Software Analysis and Architecture

Conceptual Models: Domain Models

Sajid Anwer

Department of Software Engineering,
FAST-NUCES, CFD Campus

Lecture Material

- System Analysis and Design in a Changing World (Chapter 4)

Lecture Objectives

- Fundamentals of Domain Models

- Identify Entities / Associations / Attributes

Fundamentals of Domain Models

- It is a conceptual **model** of all the topics related to a specific problem.
- It describes the various *entities (things)*, their *attributes(characteristics)*, roles, and relationships, plus the constraints that govern the problem **domain**.
- Illustrates meaningful *conceptual classes* in a problem domain.
- It is a representation of *real-world concepts*, not software components.
- It may show:
  - concepts
  - associations between concepts
  - attributes of concepts
- It does not describe *solutions* to the problem that's why not include *responsibilities* of entities.

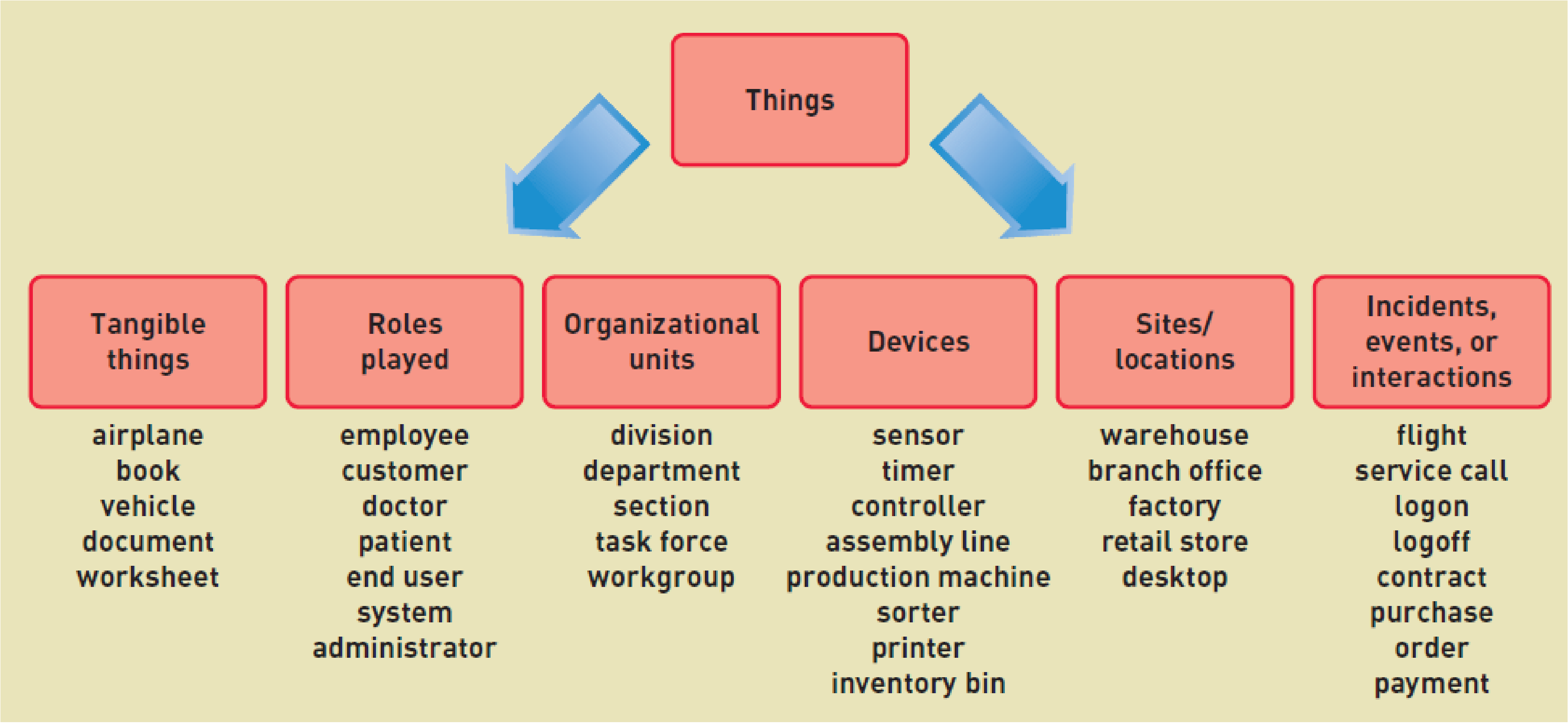
Fundamentals of Domain Models

- Steps to design domain model
  - » Identify Entities (Candidate Conceptual classes)
  - » Draw them in a Domain Model
  - » Add associations necessary to record the relationships that must be retained
  - » Add attributes necessary for information to be preserved

