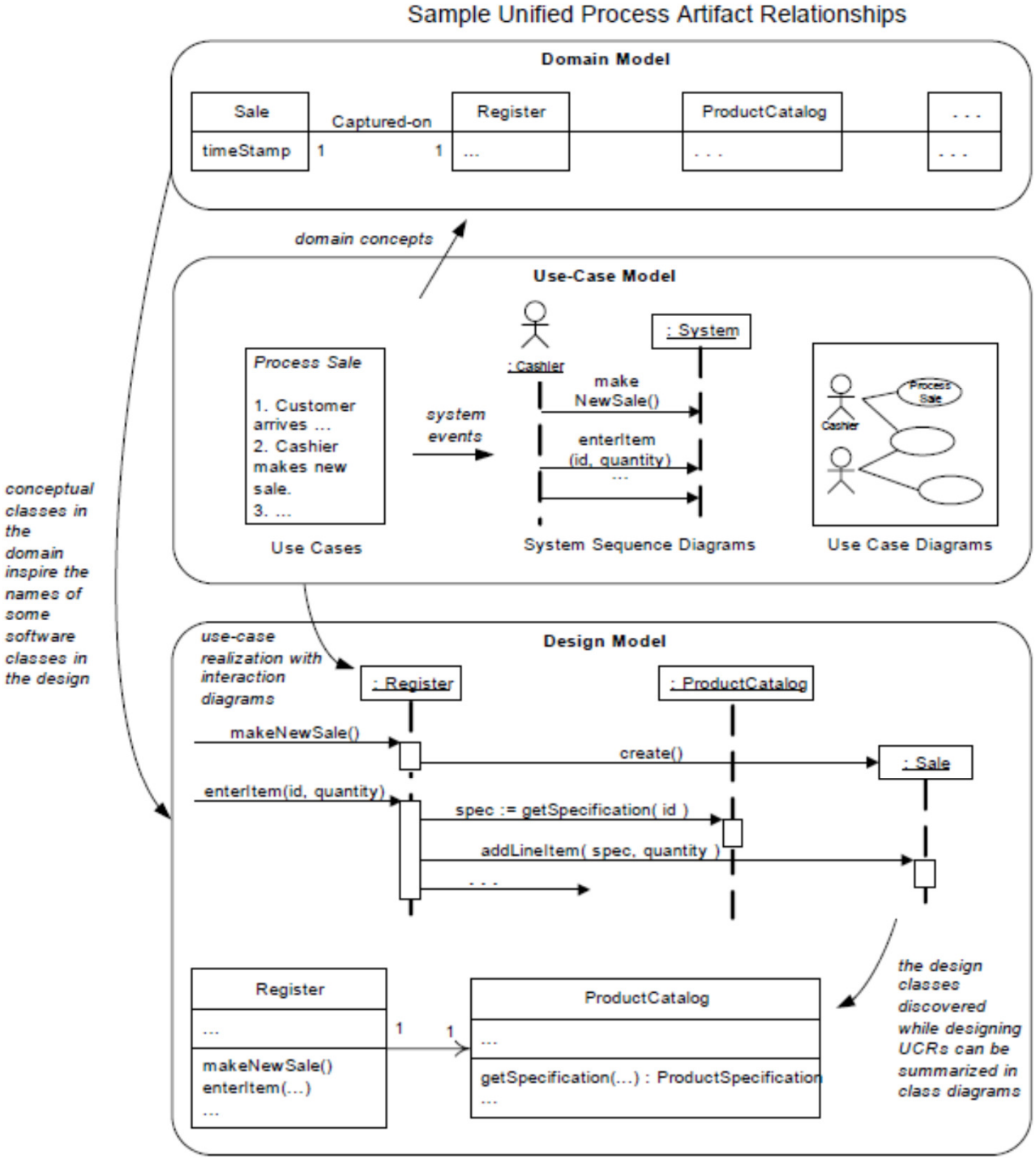
Larman Domain Model (Problem Domain Model)

- Overview of
  - domain model
  - how it relates to other UP diagrams
  - how it relates to other class diagrams

Sample Unified Process Artifacts and Timing (s-start; r-refine)

Discipline	Artifact Iteration^	Incep. 11	Elab. EL.En	Const. CL.Cn	Trans. T1..T2
Business Modeling	Domain Model		s		
Requirements	Use-Case Model	*s*	r		
	Vision	*a*	r		
	Supplementary Specification	*s*	r		
	Glossary	*s*	r		
Design	Design Model SW Architecture Document Data Model		s s s	r r	
Implementation	Implementation Model		s	r	r
Project Managemen	t SW Development Plan	*s*	r	r	r
Testing	Test Model		s	r	
Environment	Development Case	*s*	r		


