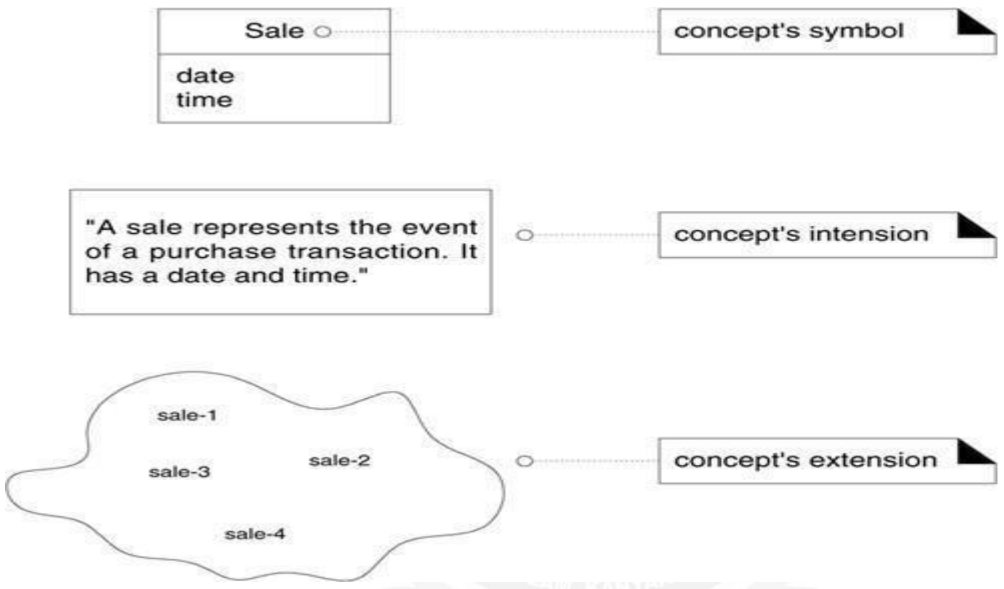
CONCEPTUAL CLASSES

A conceptual class is an idea, thing, or object. It may be considered in terms of its symbol, intension, and extension (see Figure).

- 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.

For example, consider the conceptual class for the event of a purchase transaction. I may choose to name it by the (English) symbol Sale. The intension of a Sale may state that it "represents the event of a purchase transaction, and has a date and time." The extension of Sale is all the examples of sales; in other words, the set of all sale instances in the universe.



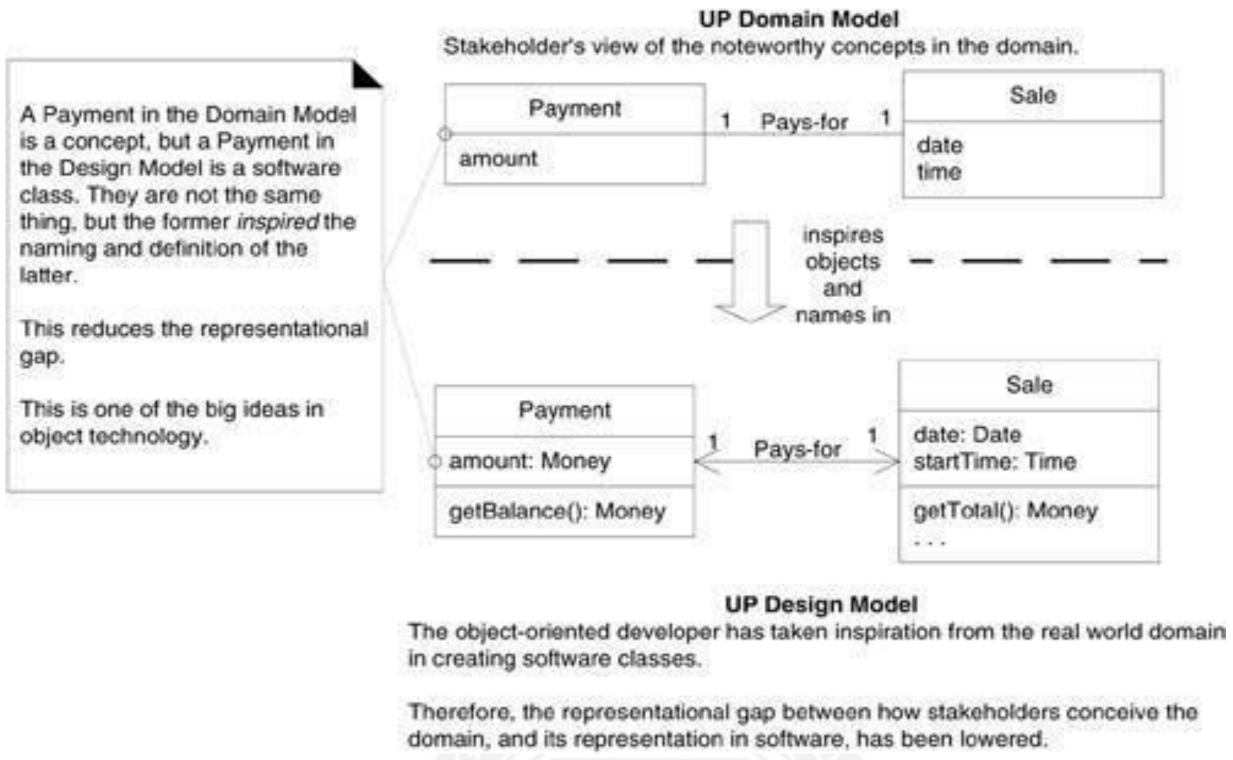
A conceptual class has a symbol, intension and extension

