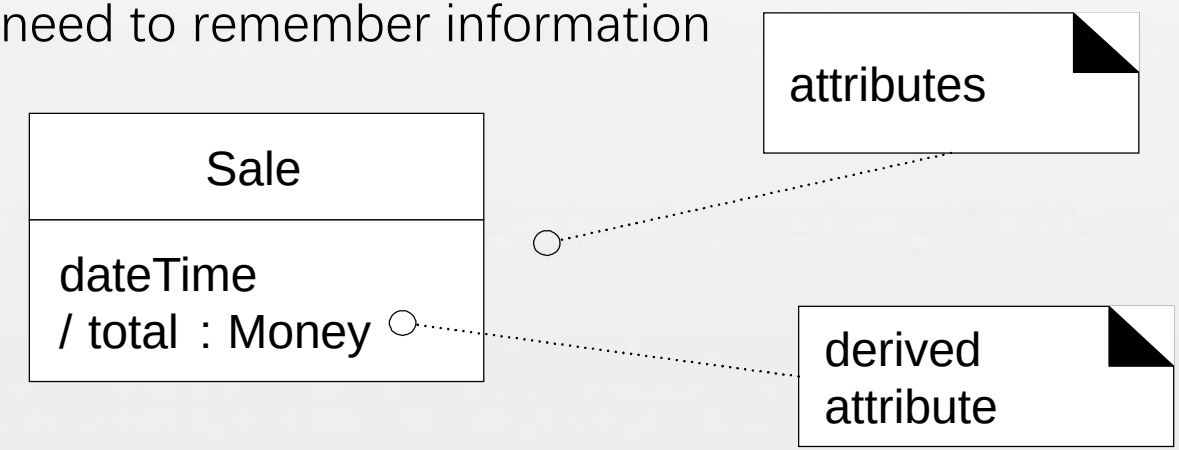
- **Example: Associations in the Domain Models**



4.2 Domain Models

Attributes

- **An attribute is a logical data value of an object**
- **When?**
  - Include attributes that the requirements (for example, use cases) suggest or imply a need to remember information
- **Attribute Notation**



  - visibility name : type multiplicity = default {property-string}

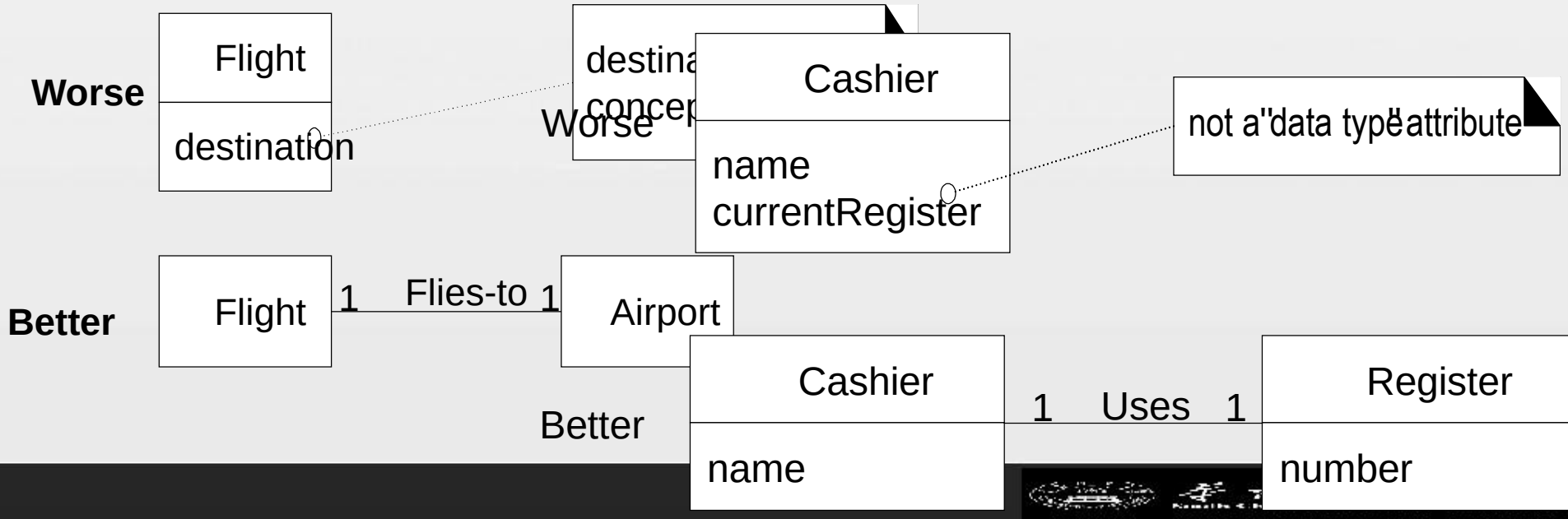
- **Derived Attributes**
  - attribute can be calculated or derived from other attribute or concept. use the UML convention: a / symbol before the attribute name

