

Applying UML & Patterns (3rd ed.)

Chapter 9

DOMAIN MODELS

Domain Models

- Fundamental OO analysis model
 - Shows important concepts in the domain of interest
 - Visual dictionary of concepts & their relationships
 - Inspiration for SW objects in class diagram
 - aka conceptual or analysis model
 - Depicts objects, their attributes, associations between objects
 - Does NOT show object's operations

Fig. 9.1

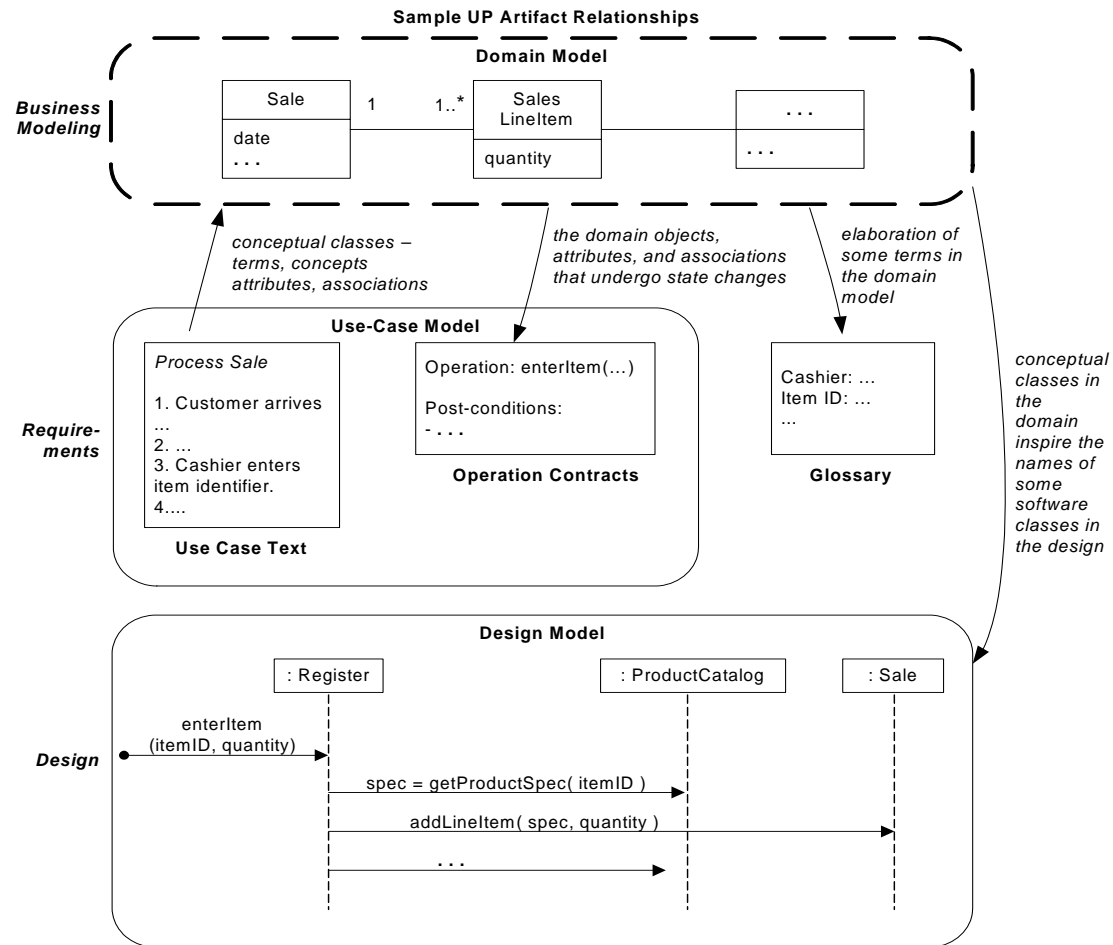


Fig. 9.2

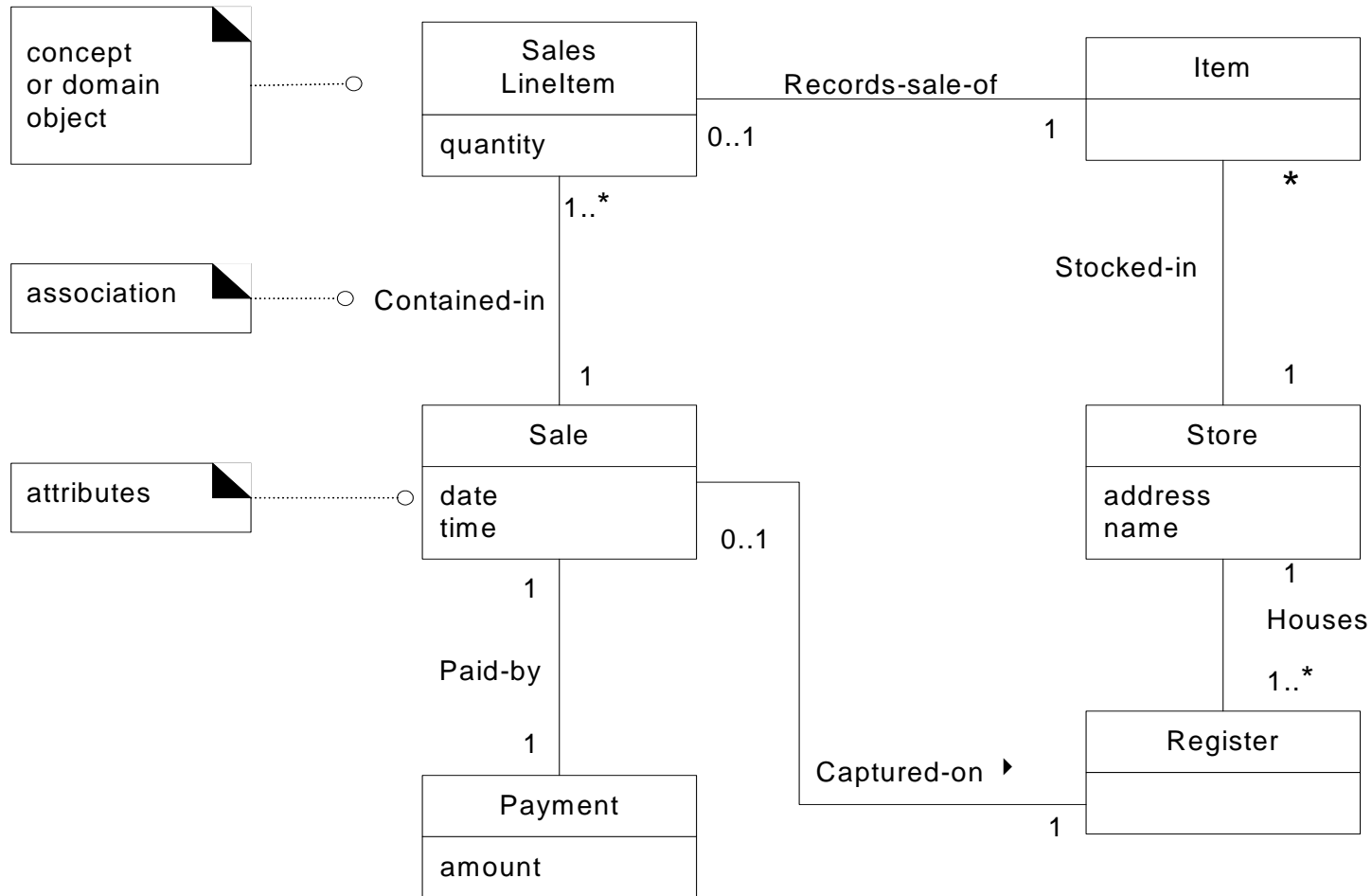


Fig. 9.3

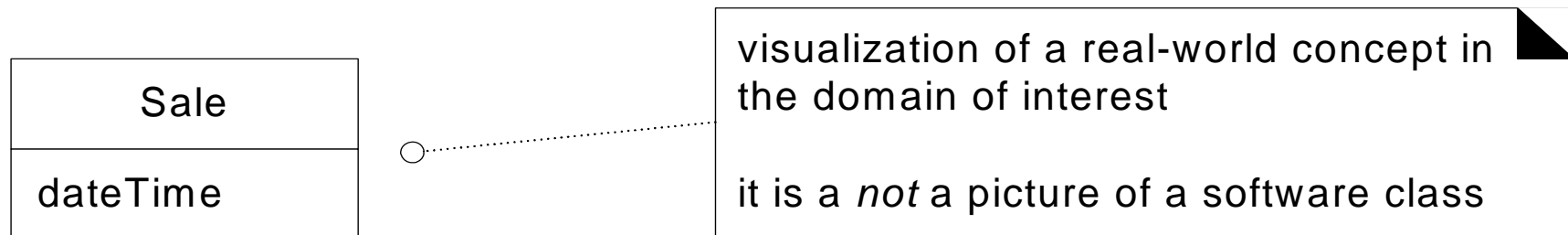


Fig. 9.4

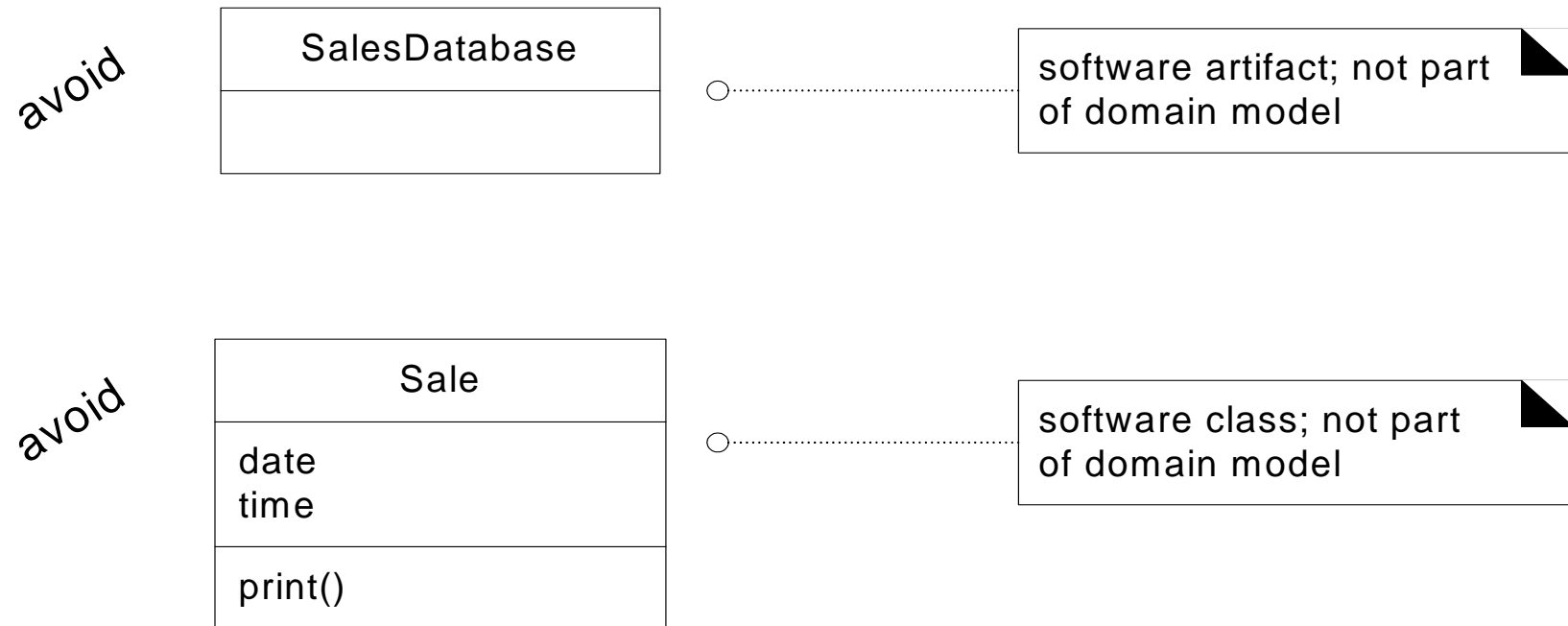
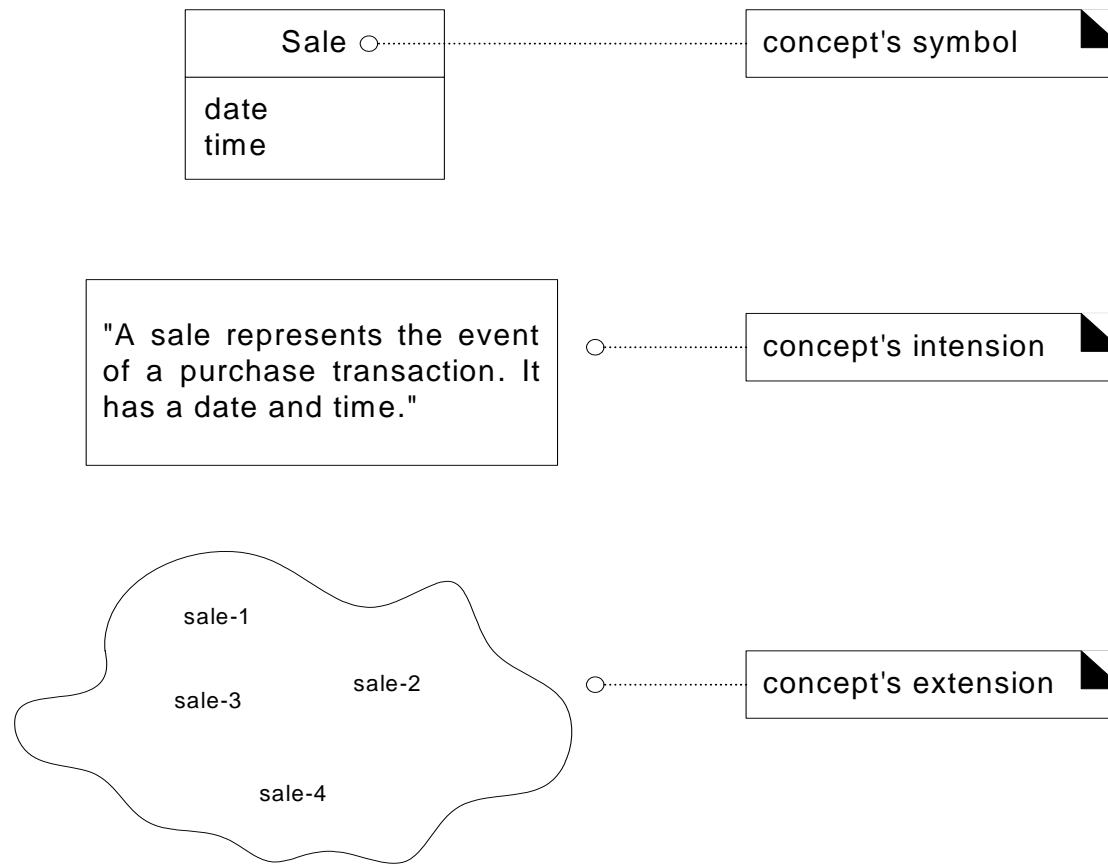


Fig. 9.5



Domain models

- Don't confuse with data model
 - Object might have attributes that aren't persisted
 - Object might have no attributes
 - Such objects wouldn't be in a data model