Domain Models – Identify Entities

- Brainstorming
- Noun / noun phrases
- Brainstorming
  - » Identify a user and a set of use cases or user stories.
  - » Brainstorm with the user to identify things involved when carrying out the use case—that is, things about which information should be *captured* by the system.
  - » Use the types of things (categories) to systematically ask questions about potential things, such as the following:
    - Are there any *tangible* things you store information about?
    - Are there any *locations* involved?
    - Are there *roles* played by people that you need to remember?
  - » Continue to work with all types of users and stakeholders to expand the brainstorming list.
  - » Merge the results, eliminate any duplicates, and compile an initial list.

Domain Models – Identify Entities



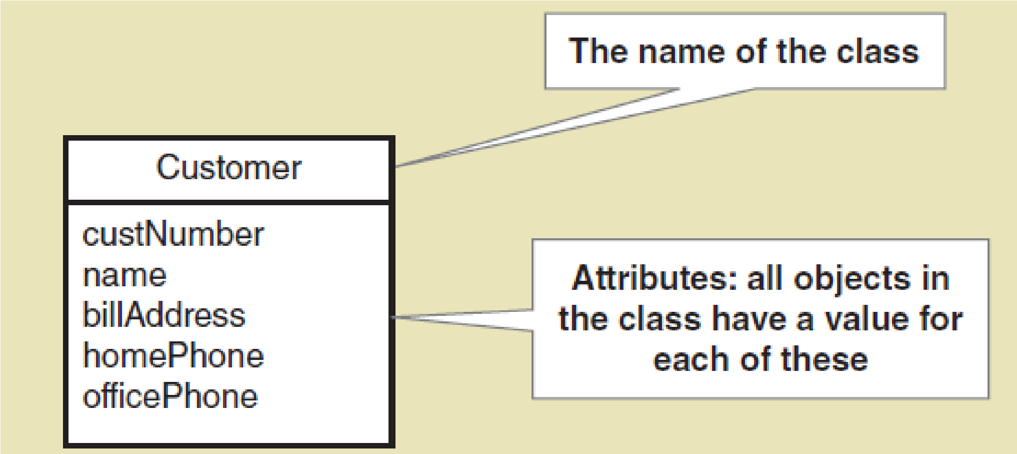
Things

Tangible things	Roles played	Organizational units	Devices	Sites/ locations	Incidents, events, or interactions
airplane book vehicle document worksheet	employee customer doctor patient end user system administrator	division department section task force workgroup	sensor timer controller assembly line production machine sorter printer inventory bin	warehouse branch office factory retail store desktop	flight service call logon logoff contract purchase order payment

Domain Models – Identify Entities

- Identify *Nouns* and *Noun Phrases* in textual descriptions of the domain.
- Fully dressed/ *expended* Use Cases are good for this type of linguistic analysis.
- It’s not strictly a mechanical process:
  - » Words may be ambiguous
  - » Different phrases may represent the same concepts.