4.2 Domain Models

Attributes

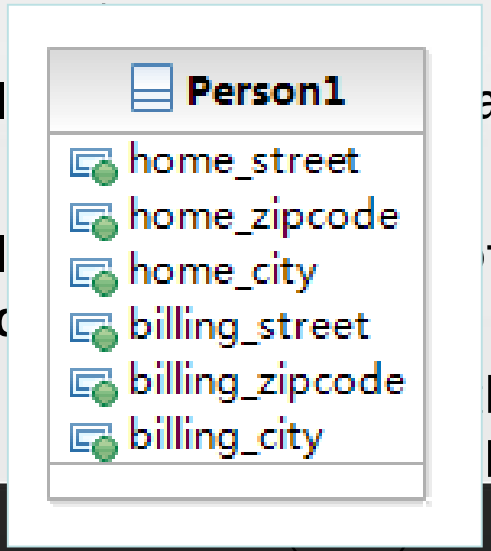
Guideline: What are Suitable Attribute Types?

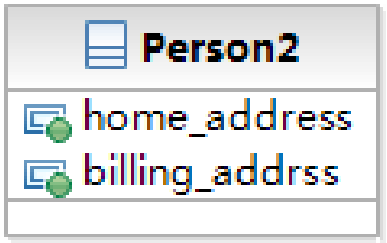
- Don't model a complex domain concept as an attribute. If there are relation between concepts, relate them with an association, not with an attribute.
- **Focus on Data Type Attributes in the Domain Model.** most attribute types should be "primitive" data types, such as numbers and boolean.

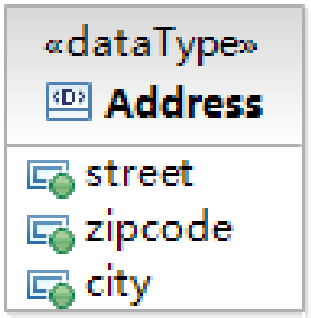


4.2 Domain Models

- **Attributes**
  - **When to Define New Data Type Classes**
    - **It is composed of separate sections.**
      - phone number, name of person
    - **There are operations associated with it, such as parsing or validation.**
      - social security number
    - **It has other attributes.**
      - promotional price could have a start (effective) date and end