Sample Unified Process Artifact Relationships
Domain Model
Sale
timeStamp
Captured-on
1
1
Register
...
ProductCatalog
. . .
. . .
. . .
domain concepts
Use-Case Model
Process Sale
1. Customer arrives ...
2. Cashier makes new sale.
3. ...
Use Cases
system events
: Cashier
: System
make NewSale()
enterItem (id, quantity)
...
System Sequence Diagrams
Process Sale
Cashier
Use Case Diagrams
conceptual classes in the domain inspire the names of some software classes in the design
use-case realization with interaction diagrams
Design Model
: Register
: ProductCatalog
: Sale
makeNewSale()
create()
enterItem(id, quantity)
spec := getSpecification(id)
addLineItem(spec, quantity)
. . .
Register
...
makeNewSale()
enterItem(...)
...
1
1
ProductCatalog
...
getSpecification(...) : ProductSpecification
...
the design classes discovered while designing UCRs can be summarized in class diagrams

Domain (conceptual) model

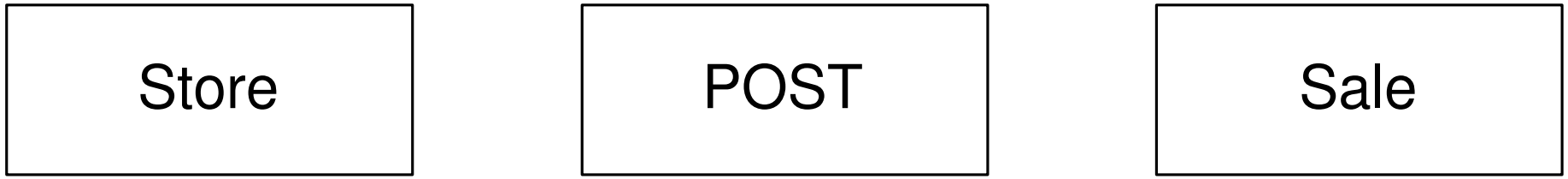
Following process as outlined by Larman

- analysis of the domain based on concepts (classes, objects) not functions
- purpose: to make a model that decomposes the problem and communicates the important domain terms and how they are related

Problem Domain

- conceptual model of the real world, not of the software
- no software artifacts such as GUI, database
- no responsibilities or methods

What is the information in the real world relevant to this problem; what are the relevant classes?


Store
POST
Sale

Conceptual model

- Concept is an idea, thing, or object
- concept
  - symbol: words or image to represent concept
  - intension: definition of what it is
  - extension: examples or external properties which define the concept
  - e.g. Sale; Sale represents a purchase transaction and has a date, time, item and amount; examples of sales,

Domain Classes

- Noun phrases in requirements, use cases
- Previous system
- Categories

Discipline	Artifact Iteration→	Incep. Il	Elab. E1..En	Const. C1..Cn	Trans. 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	
Implementation	Implementation Model		s	r	r
Project Management	SW Development Plan	s	r	r	r
Testing	Test Model		s	r	
Environment	Development Case	s	r		

Table 10.2 Sample UP artifacts and timing, s - start; r - refine

Identifying concepts

- Start with noun, noun phrases in the descriptions of the problem - in the requirements document and use cases
- concepts can be behavioural without an information storing role
- use names used in that domain
- do not add irrelevant items