Are Domain and Data Models the Same Thing?

A domain model is not a data model (which by definition shows persistent data to be stored somewhere), so do not exclude a class simply because the requirements don't indicate any obvious need to remember information about it or because the conceptual class has no attributes. For example, it's valid to have attribute less conceptual classes, or conceptual classes that have a purely behavioral role in the domain instead of an information role.

Motivation: Why Create a Domain Model ?

Lower Representational Gap with OO Modeling : This is a key idea in OO: Use software class names in the domain layer inspired from names in the domain model, with objects having domain-familiar information and responsibilities. This supports a low representational gap between our mental and software models.


UP Domain Model
Stakeholder's view of the noteworthy concepts in 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.
Payment
amount
1 Pays-for 1
Sale
date
time
inspires objects and names in
Payment
amount: Money
getBalance(): Money
1 Pays-for 1
Sale
date: Date
startTime: Time
getTotal(): Money
. . .
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.

Lower Representational Gap with OO Modeling

Guideline: How to Create a Domain Model?

Bounded by the current iteration requirements under design:

1. Find the conceptual classes (see a following guideline).
2. Draw them as classes in a UML class diagram.
3. Add associations and attributes.

Guideline: To Find Conceptual Classes

Three Strategies to Find Conceptual Classes :

1. Reuse or modify existing models. This is the first, best, and usually easiest approach. There are published, well-crafted domain models and data models for many common domains, such as inventory, finance, health, and so forth.
2. Use a category list. (Method 2)
3. Identify noun phrases. (Method 3)

Method 2: Use a Category List

We can create a domain model by making a list of candidate conceptual classes. The guidelines also suggest some priorities in the analysis. Examples are drawn from the 1) POS, 2) Monopoly game 3) airline reservation domains.

Method 3: Finding Conceptual Classes with Noun Phrase Identification

Another useful technique (because of its simplicity) suggested is linguistic analysis: Identify the nouns and noun phrases in textual descriptions of a domain, and consider them as candidate conceptual classes or attributes.

Guideline

Linguistic analysis has become more sophisticated; it also goes by the name natural language modeling. for example, The current scenario of the Process Sale use case can be used.

Main Success Scenario (or Basic Flow):

1. **<u>Custome</u>**r arrives at a **<u>POS checkout</u>** with goods and/or **services** to purchase.
2. **<u>Cashier</u>** starts a new **<u>sale</u>**.
3. **<u>Cashier</u>** enters item **<u>identifier</u>.**
4. System records **<u>sale line item</u>** and presents **<u>item description</u> , price**, and running **<u>total</u>**. Price calculated from a set of price rules.