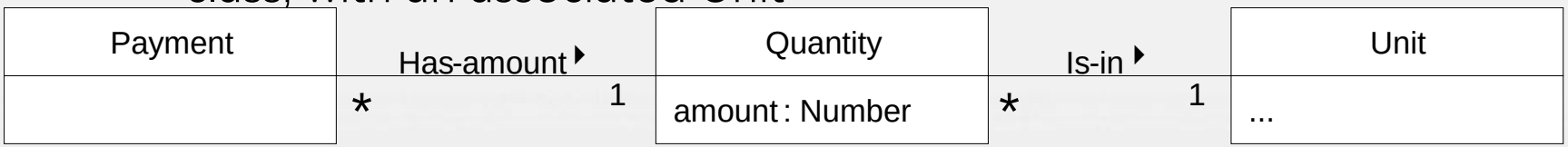


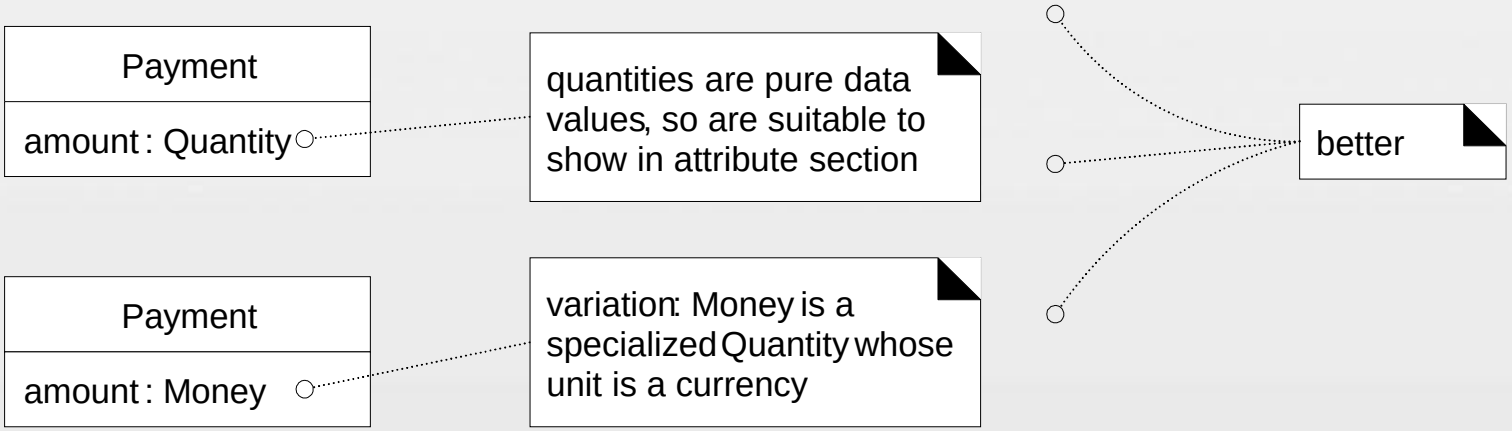
4.2 Domain Models

Attributes

Modeling Quantities and Units

- Most numeric quantities should not be represented as plain numbers, These are quantities with associated units
- In the general case, the solution is to represent Quantity as a distinct class, with an associated Unit