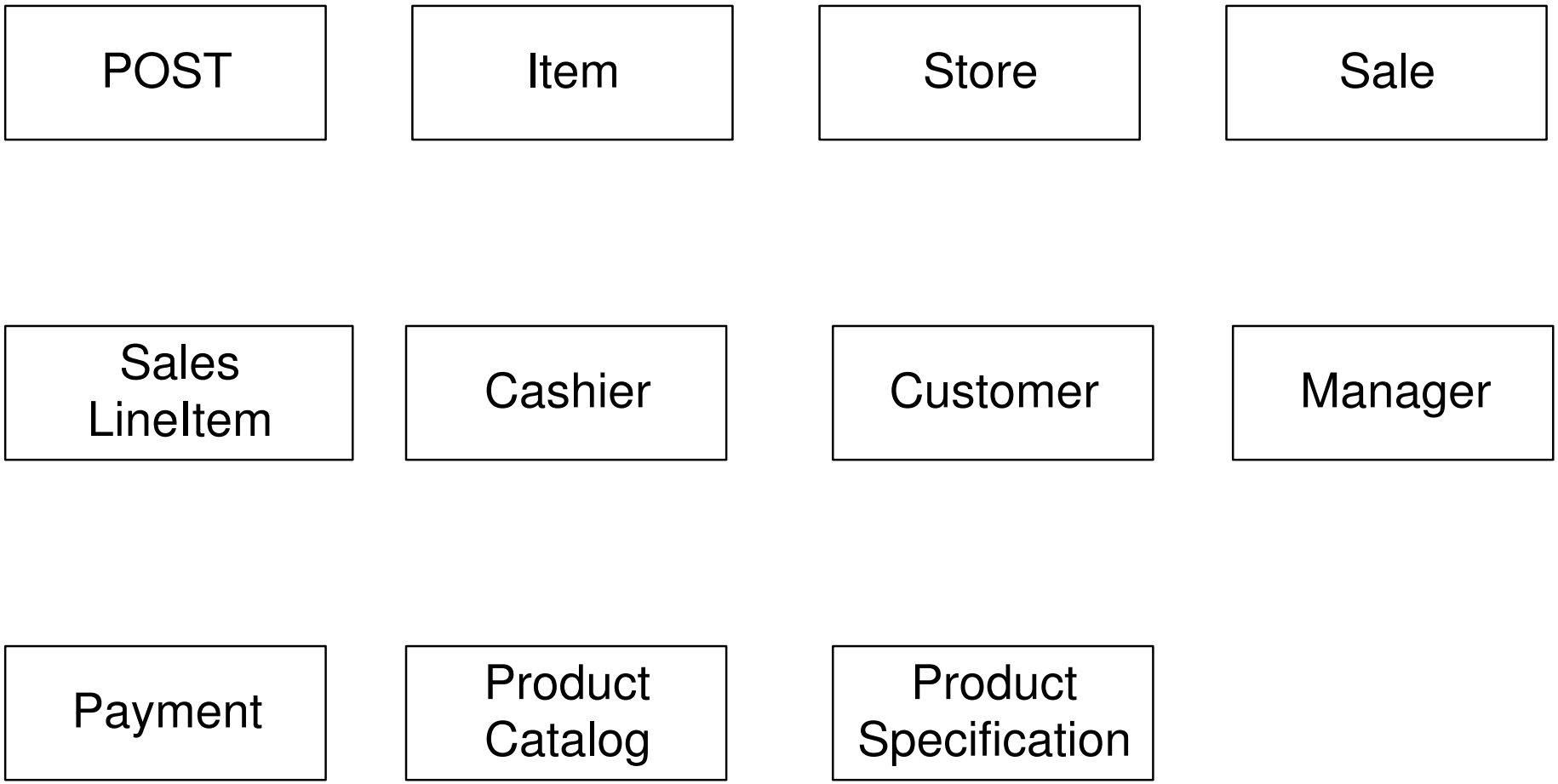
- Use Case: **Buy Items with Cash** (essential use case)
- Actors: Customer (Initiator)
- Purpose: Capture a sale and its cash payment.

Overview: A Customer arrives at a checkout with items to purchase. The cashier records the purchase items and collects a cash payment. On completion, the Customer leaves with the items and a receipt.

- Type: Primary and essential.
- Functions: R1.1, R1.2, R1.3, R1.7, R1.9, R2.1 (Cross References)

- **Typical Course of Events:**

Actor Action	**System Response**
1. This use case begins when a Customer arrives at a POST checkout with items to purchase.	
2. The Customer gives UPC to the system. If there is more than one item, the Customer can give the quantity as well.	3. Displays the price and running total.
4. The customer states there are no more items.	5. Displays the sale total.
6. The customer makes a payment.	7. Displays the balance due and prints a receipt.
8. The Customer leaves with the items purchased, change, and receipt.	


POST
Item
Store
Sale
Sales LineItem
Cashier
Customer
Manager
Payment
Product Catalog
Product Specification

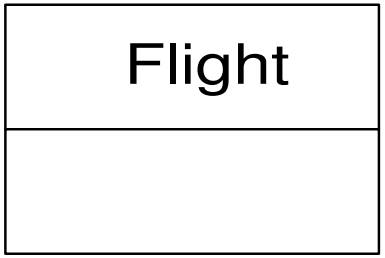
Guidelines

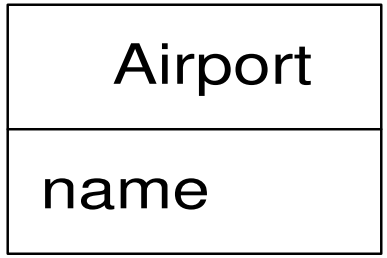
- Have more not less concepts
- if do not think of something as a number or text in the real world then probably a concept rather than attribute
- if in doubt, make a separate concept

In airline domain is destination an attribute of flight or a separate concept?