Domain Models

- Domain model simpler than full-blown design class model
- Easier for users to understand
- Use it to educate users about basic OO modeling
- Reduces tendency for modeler to drill too deep too quickly

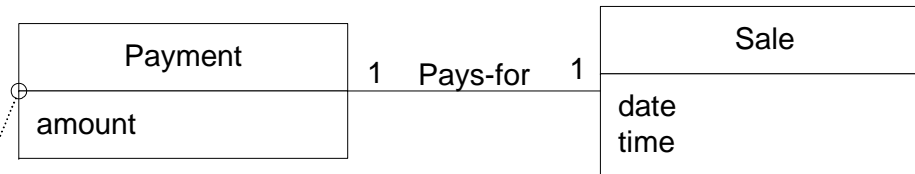
Fig. 9.6

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.

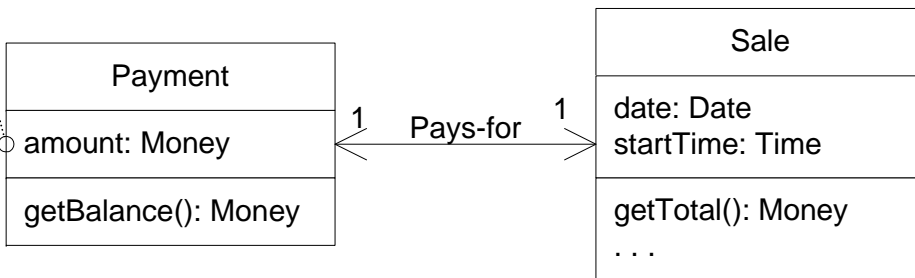
This reduces the representational gap.

This is one of the big ideas in object technology.

UP Domain Model
Stakeholder's view of the noteworthy concepts in the domain.



inspires
objects
and
names in



UP Design Model

The object-oriented developer has taken inspiration from the real world domain in creating software classes.

Therefore, the representational gap between how stakeholders conceive the domain, and its representation in software, has been lowered.