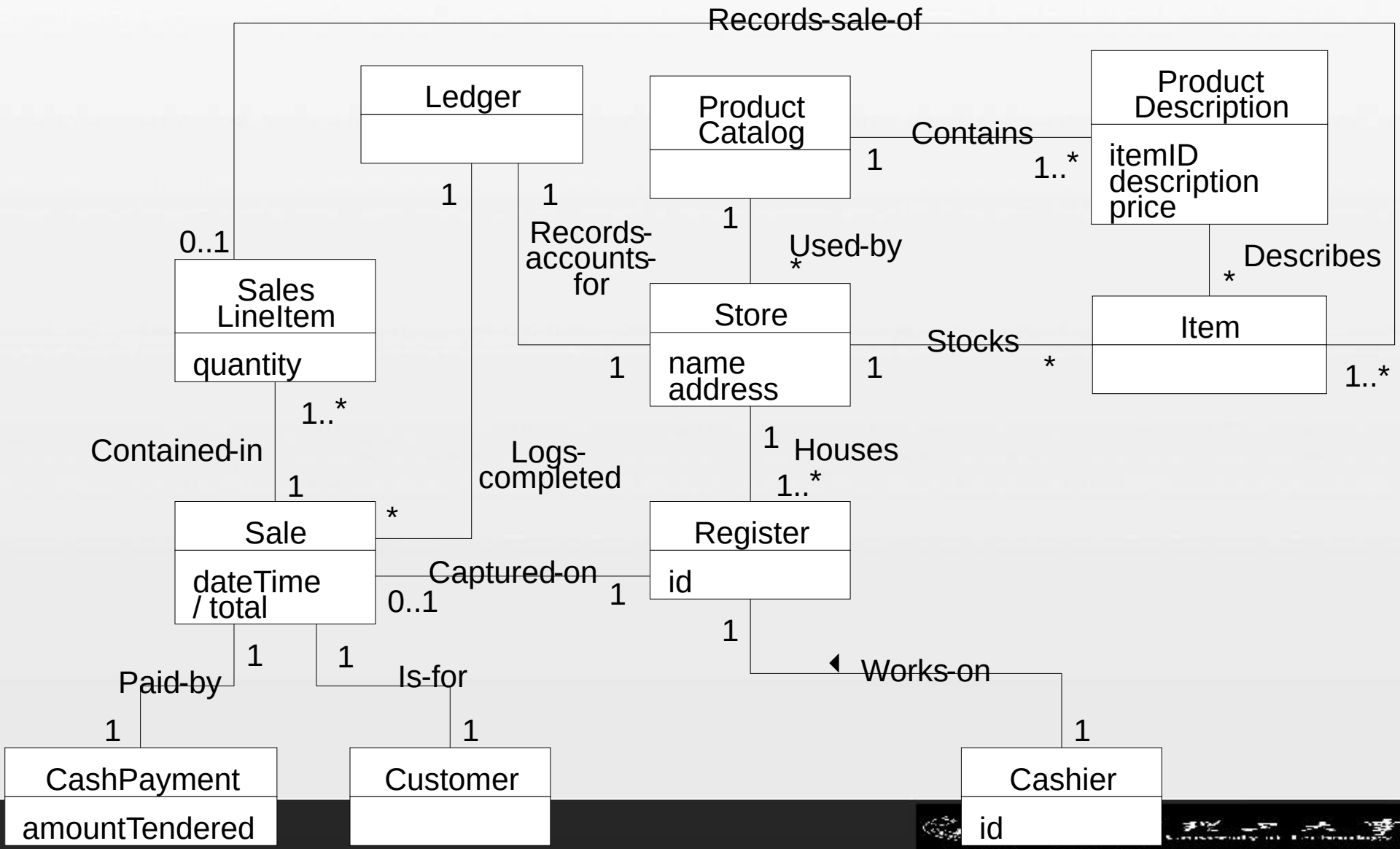




No Attributes Representing Foreign Keys. Once again, relate types with an association, not with an attribute.

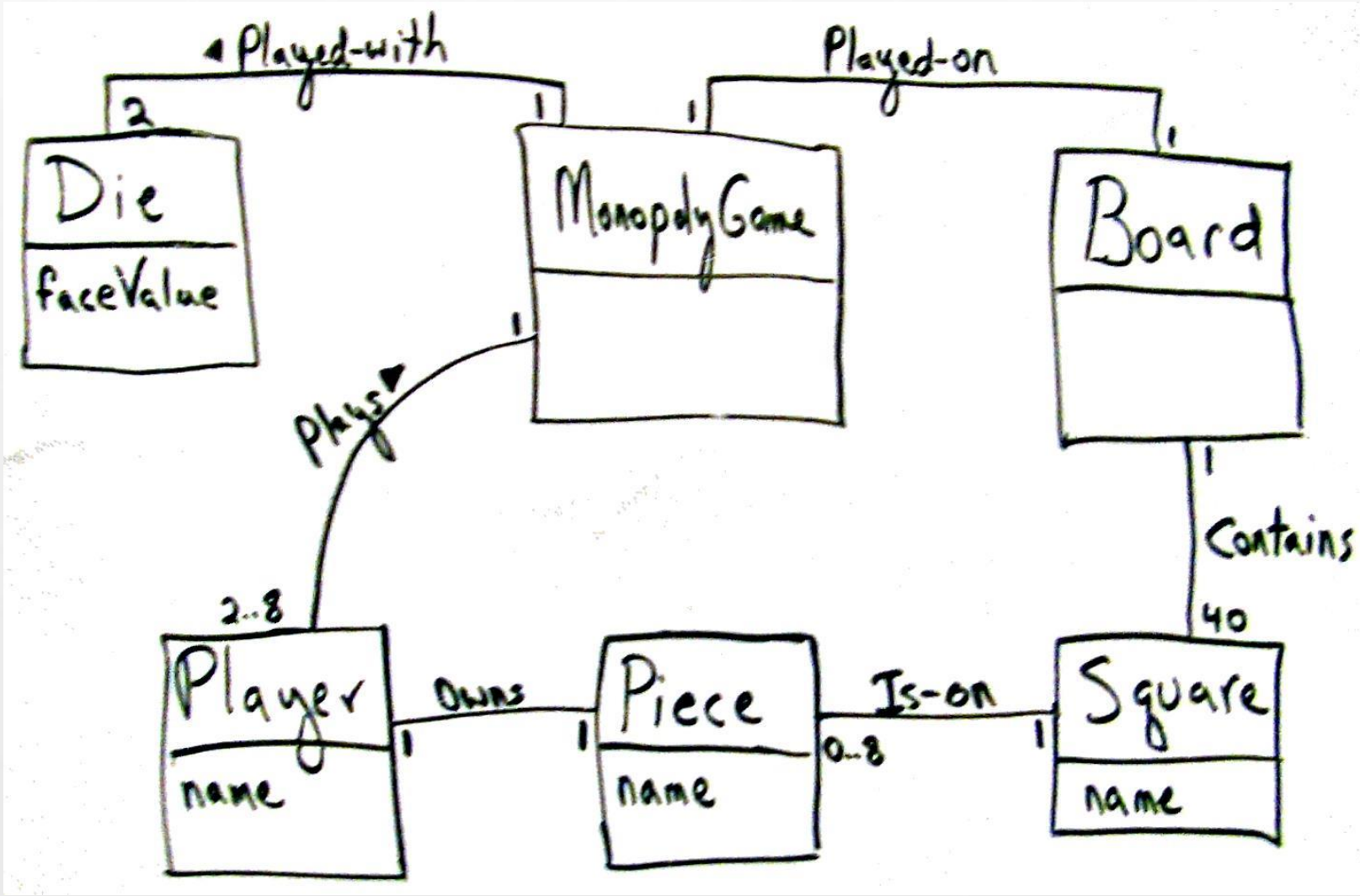
4.2 Domain Models

- **Example: Attributes in the Domain Models**



4.2 Domain Models

- Example: Attributes in the Domain Models



4.2 Domain Models

- Iterative and Evolutionary Domain Modeling
  - incrementally evolve a domain model over several iterations
  - Avoid a waterfall-mindset big-modeling effort to make a thorough or "correct" domain model