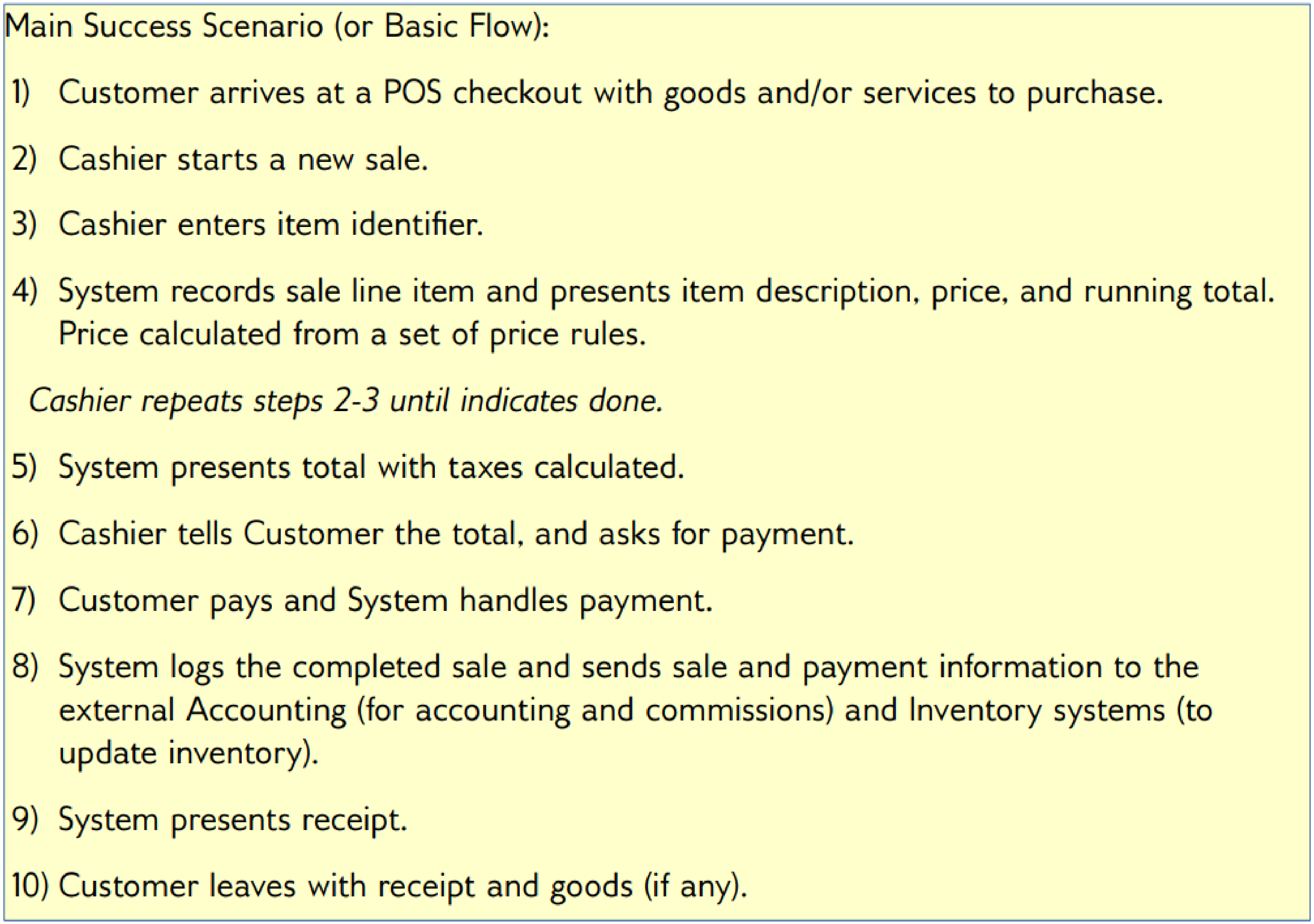
Domain Models – Identify Entities



- *class* a category or classification of a set of objects or things

- *domain classes are* classes that describes objects from the problem domain

Domain Models – Identify Entities



Main Success Scenario (or Basic Flow):

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. Price calculated from a set of price rules.

Cashier repeats steps 2-3 until indicates done.

5) System presents total with taxes calculated.

6) Cashier tells Customer the total, and asks for payment.