Flight
destination

or... ?

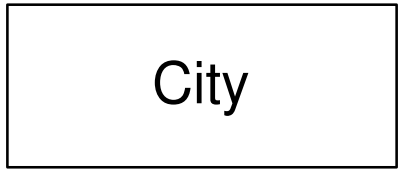

Flight


Airport
name


Flight
destination

or…


Flight


City

Specification/description concepts

- Needed when
  - deleting all instances loses information
  - should reduce replicated/redundant info
- E.g. product specification concept, recording all information about a product rather than having info in each instance

Making a Domain Model

- Identify concepts
  - noun phrases in requirements documents
  - using the Concept Category List of Larman
- Draw a domain/concept diagram (simplified class diagram)
- Add associations to record relationships
- Add attributes for information required

Conceptual Class Category	Examples
physical or tangible objects	*Register* *Airplane*
specifications, designs, or descriptions of things	*ProductSpecification* *FlightDescription*
places	*Store* *Airport*
transactions	*Sale, Payment* *Reservation*
transaction line items	*SalesLineItem*
roles of people	*Cashier* *Pilot*
containers of other things	*Store, Bin* *Airplane*
things in a container	*Item* *Passenger*
other computer or electro-mechanical systems external to the system	*CreditPaymentAuthorizationSystem* *AirTrafficControl*
abstract noun concepts	*Hunger* *Acrophobia*
organizations	*SalesDepartment* *ObjectAirline*
events	*Sale, Payment, Meeting* *Flight, Crash, Landing*
processes (often *not* represented as a concept, but may be)	*SellingAProduct* *BookingASeat*
rules and policies	*RefundPolicy* *CancellationPolicy*
catalogs	*ProductCatalog* *PartsCatalog*

Conceptual Class Category	Examples
records of finance, work, contracts, legal matters	*Receipt, Ledger, EmploymentContract* *MaintenanceLog*
financial instruments and services	*LineOfCredit* *Stock*
manuals, documents, reference papers, books	*DailyPriceChangeList* *RepairManual*

Association

- Association:
  - structural relationship between objects (classes) of different type
  - records some meaningful/interesting relationship that we should remember (need-to-know association)

Association Guidelines

- start with need-to-know associations
- add others necessary for comprehension
- use Common Association List of Larman
- avoid redundant/derivable associations
- concepts more important, avoid too many associations

Finding associations

High-Priority Associations

- A is a *physical or logical part of B.*
- A is *physically or logically contained in/on B.*
- A is *recorded in B.*

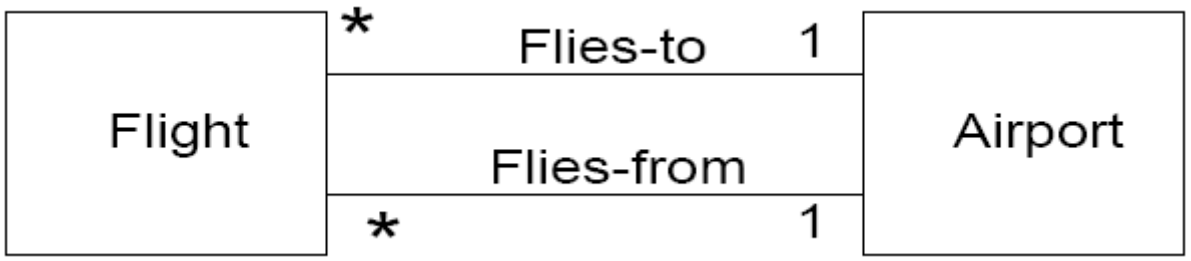
Use Larman's Association List (following slide)

Category	Examples
A is a physical part of B	*Drawer — Register (or more specifically, a POST)* *Wing — Airplane*
A is a logical part of B	*SalesLineItem — Sale* *FlightLeg—FlightRoute*
A is physically contained in/on B A is logically contained in B	*Register — Store, Item — Shelf* *Passenger — Airplane* *ItemDescription — Catalog* *Flight— FlightSchedule*
A is a description for B	*ItemDescription — Item* *FlightDescription — Flight*
A is a line item of a transaction or report B A is known/logged/recorded/reported/captured in B	*SalesLineItem — Sale* *Maintenance Job — Maintenance-Log* *Sale — Register* *Reservation — FlightManifest*
A is a member of B	*Cashier — Store* *Pilot — Airline*
A is an organizational subunit of B A uses or manages B	*Department — Store* *Maintenance — Airline* *Cashier — Register* *Pilot — Airplane*
A communicates with B	*Customer — Cashier* *Reservation Agent — Passenger*
A is related to a transaction B A is a transaction related to another transaction B	*Customer — Payment* *Passenger — Ticket* *Payment — Sale* *Reservation — Cancellation*
A is next to B	*SalesLineItem — SalesLineItem* *City— City*
A is owned by B A is an event related to B	*Register — Store* *Plane — Airline* *Sale — Customer, Sale — Store* *Departure — Flight*