Domain Models

- How to create
 - Constrain model with current reqs--stay in scope!
 - Find conceptual classes using Larman's guidelines (p. 140)
 - Can also use noun's harvested from UCs
 - Add associations & attributes
 - Changes frequently so beware of wasted effort drawing models

Fig. 9.7



Fig. 9.8

MonopolyGame

Player

Piece

Die

Board

Square

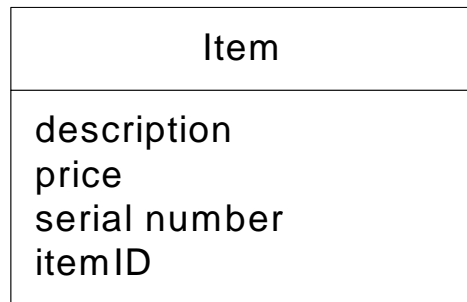
Domain Models

- Objects vs. attributes
 - Attributes are ‘simple’ data types
 - ex) number, text
 - Concepts that are described by simple attributes are objects
 - A Store has an address, phone number, etc.

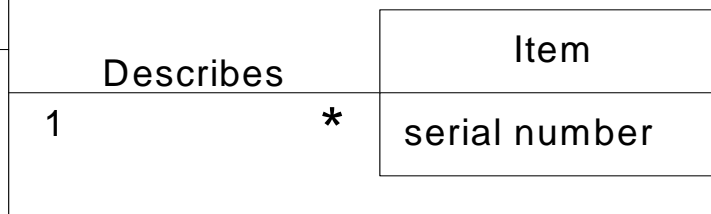
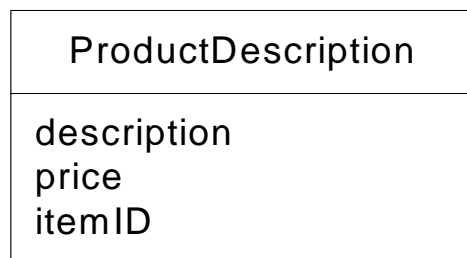
Domain Models

- Description classes
 - Contain info that describes something else, very common in OO models
 - If description is not separate from the underlying object, then deletion of the object results in loss of all info about that object (Item example, p. 147)
 - Also error prone due to data duplication

Fig. 9.9

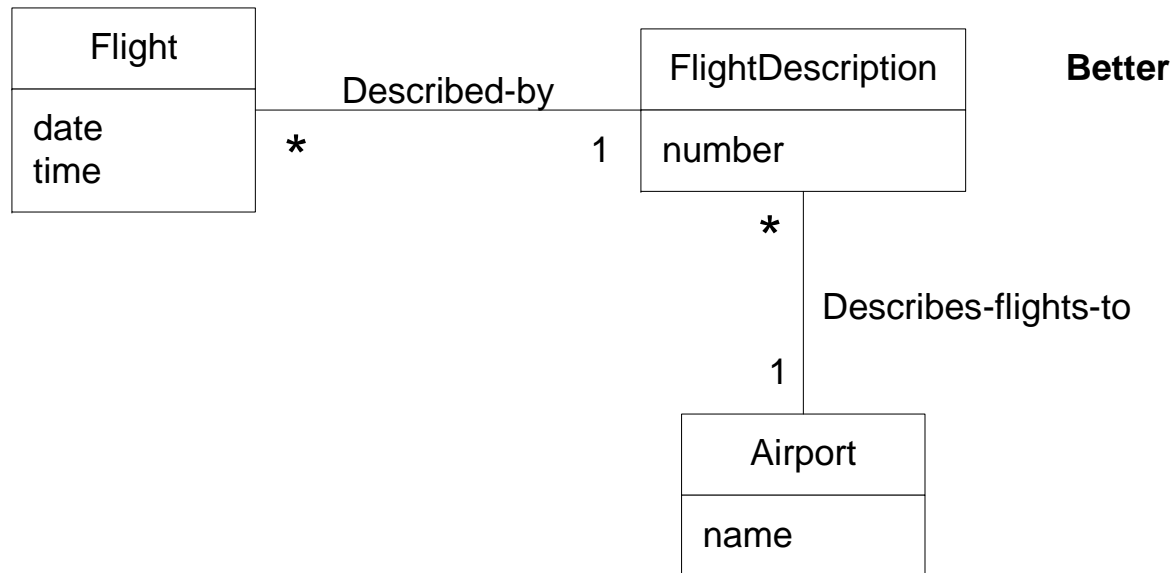
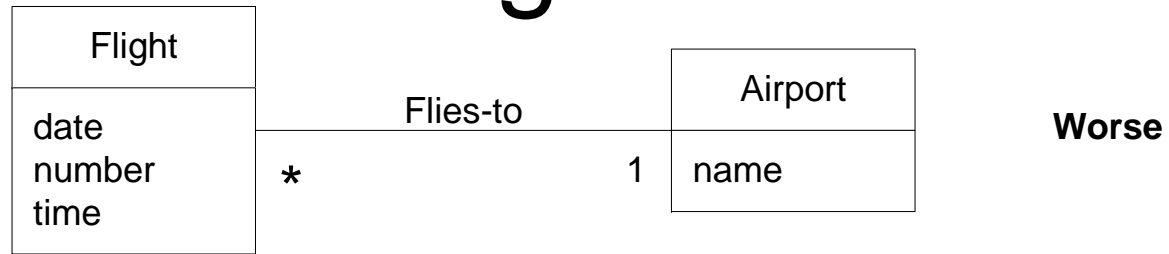


Worse



Better

Fig. 9.10



Domain Models

- Associations
 - A meaningful relationship between objects
 - When knowledge of relationship needs to be preserved
 - Derived from list of common associations (p. 155)

Domain Models

- Associations are NOT
 - A model of data flows
 - DB foreign keys
 - Instance variables
 - SW object connections
- Many, but not all, associations will be implemented eventually