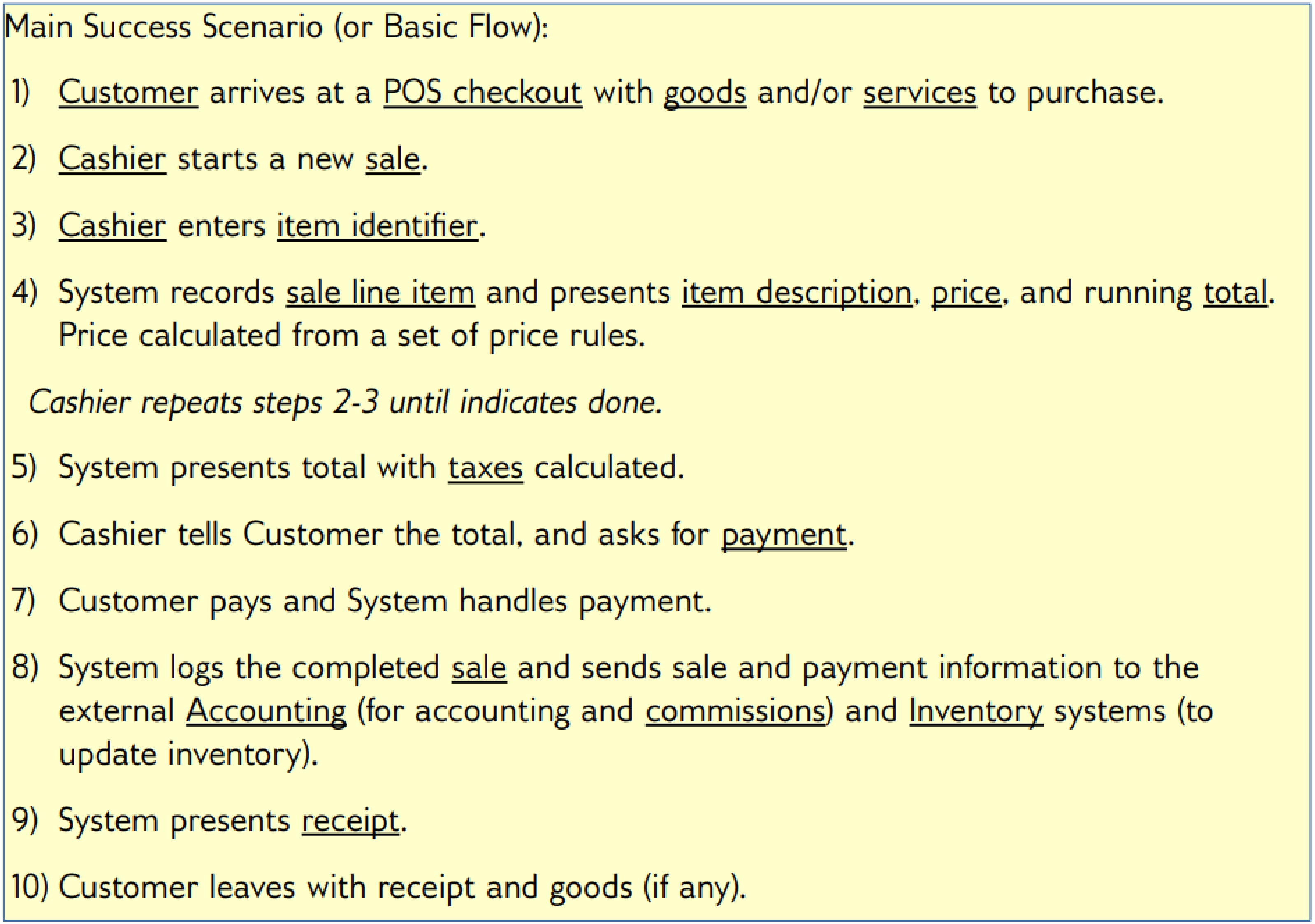
7) Customer pays and System handles payment.

8) System logs the completed sale and sends sale and payment information to the external Accounting (for accounting and commissions) and Inventory systems (to update inventory).

9) System presents receipt.

10) Customer leaves with receipt and goods (if any).

Domain Models – Identify Entities



Main Success Scenario (or Basic Flow):

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. Price calculated from a set of price rules.

Cashier repeats steps 2-3 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.

8) System logs the completed sale and sends sale and payment information to the external Accounting (for accounting and commissions) and Inventory systems (to update inventory).

9) System presents receipt.