Multiple associations

Can sometimes be a need to model multiple associations

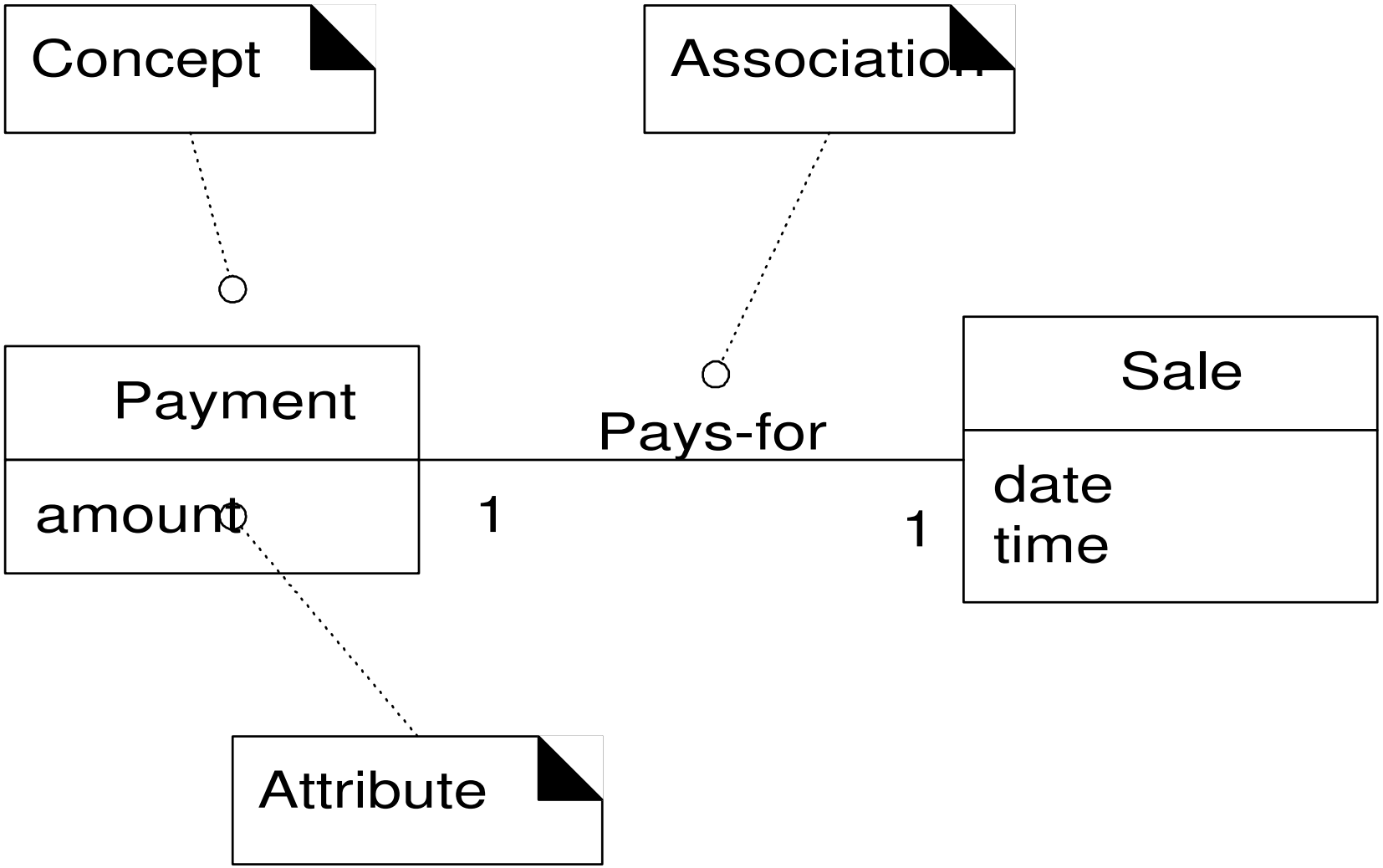
For example in the airline application, may need to have in the UML Domain Model:



Representing associations

- usual
  - line between concepts (classes)
  - name: a verb phrase
  - multiplicity at each end of line
  - convention to read concept verb-phrase concept from top to bottom and left to right, arrow to indicate direction if needed

alternative: name role at each end rather than assoc.