Table 9.4. Sample UP artifacts and timing. s - start; r - refine

Discipline	Artifact	Incep.	Elab.	Const.	Trans.
	Iteration→	I1	E1..En	C1..Cn	T1..T2
Business Modeling	***Domain Model***		s		
Requirements	Use-Case Model (SSDs)	s	r		
	Vision	s	r		
	Supplementary Specification	s	r		
	Glossary	s	r		
Design	Design Model		s	r	
	SW Architecture Document		s		
	Data Model		s	r	

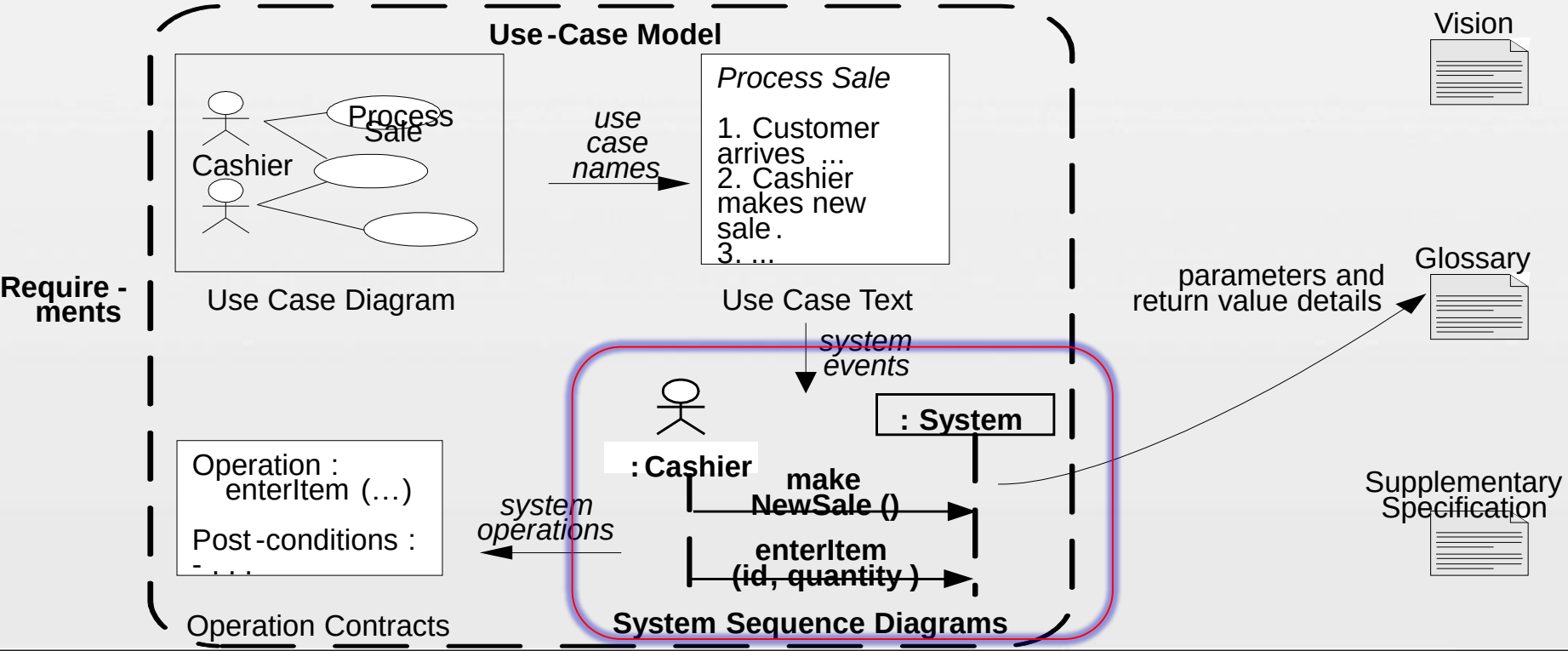
4.2 Domain Models

- Summary?
  - What is domain model?
  - Why domain model
  - How to create domain model
    - Identify concept classes
    - Identify associations
    - Identify attributes

4.3 System Sequence Diagrams

- The goal of System Sequence Diagrams
  - Identify system events.
  - Create system sequence diagrams for use case scenarios.