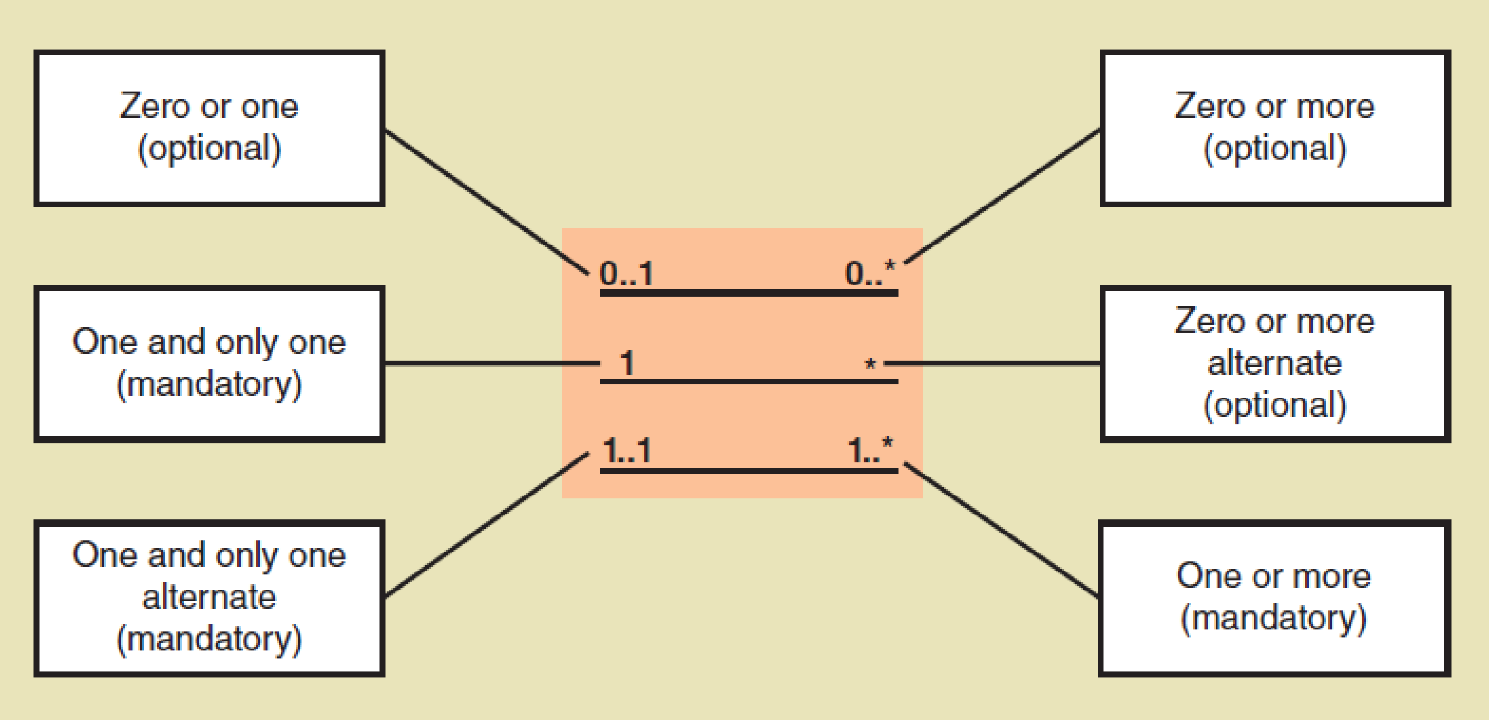
10) Customer leaves with receipt and goods (if any).