Concept
Association
Payment
amount
Pays-for
1
1
Sale
date
time
Attribute

Attributes

- A logical data value of an object
- attributes in the conceptual model
  - what the requirements suggest is information to be remembered, e.g. time of a transaction

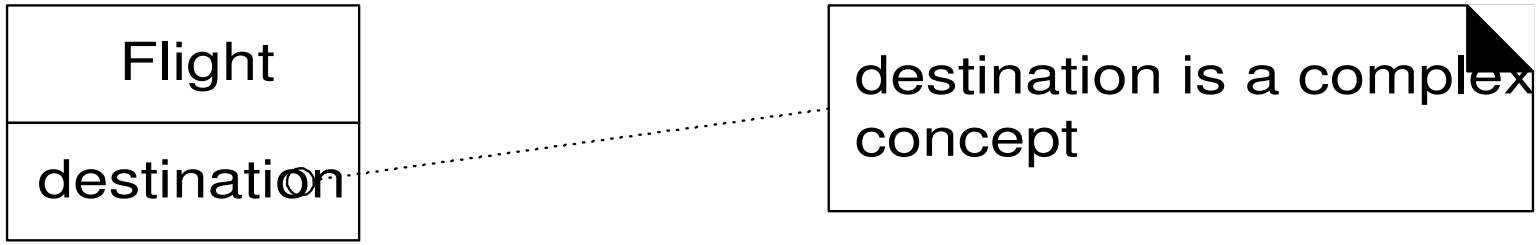
Attribute Guidelines

- Keep attributes simple
- most attributes are primitive data types, e.g. string, integer, …
- complex attribute
  - suggests a separate concept and association
- simple data types (pure data values) when not meaningful to distinguish two instances with same value; probably attribute but may be a concept

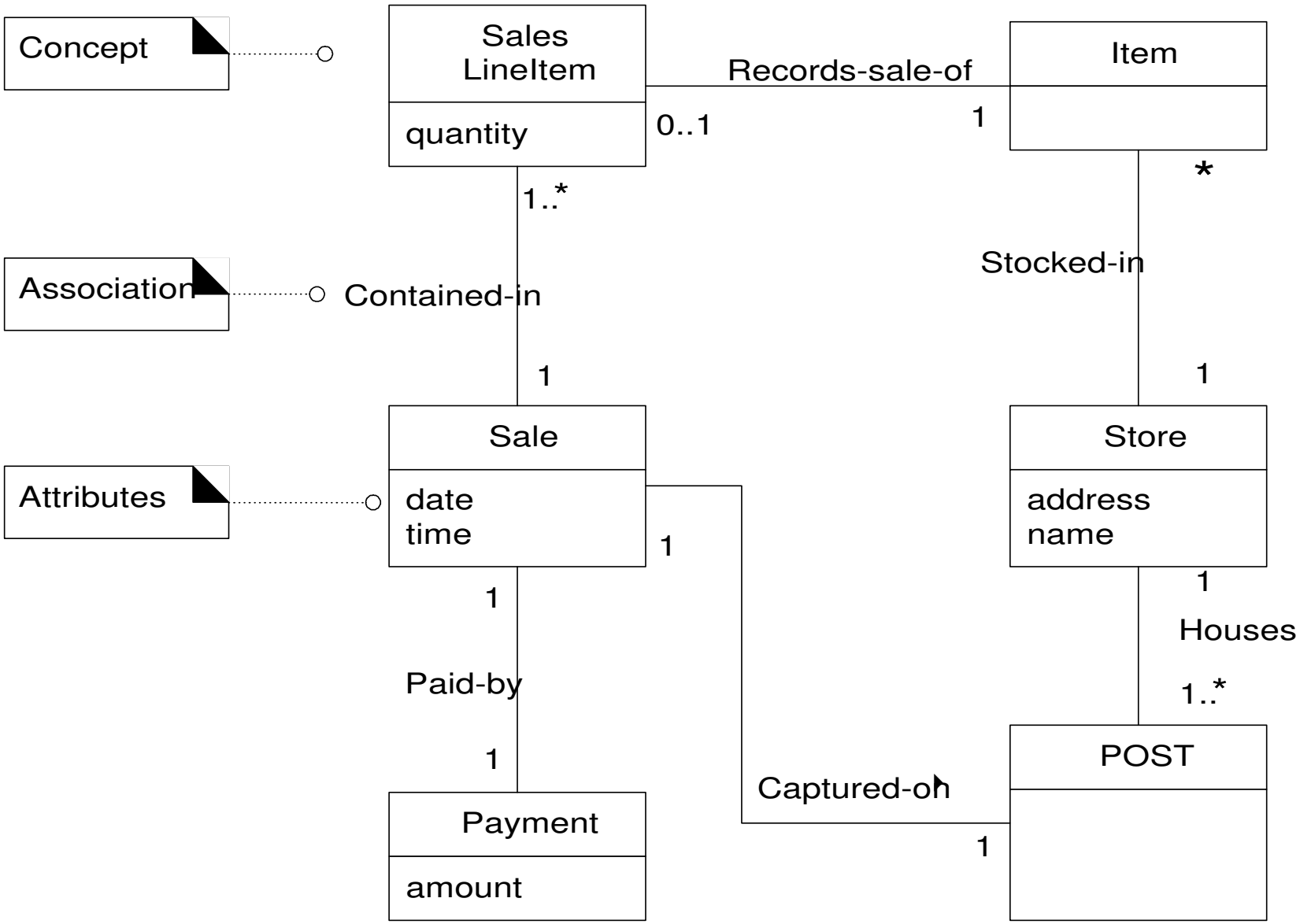
Attribute Guidelines ...