- What is looks like?



4.3 System Sequence Diagrams

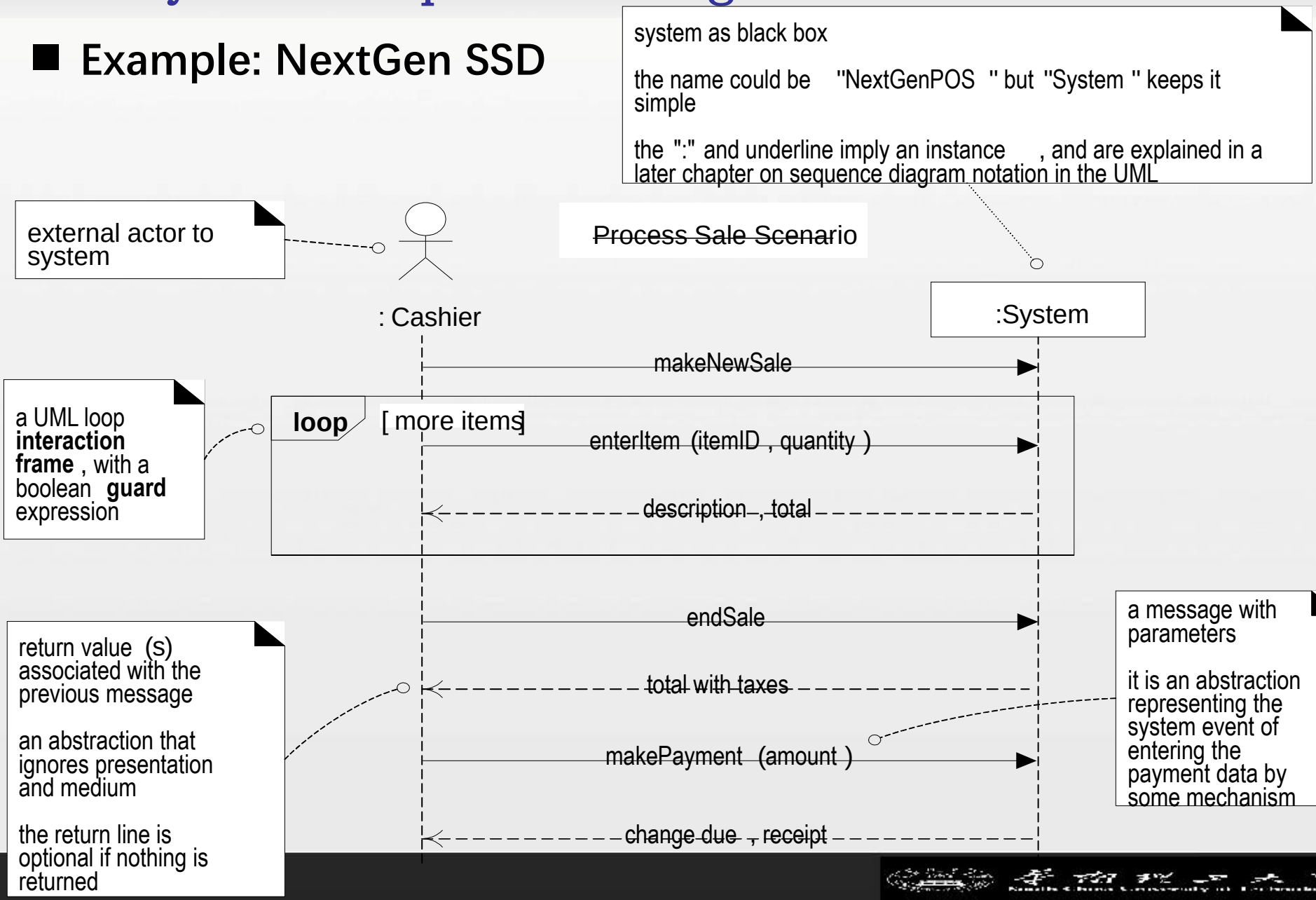
What are System Sequence Diagrams

- A system sequence diagram is a picture that shows, for one particular scenario of a use case, the events that external actors generate, their order, and inter-system events. All systems are treated as a black box; the emphasis of the diagram is events that cross the system boundary from actors to systems

- **Guideline**
  - **Draw an SSD for a main success scenario of each use case, and frequent or complex alternative scenarios.**

4.3 System Sequence Diagrams

- **Example: NextGen SSD**



4.3 System Sequence Diagrams

Applying UML: Sequence Diagrams

- The UML does not define something called a "system" sequence diagram but simply a "sequence diagram." The qualification is used to emphasize its application to systems as black boxes

Relationship Between SSDs and Use Cases?