Domain Models – Identify Associations

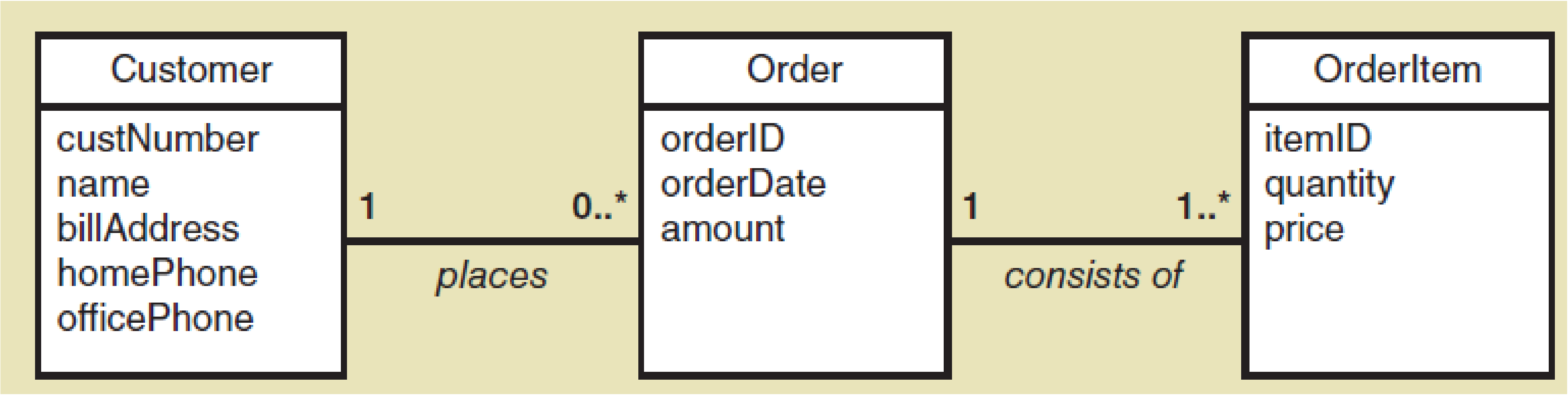
- Identify *associations* of conceptual classes needed to satisfy the information requirements of *current scenarios*.
- An association is a relationship between *instances of types* that indicates some *meaningful* and interesting connection.
- Also identify the association name to aid in comprehending the domain model.
- An association is represented as a line between classes with an association name.
- Associations are *inherently* bidirectional.
- Optional reading direction arrow is only an aid to the reader of the diagram.

Domain Models – Identify Associations

- Association Multiplicity



Domain Models – Identify Associations

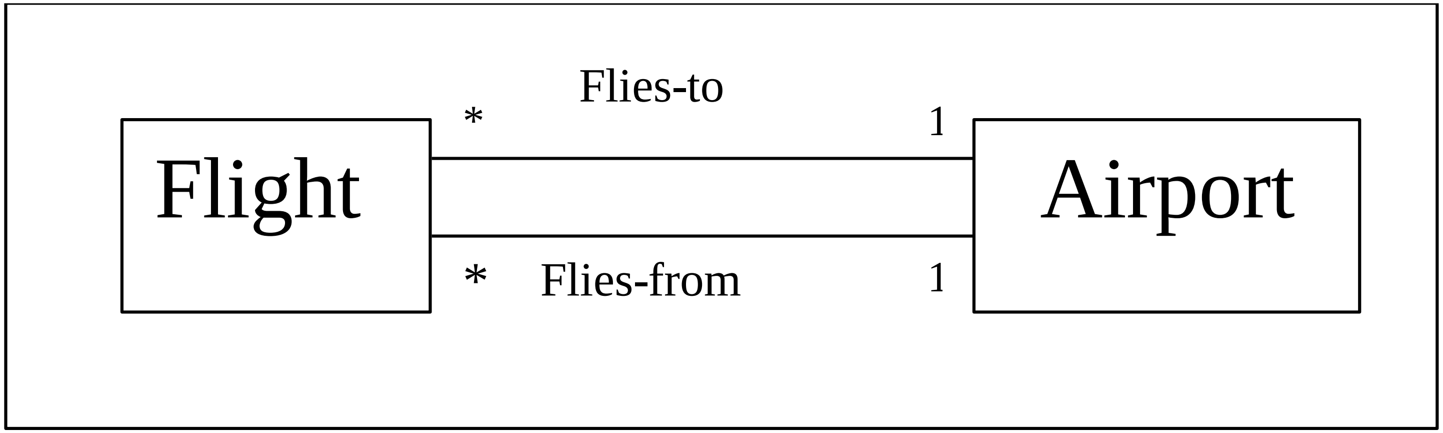


Domain Models – Identify Associations

- The common categories that are worth considering are:
  - » A is a *physical part* of B . *Eg: Wing-Airplane*
  - » A is a *logical part* of B. *Eg: SalesLineItem-Sale.*
  - » A is *physically contained* in B . *Eg: Register- Store.*
  - » A is *logically contained* in B. *Eg:ItemDescription-Catalog.*
  - » A is a *description* of B.*Eg:ItemDescription-Item.*
  - » A is a line item of a transaction or report B.*Eg:SalesLineItem-Sale*.
  - » A is a *member* of B .*Eg: Cashier-Store.*
  - » A *uses or manages* B.*Eg:Cashier-Register.*

Domain Models – Identify Associations

- Multiple associations
  - » Two objects may have multiple associations between them.