- If in doubt, introduce concept
- attributes are of the conceptual model, not the code; associations in conceptual model may be implemented using pointer attributes in the code
- beware pointer keys as attributes; should be associations

Worse



Better


Concept
Association
Attributes
Sales LineItem
quantity
Records-sale-of
0..1
1
Item
*
1..*
Contained-in
Stocked-in
1
1
Sale
date
time
Store
address
name
1
1
1
Houses
1..*
Paid-by
1
1
Payment
amount
Captured-on
1
POST

Non-primitive types

- Attribute may need to be a non-primitive type if
  - it has separate sections, e.g. name
  - it has associated operations such as parsing or validation, e.g. account number
  - it has other attributes, e.g. promotional price may have a start and end date
  - it has units, e.g. price, throughput

Non-primitive attribute types

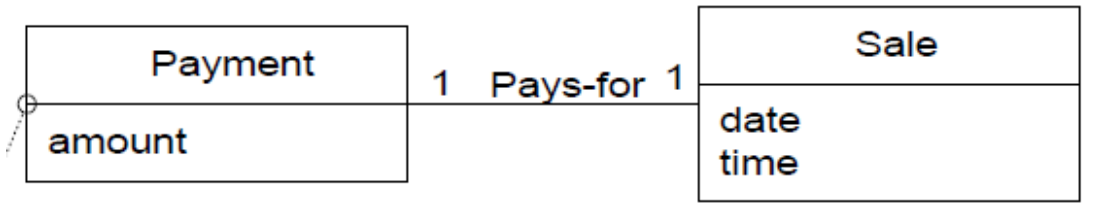
- Non-primitive types even though a pure data item may need to be separate concept

Domain/Conceptual Model

- Initial analysis
  - **The problem domain NOT** the solution
- Later – moving on to design, the solution, influenced by the conceptual model
  - full class diagram
  - responsibilities
  - collaboration
  - operations/methods
  - etc