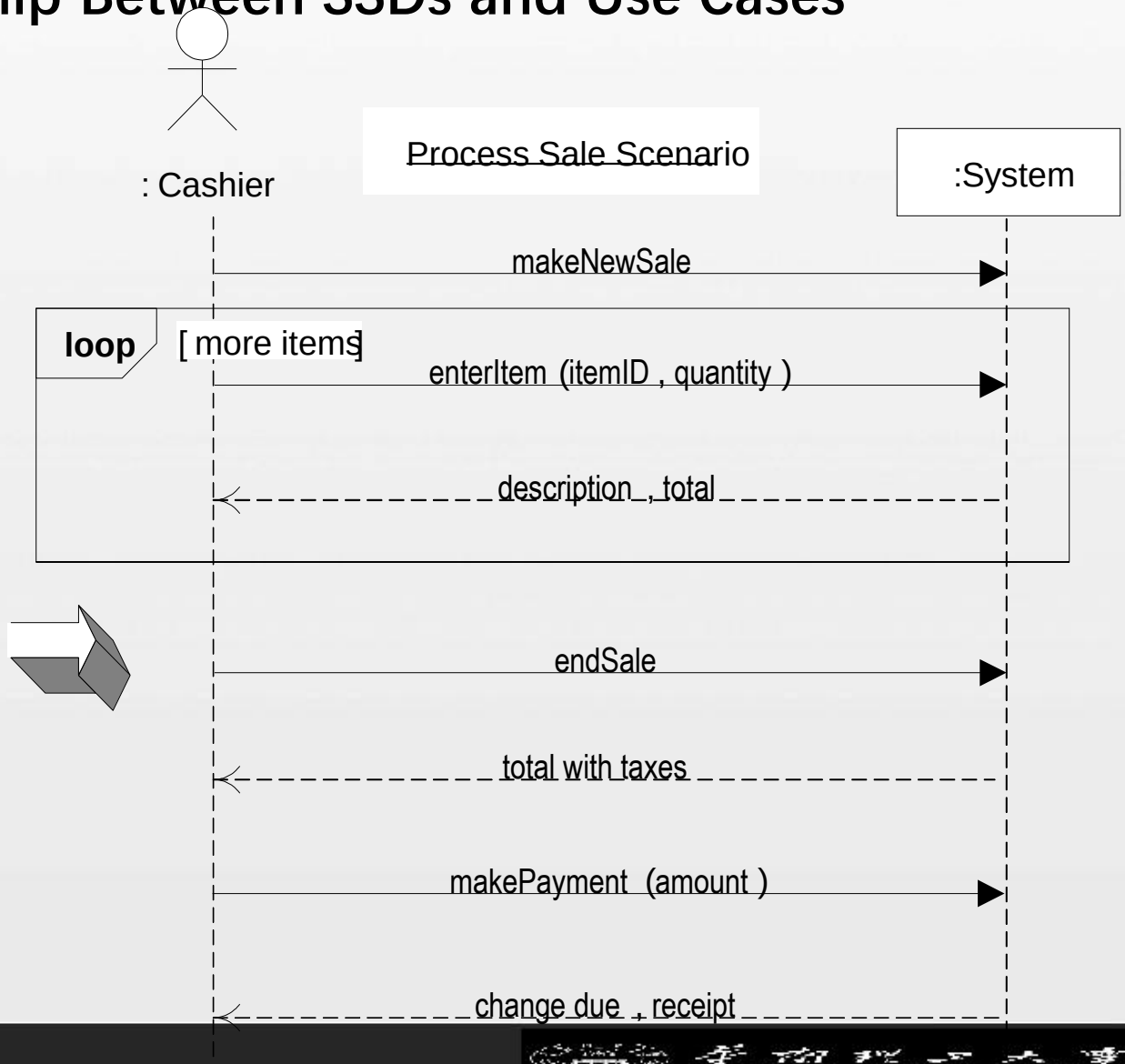
- SSD shows system events for one scenario of a use case, therefore it is generated from inspection of a use case
- SSDs can also be used to illustrate collaborations between systems, such as between the POS and the external credit payment authorizer