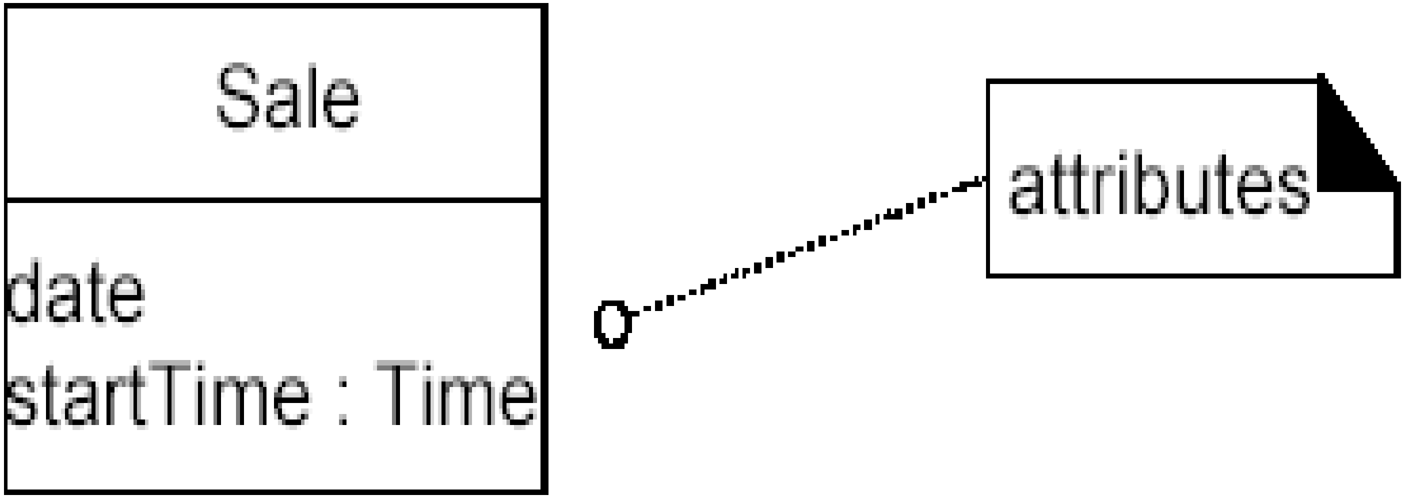
Domain Models – Design Issues

- When in doubt if the concept is required, keep the concept, *Why?*
- When in doubt if the association is required, drop it, *why?*
- Do not keep derivable association.

Domain Models – Identify Attributes

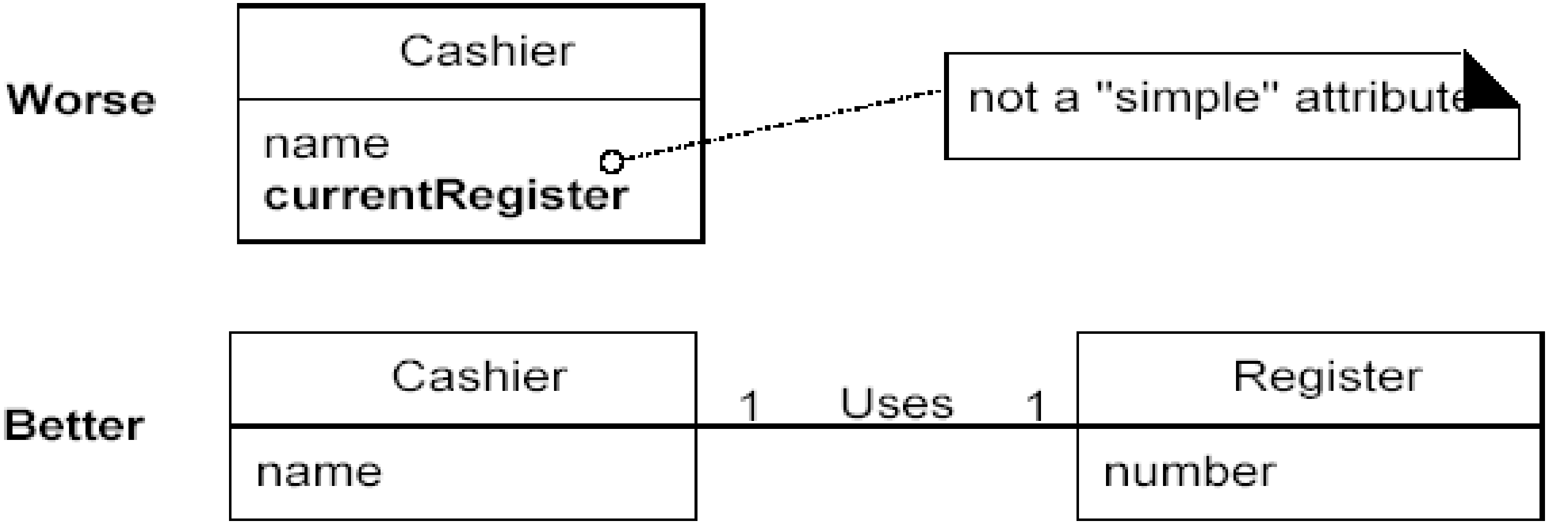
- *An attribute* is a *logical* data value of an object.