Fig. 9.11

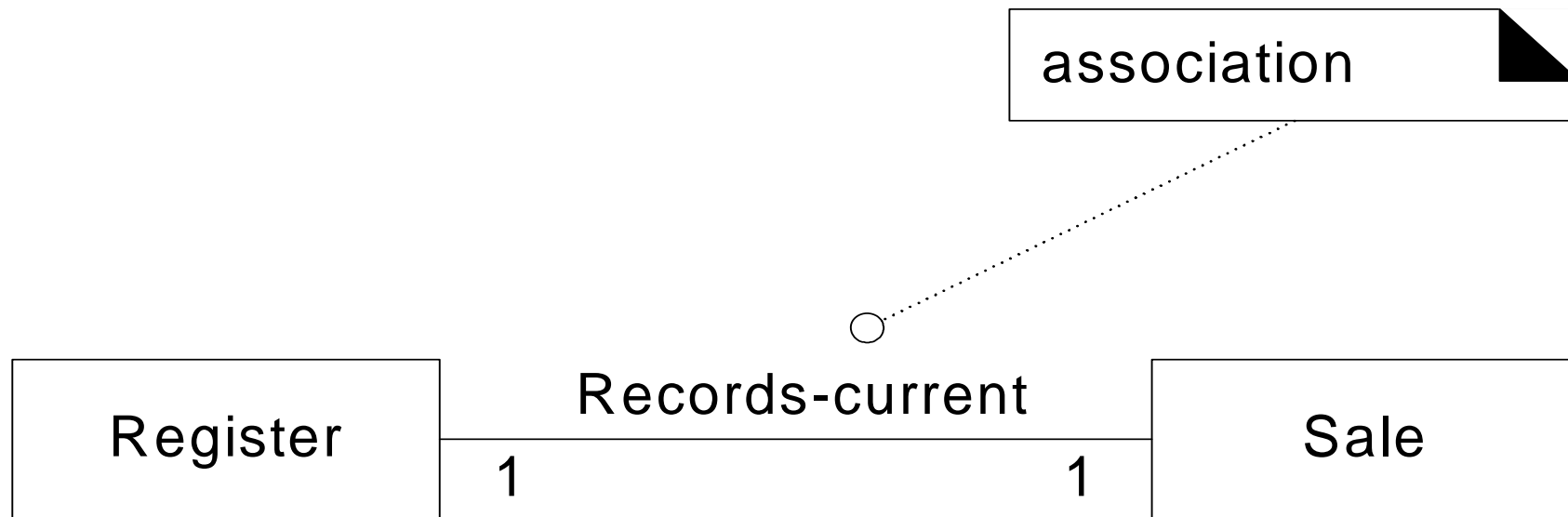


Fig. 9.12

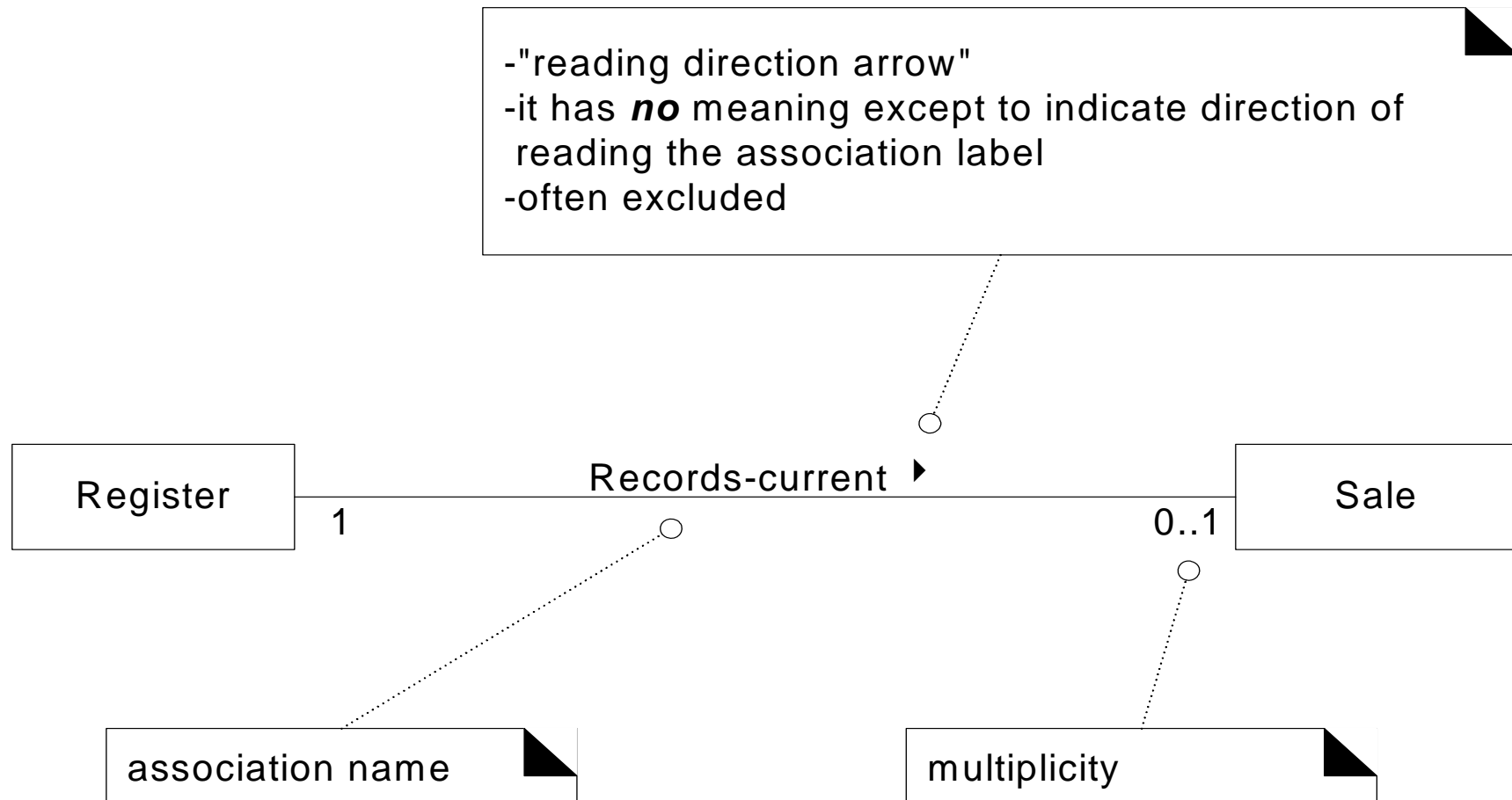
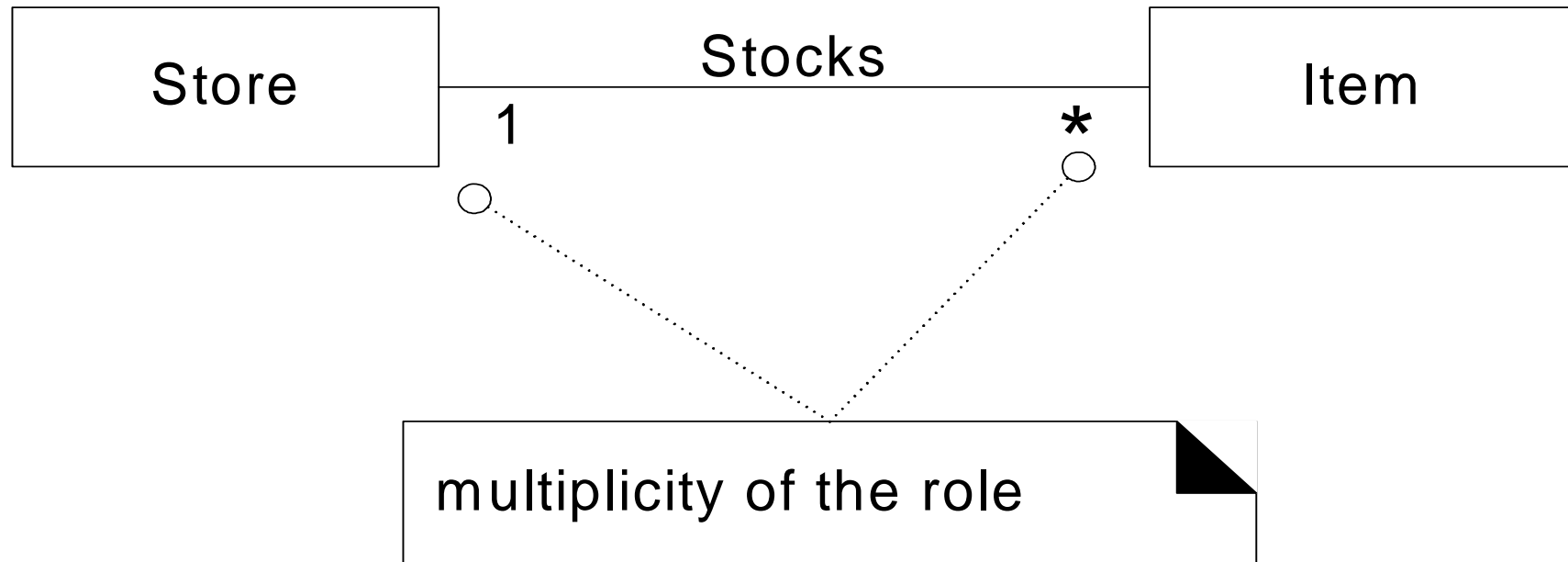