4.3 System Sequence Diagrams

- **Example: Relationship Between SSDs and Use Cases**

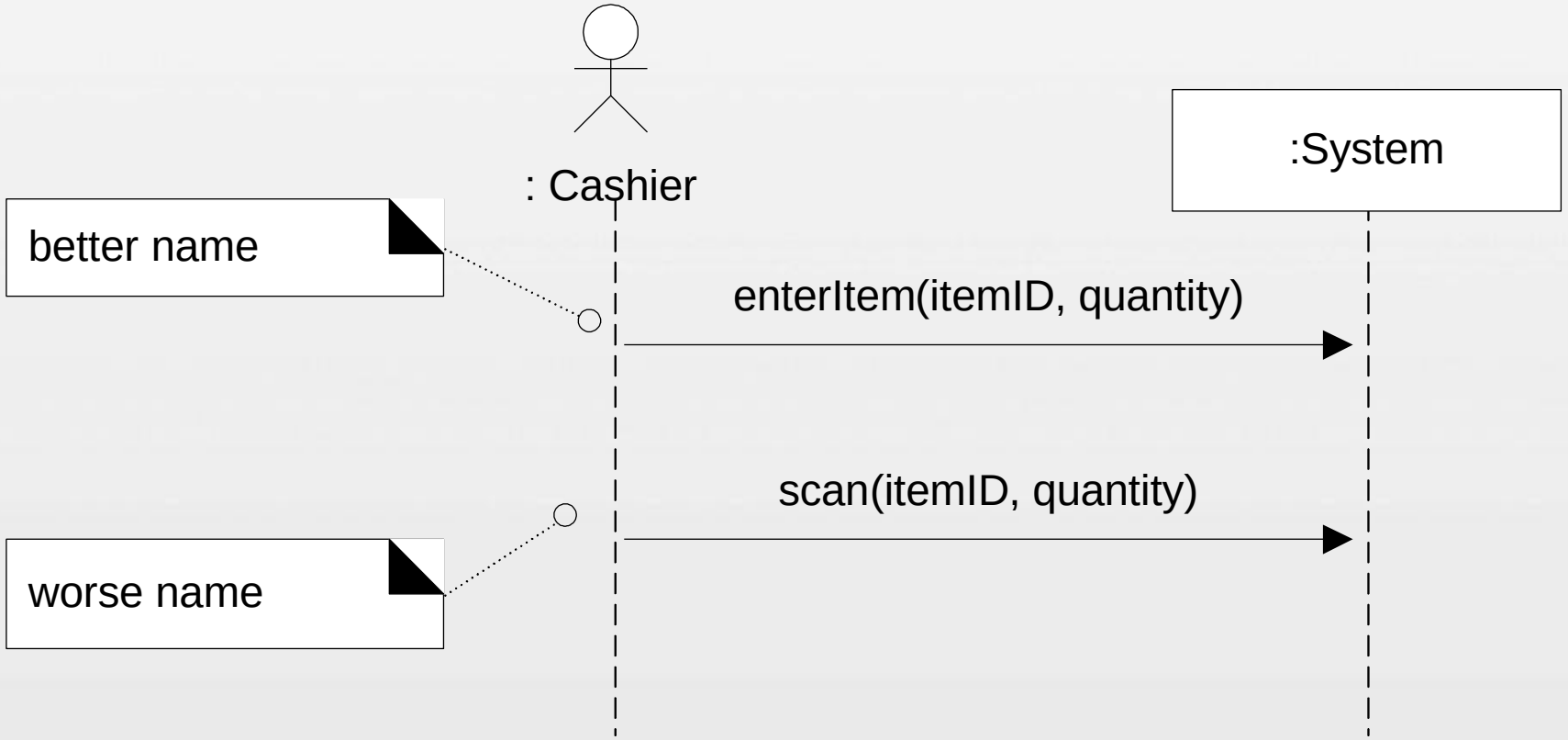
Simple cash -only Process Sale scenario :

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 .
...



4.3 System Sequence Diagrams

- **How to Name System Events and Operations?**
  - System events should be expressed at the abstract level of intention rather than in terms of the physical input device.



4.3 System Sequence Diagrams

What SSD Information to Place in the Glossary?

- The elements shown in SSDs (operation name, parameters, return data) are terse. These may need proper explanation so that during design it is clear what is coming in and going out. The Glossary is a great place for these details

- **Guideline**
  - **In general for many artifacts, show details in the Glossary.**

4.3 System Sequence Diagrams

- Example: Monopoly SSD