- Include the following attributes in a domain model:
  - » Those for which the requirements (for example, use cases) suggest or imply a need to remember information.
  - » Data type is *optional* in domain model.



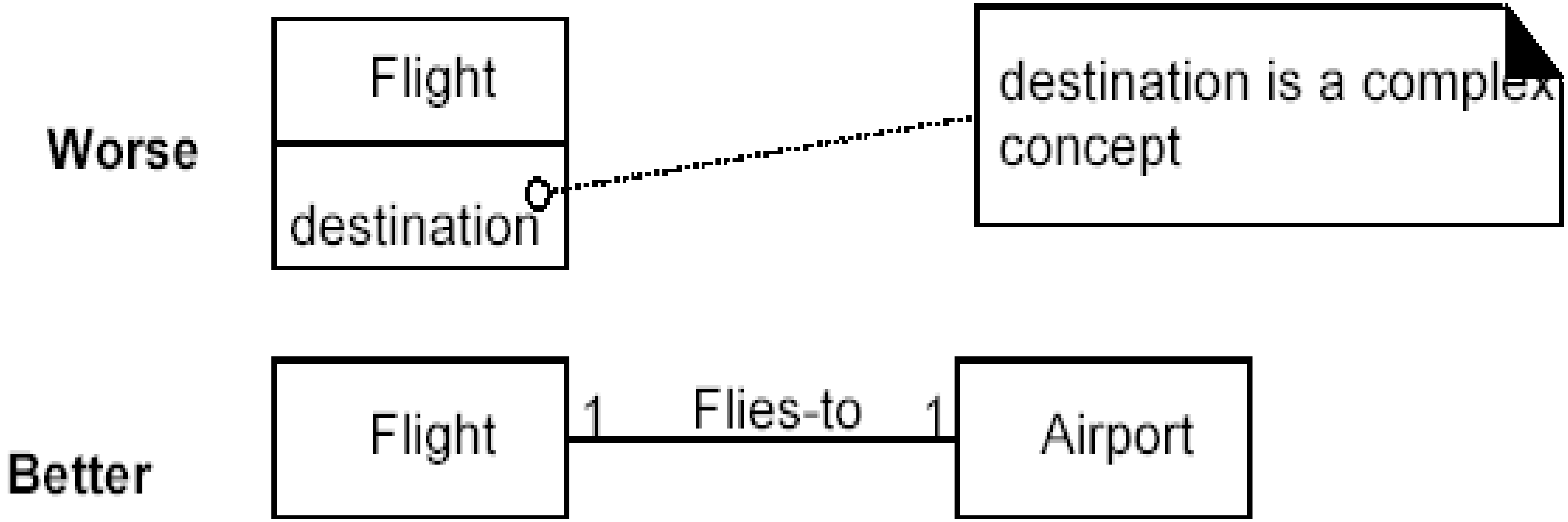
Domain Models – Identify Attributes

- Keep Attributes Simple
  - » If in *doubt*, define something as a *separate conceptual* class rather than as an attribute.



Domain Models – Identify Attributes

- Keep Attributes Simple


Worse
Flight
destination
destination is a complex concept
Better
Flight
1
Flies-to
1
Airport