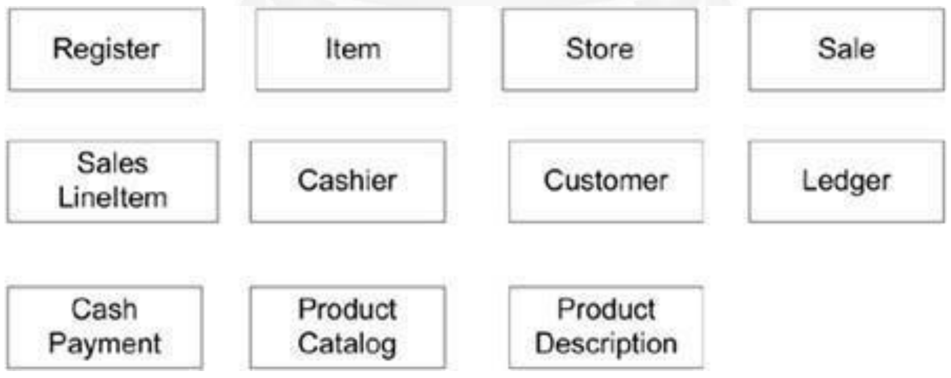
   Underlined words are nouns. The next level of scrutiny derives class names.

<u>Example: Find and Draw Conceptual Classes</u>

Case Study: POS Domain

From the category list and noun phrase analysis, a list is generated of candidate conceptual classes for the domain. There is no such thing as a "correct" list. It is a somewhat arbitrary collection of abstractions and domain vocabulary

Initial POS domain model.



Guidelines

1. Agile Modeling Sketching a Class Diagram : The sketching style in the UML class diagram is to keep the bottom and right sides of the class boxes open. This makes it easier to grow the classes as we discover new elements.

2. Agile Modeling Maintain the Model in a Tool? The purpose of creating a domain model is to quickly understand and communicate a rough approximation of the key concepts.

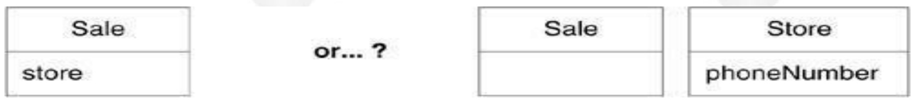
3. Report Objects - Include 'Receipt' in the Model? Receipt is a term in the POS domain. But it's only a report of a sale and payment, and thus duplicate information.

4. Use Domain Terms :
Make a domain model in the spirit of how a cartographer or mapmaker works:

- Use the existing names in the territory. For example, if developing a model for a library, name the customer a "Borrower" or "Patron" the terms used by the library staff.
- Exclude irrelevant or out-of-scope features. For example, in the Monopoly domain model for iteration-1
- Do not add things that are not there.

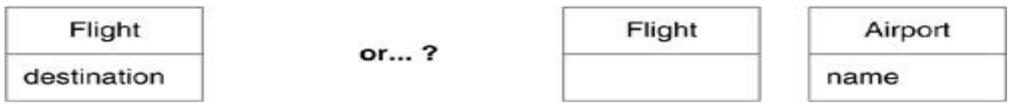
5. How to Model the Unreal World? Some software systems are for domains that find very little analogy in natural or business domains; software fortelecommunications is an example. For example, here are candidate conceptual classes related to the domain of a telecommunication switch: Message, Connection, Port, Dialog, Route, and Protocol.

6. A Common Mistake with 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. As an example, should store be an attribute of Sale, or a separate conceptual class Store?



In the real world, a store is not considered a number or text the term suggests a legal entity, an organization, and something that occupies space. Therefore, Store should be a conceptual class.

As another example, consider the domain of airline reservations. Should destination be an attribute of Flight, or a separate conceptual class Airport?

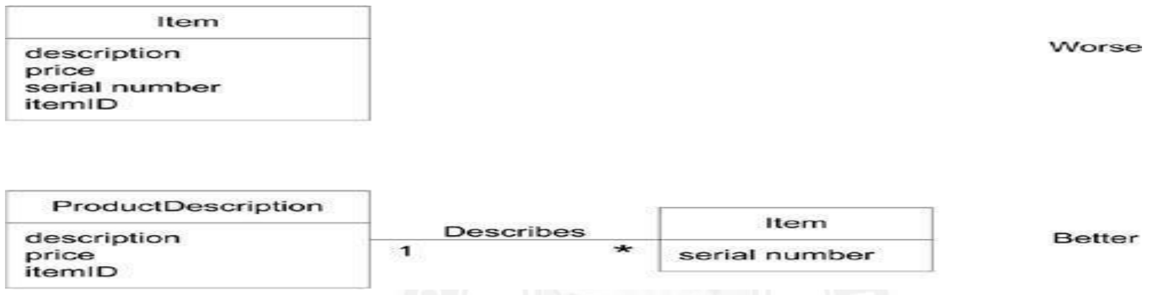


In the real world, a destination airport is not considered a number or text-it is a massive thing that occupies space. Therefore, Airport should be a concept.