Fig. 9.13



Normally, the multiplicity at a particular moment in time

Fig. 9.14

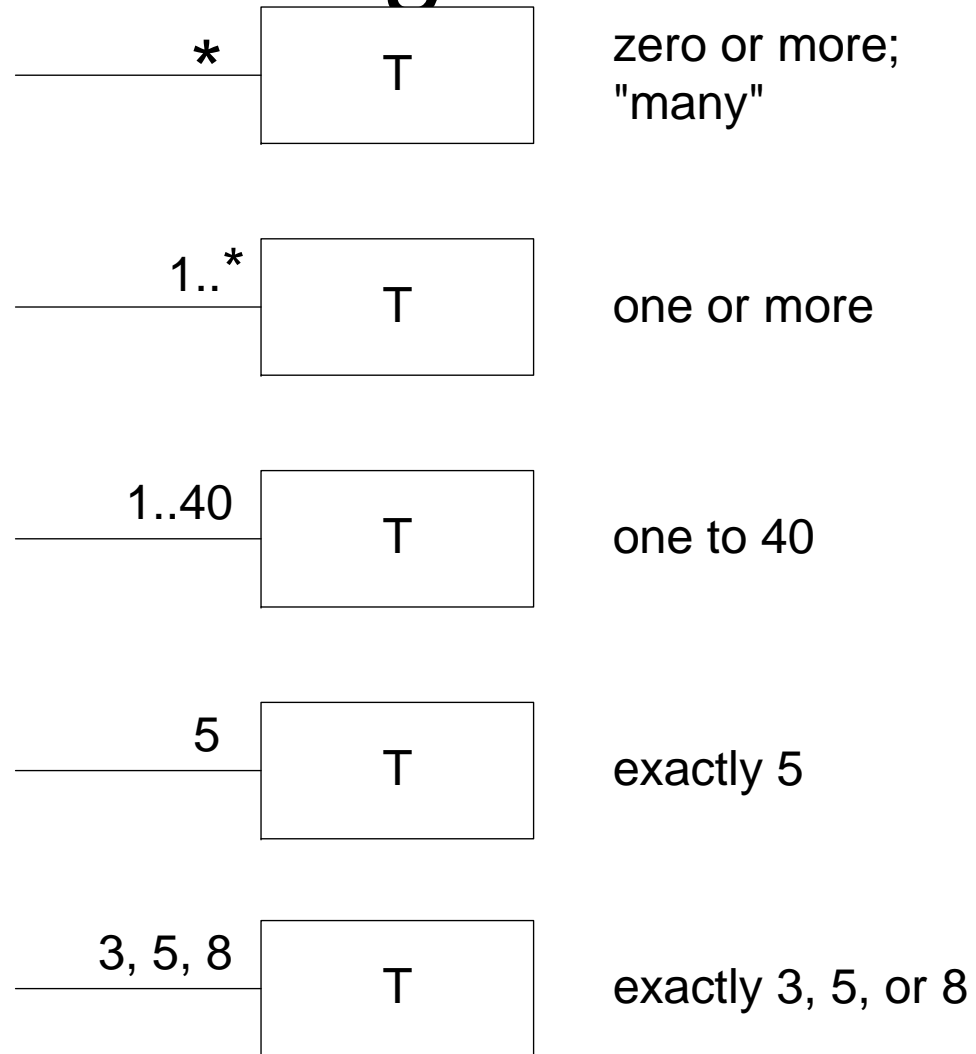


Fig. 9.15



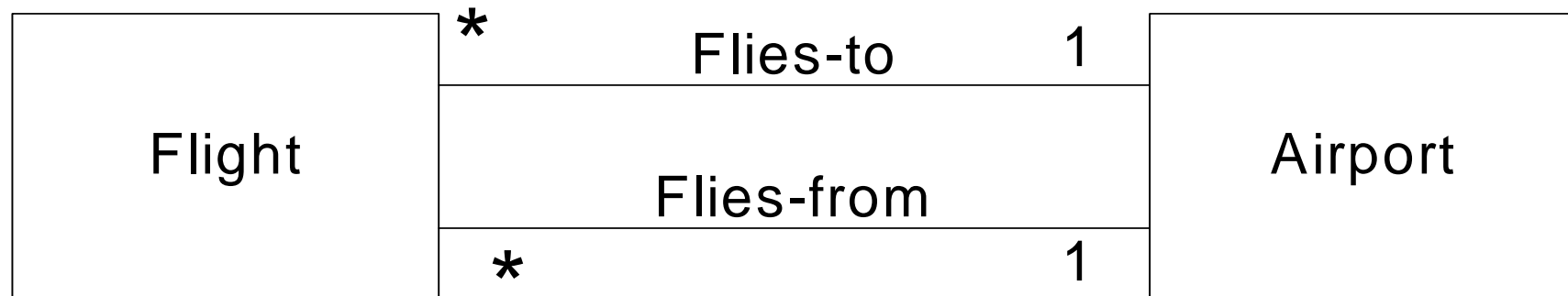
Multiplicity should "1" or "0..1"?

The answer depends on our interest in using the model. Typically and practically, the multiplicity communicates a domain constraint that we care about being able to check in software, if this relationship was implemented or reflected in software objects or a database. For example, a particular item may become sold or discarded, and thus no longer stocked in the store. From this viewpoint, "0..1" is logical, but ...

Do we care about that viewpoint? If this relationship was implemented in software, we would probably want to ensure that an *Item* software instance would always be related to 1 particular *Store* instance, otherwise it indicates a fault or corruption in the software elements or data.

This partial domain model does not represent software objects, but the multiplicities record constraints whose practical value is usually related to our interest in building software or databases (that reflect our real-world domain) with validity checks. From this viewpoint, "1" may be the desired value.

Fig. 9.16



Can have more than 1 association between classes.
A class can also have an association with itself! Can you think of an example?

Fig. 9.17

Records-sale-of

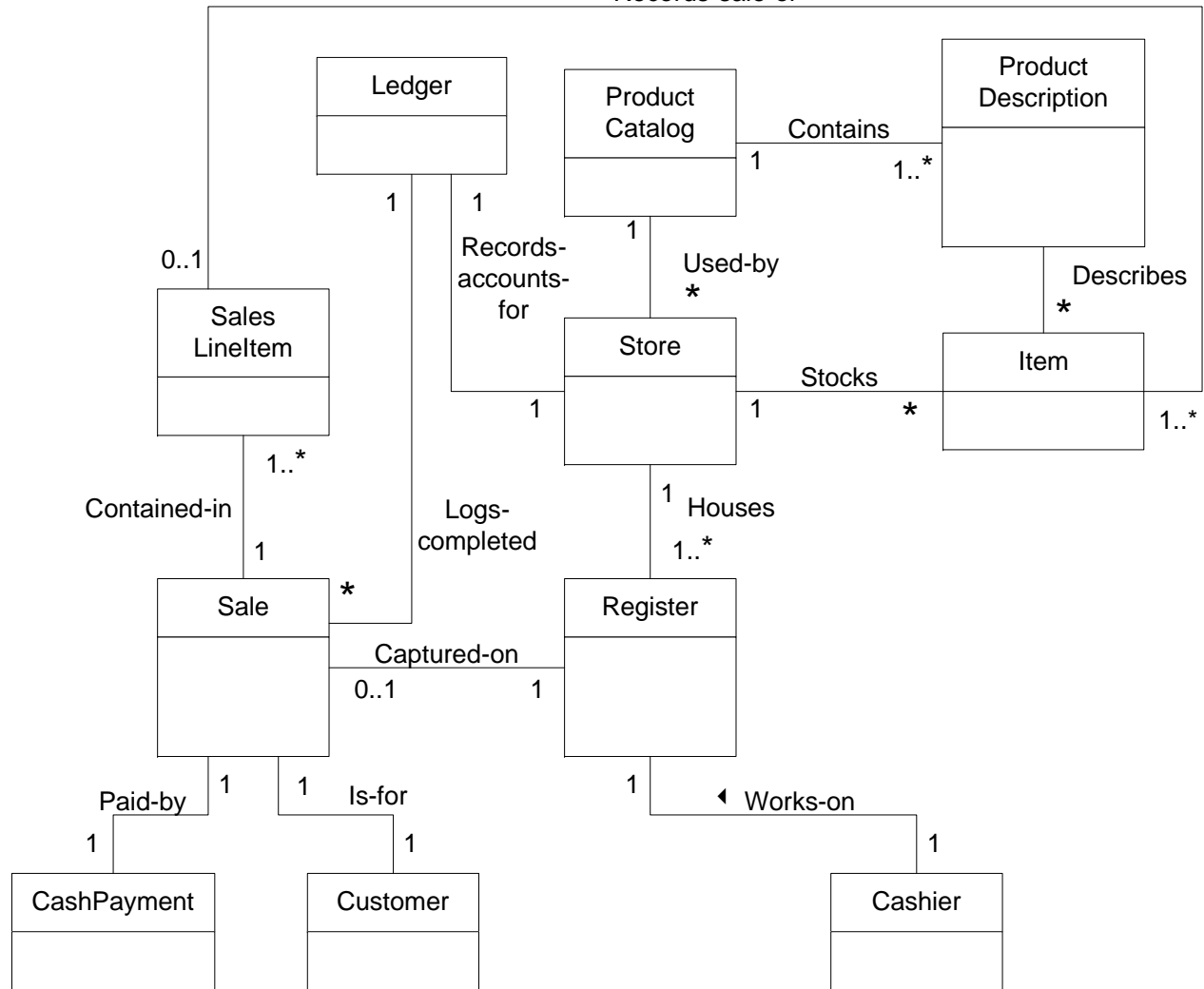
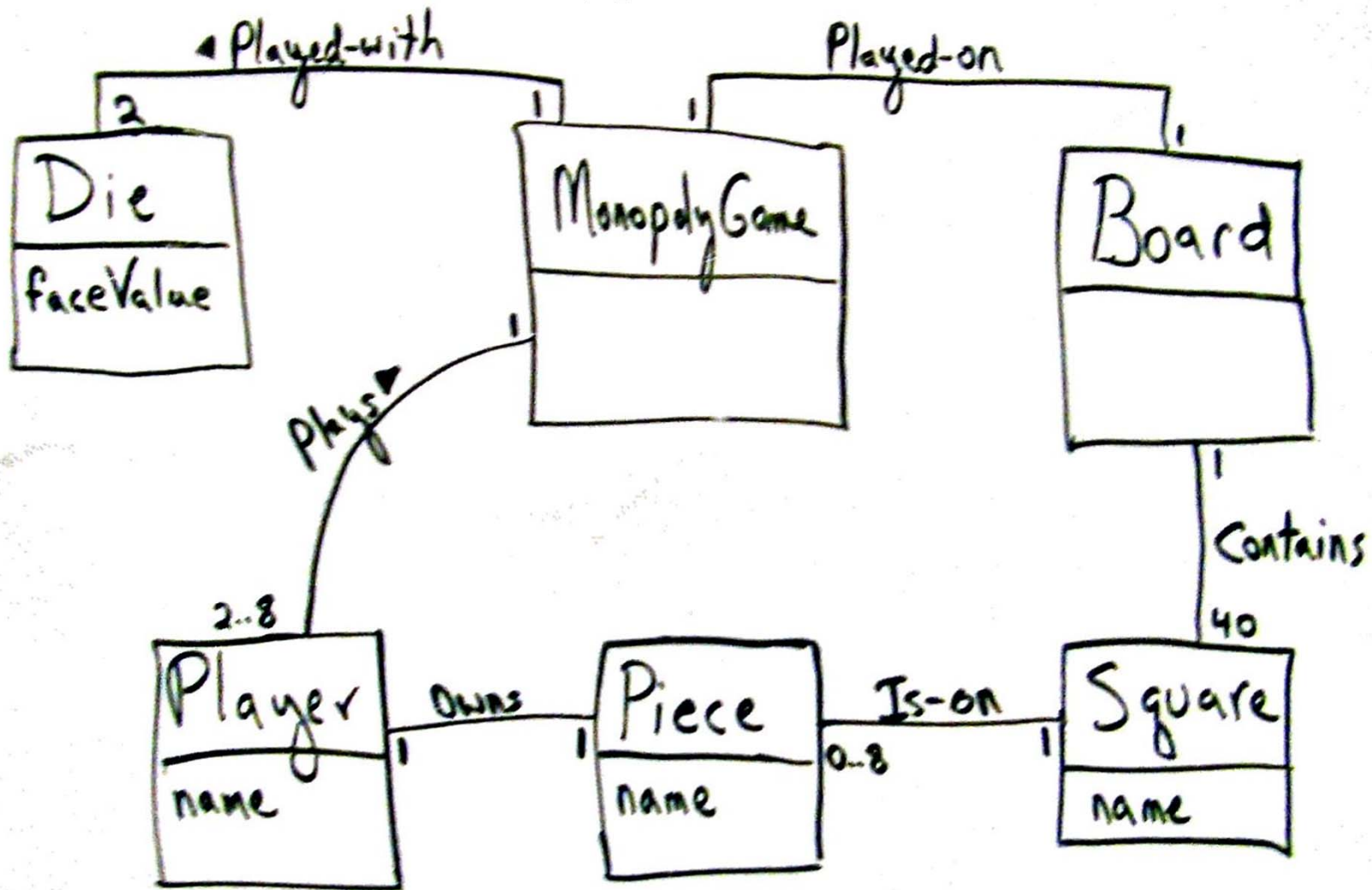