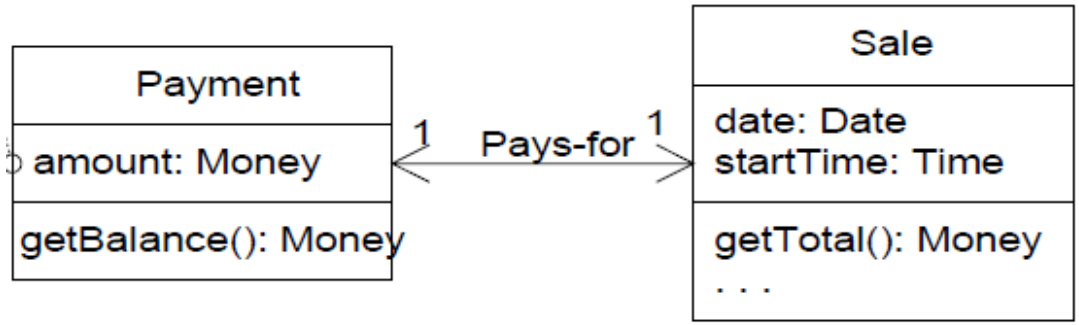
UP Domain Model

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



inspires objects and names in

al

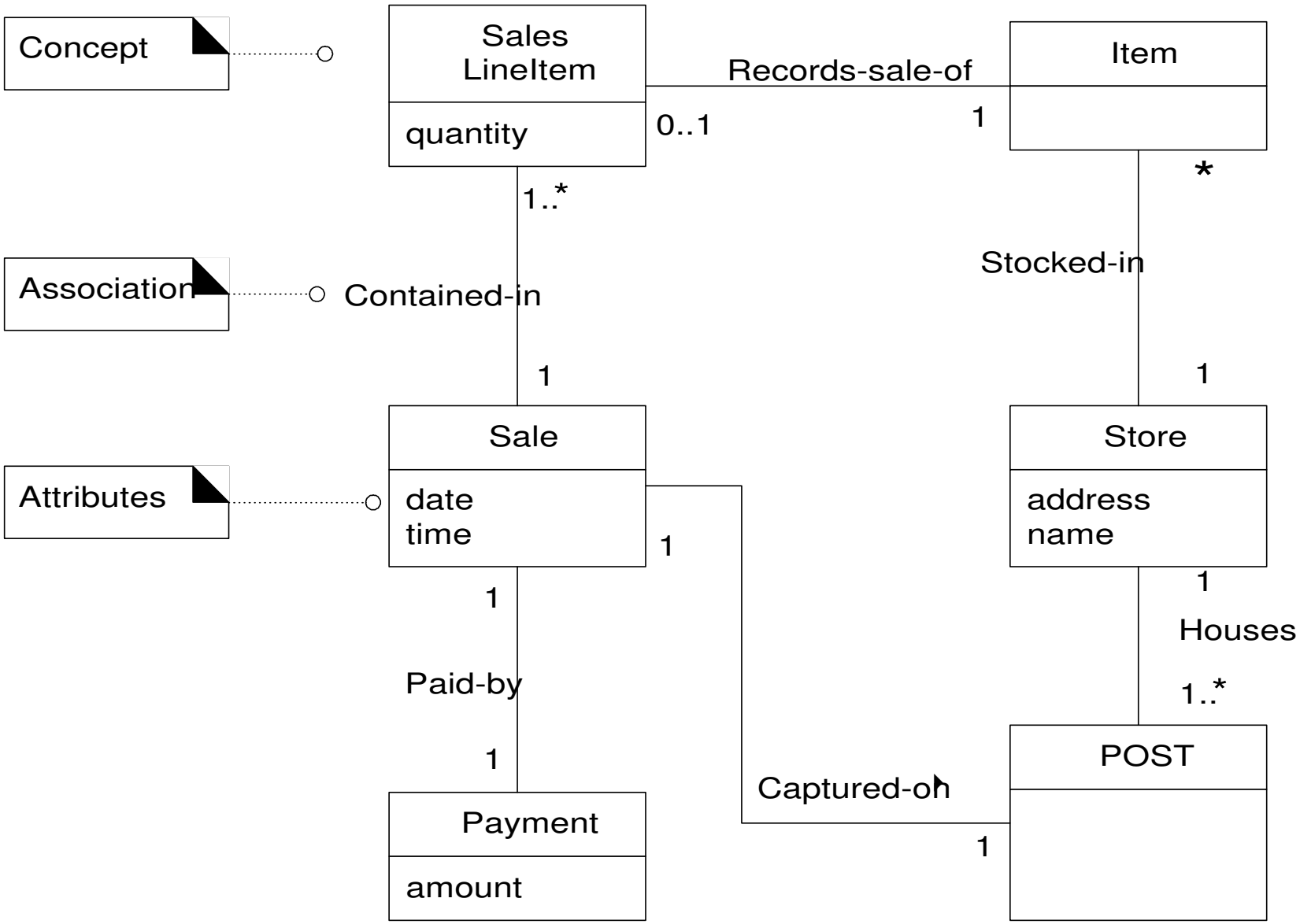


UP Design Model

Larman: class-related terminology

To keep things clear, this book will use class-related terms as follows, which is consistent with the UML and the UP:

- Conceptual class — real-world concept or thing. A conceptual or essential perspective. The UP Domain Model contains conceptual classes.
- Software class — a class representing a specification or implementation perspective of a software component, regardless of the process or method.
- Design class — a member of the UP Design Model. It is a synonym for software class, but for some reason I wish to emphasize that it is a class in the Design Model. The UP allows a design class to be either a specification or implementation perspective, as desired by the modeler.
- Implementation class — a class implemented in an object-oriented language such as Java.
- Class — as in the UML, the general term representing either a real-world thing (a conceptual class) or software thing (a software class).


Concept
Association
Attributes
Sales LineItem
quantity
Records-sale-of
0..1
1
Item
*
1..*
Contained-in
Stocked-in
1
1
Sale
date
time
Store
address
name
1
1
1
Houses
1..*
Paid-by
1
1
Payment
amount
Captured-on
1
POST