'Description' Classes

7 When to Model with 'Description' Classes? A description class contains information that describes something else. For example, a Product Description that records the price, picture, and text description of an Item.

Motivation: Why Use 'Description' Classes? The need for description classes is common in many domain models. The 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 Figure. Descriptions about other things. The * means a multiplicity of "many." It indicates that one Product Description may describe many (*) Items**.**



When Are Description Classes Useful?

Add a description class (for example, Product Description) when:

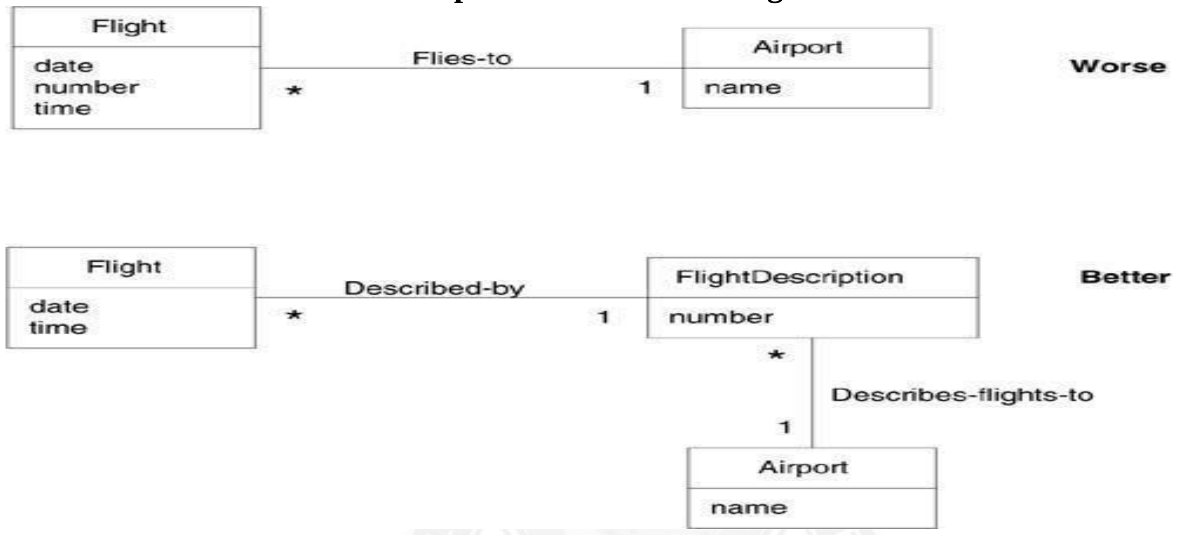
- There needs to be a description about an item or service, independent of the current existence of any examples of those items or services.
- Deleting instances of things they describe (for example, Item) results in a loss of information that needs to be maintained, but was incorrectly associated with the deleted thing.
- It reduces redundant or duplicated information.

Example: Descriptions in the Airline Domain

As another example, consider an airline company that suffers a fatal crash of one of its planes. Assume that all the flights are cancelled for six months pending completion of an investigation. Also assume that when flights are cancelled, their corresponding Flight software objects are deleted from computer memory. Therefore, after the crash, all Flight software objects are deleted.

If the only record of what airport a flight goes to is in the Flight software instances, which represent specific flights for a particular date and time, then there is no longer a record of what flight routes the airline has The problem can be solved, both from a purely conceptual perspective in a domain model and from a software perspective in the software designs, with a FlightDescription that describes a flight and its route, even when a particular flight is not scheduled in following figure

Descriptions about other things.



Note that the prior example is about a service (a flight) rather than a good. Descriptions of services or service plans are commonly needed.