Fig. 9.18



Domain Models

- Attributes
 - A logical data value of an object that needs to be remembered
 - Some attributes are derived from other attributes
 - The usual ‘primitive’ data types
 - Numbers, characters, booleans
 - Common compound data types
 - Date, time (or dateTime), address, SSN, phoneNumber, bar codes, etc.
 - May become full class objects in design (see. p. 164)

Fig. 9.19

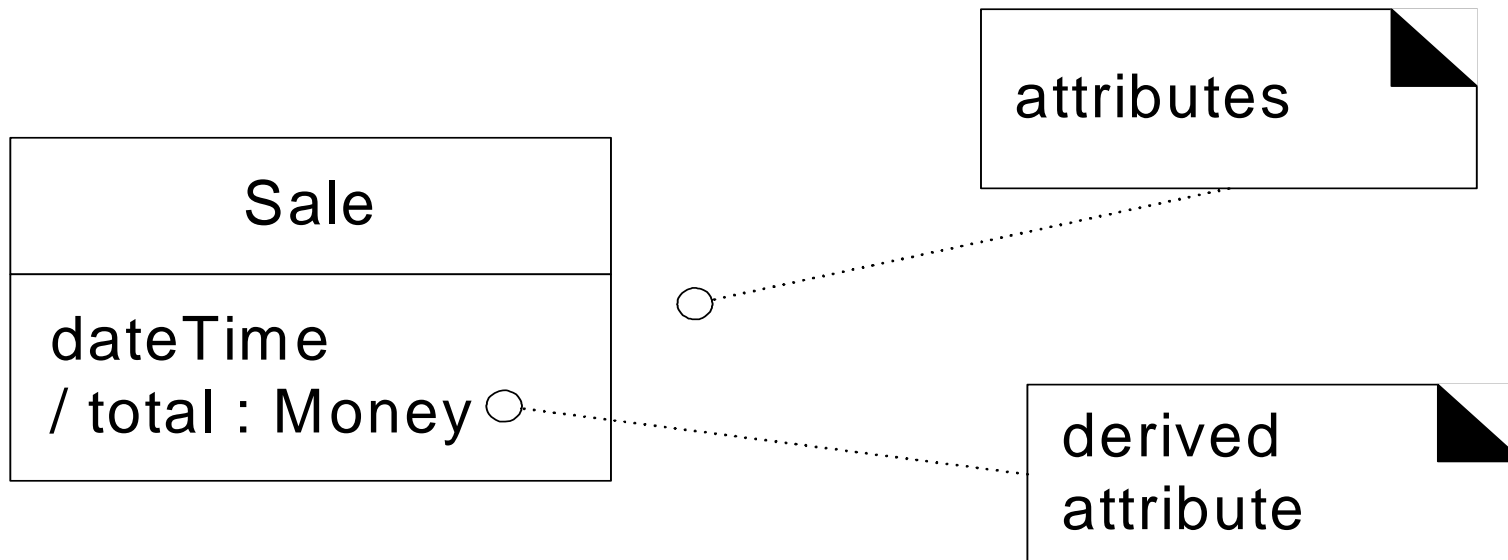
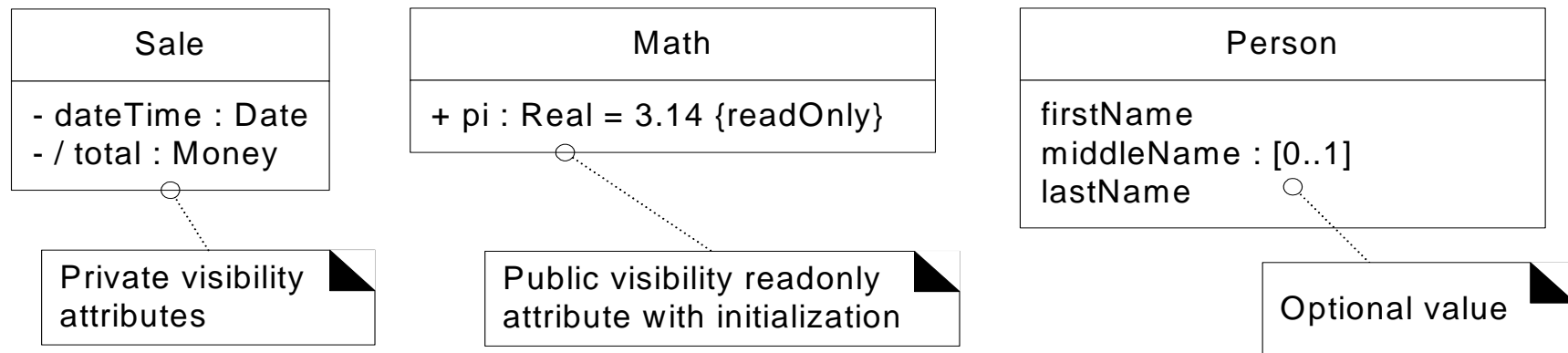


Fig. 9.20



Showing visibility on domain model is probably overkill. We'll talk about visibility later in the course.

Fig. 9.21

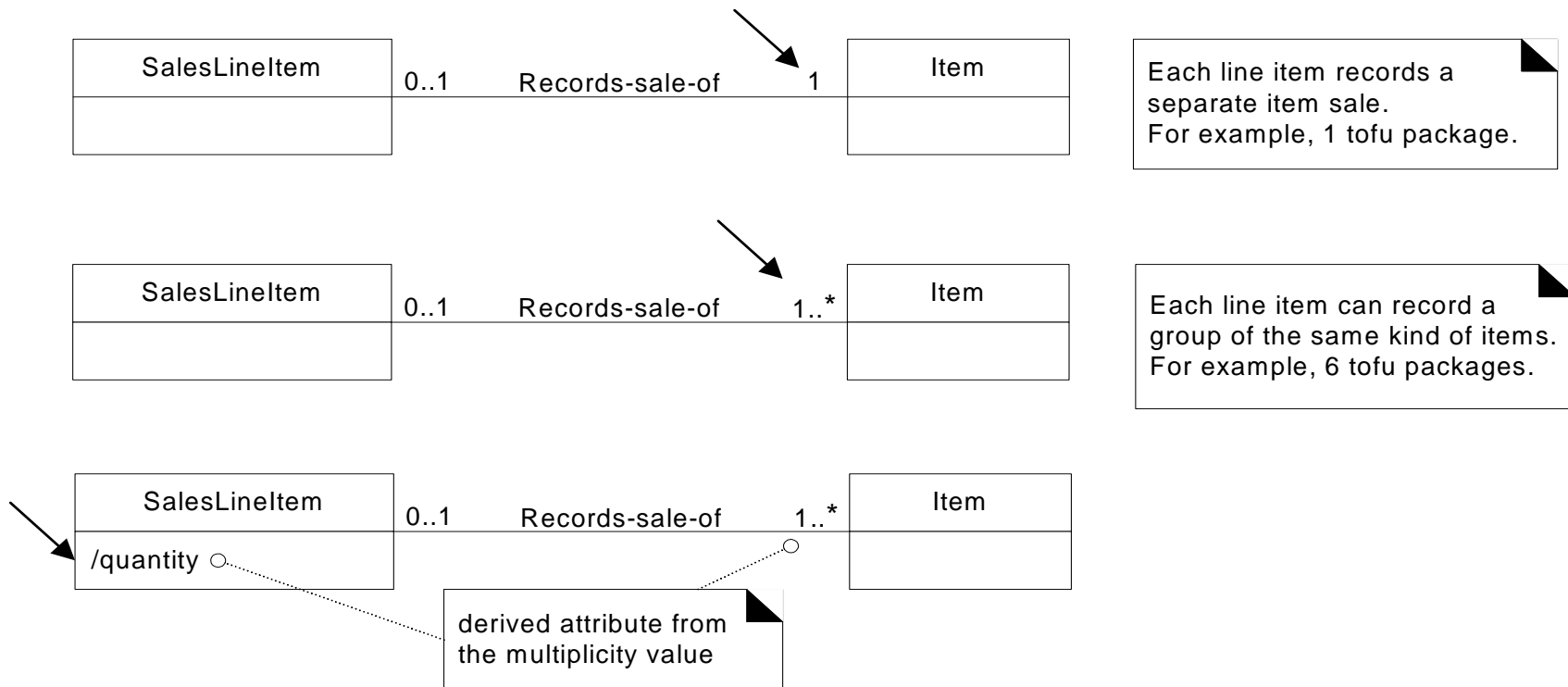
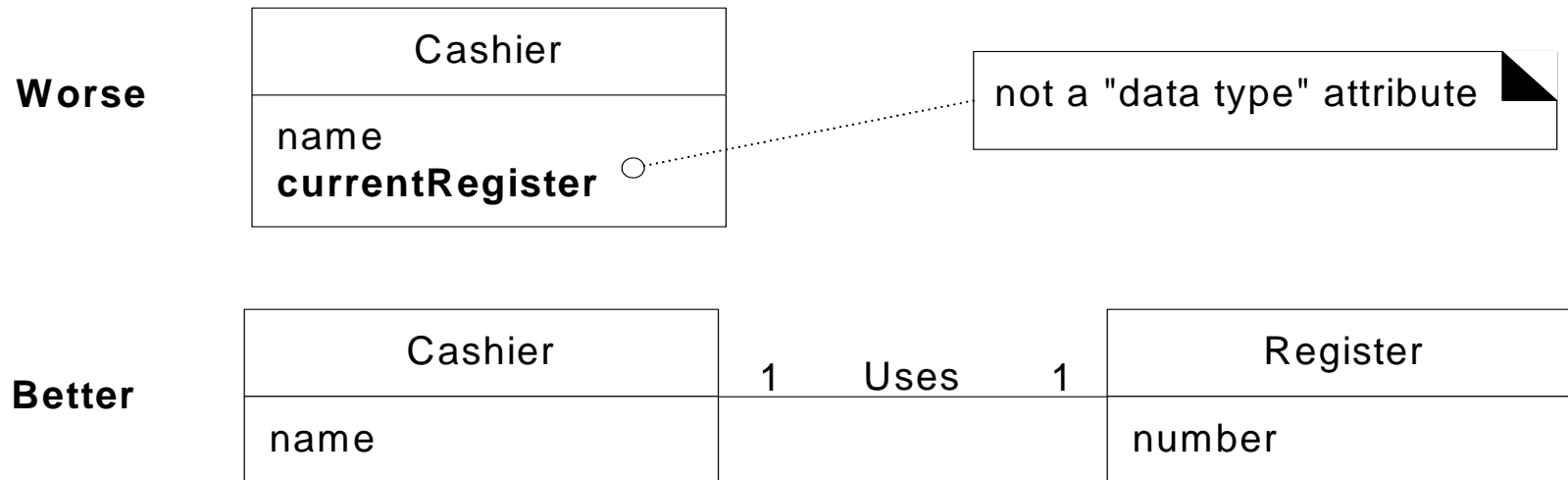


Fig. 9.22



Don't use an attribute in lieu of an association

Fig. 9.23

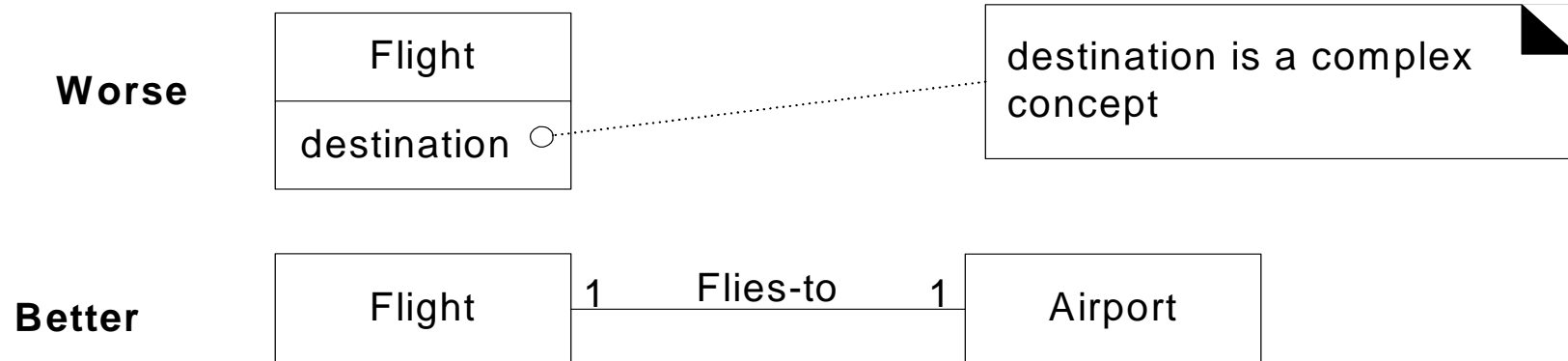


Fig. 9.24

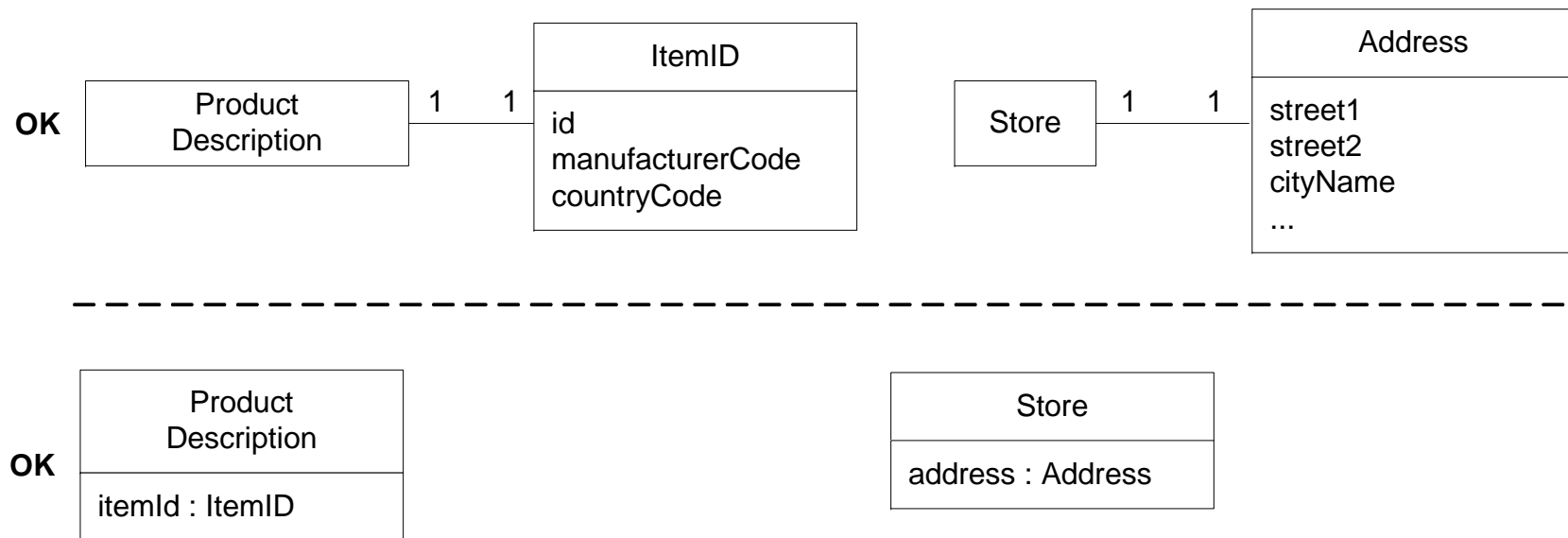


Fig. 9.25

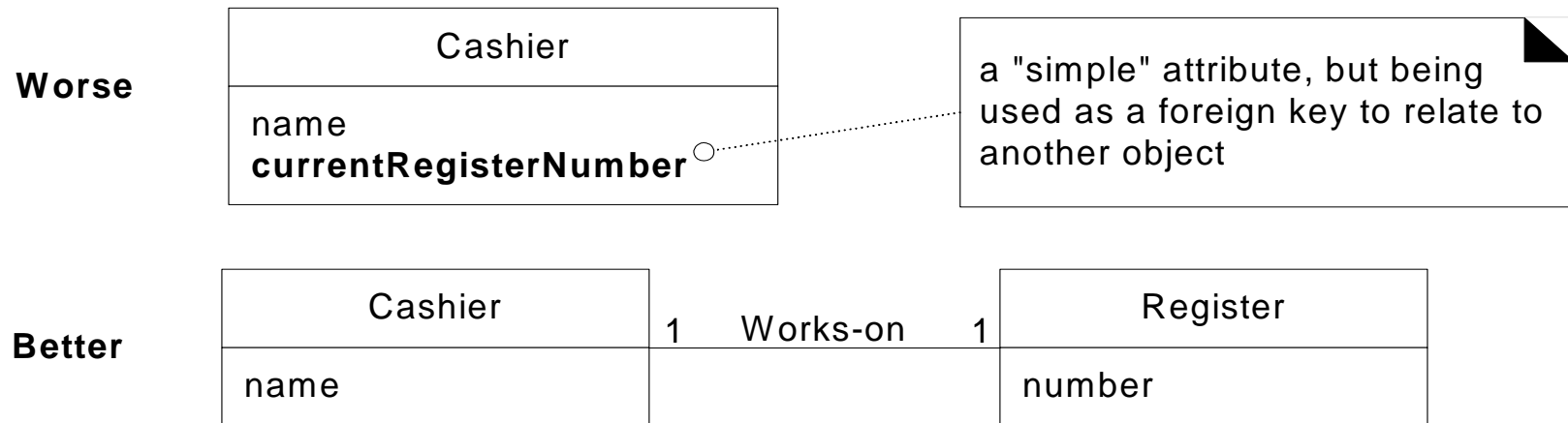


Fig. 9.26

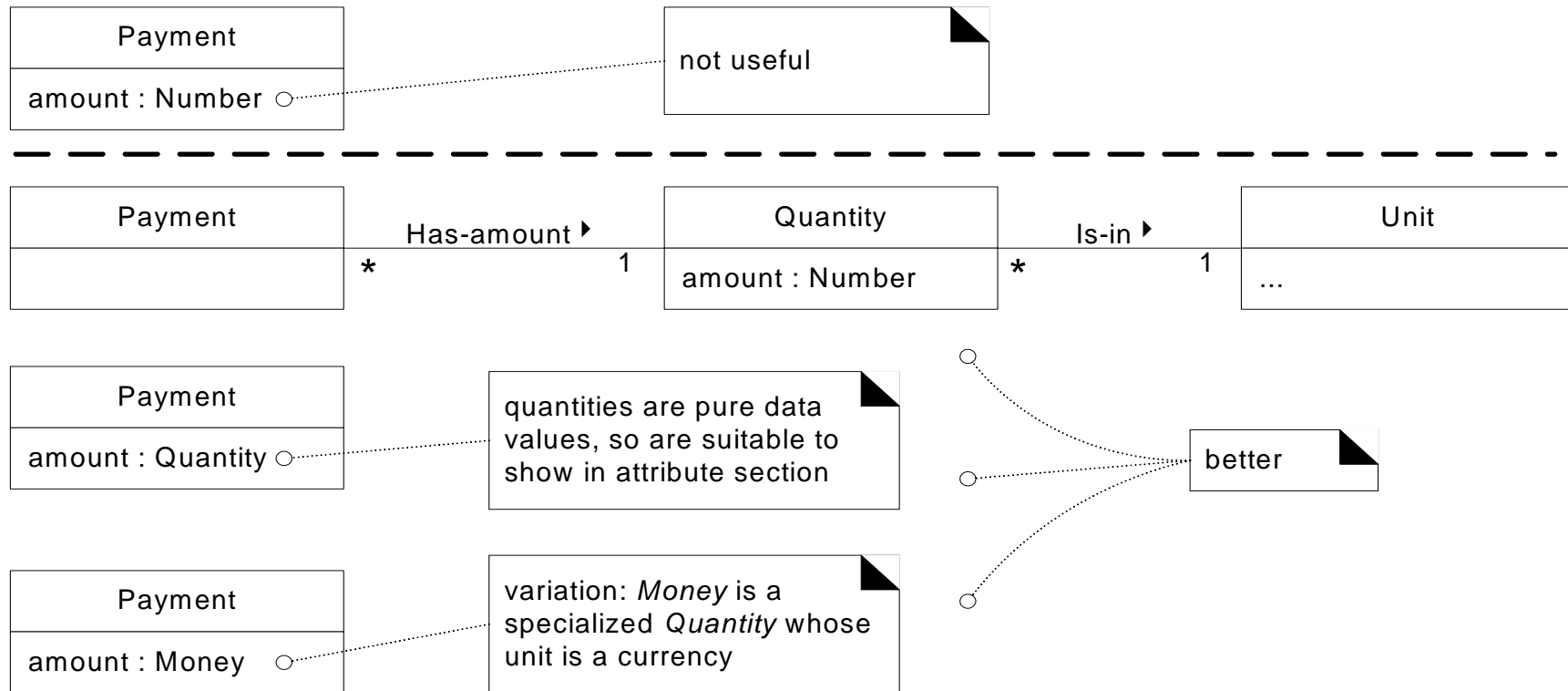


Fig. 9.27

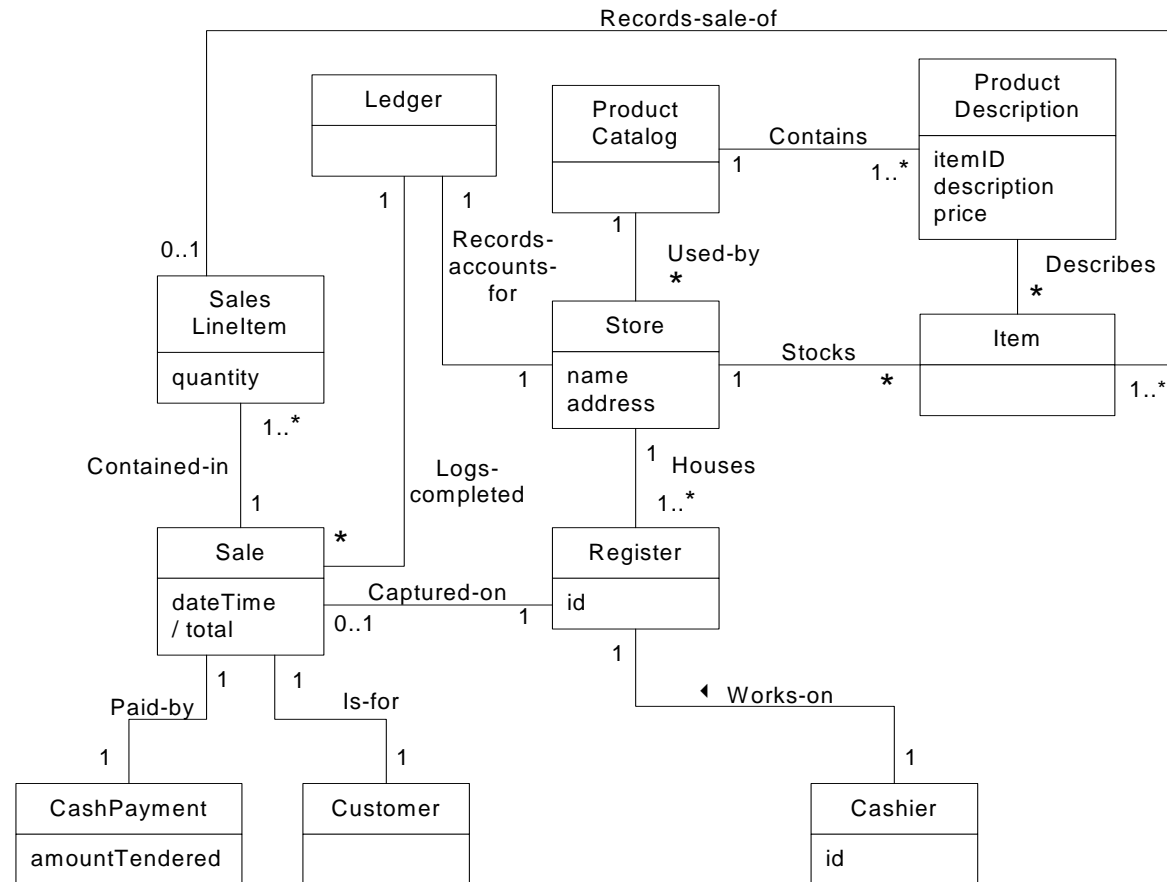


Fig. 9.28

