Object-Oriented Software Analysis and Design

School of Computer Science
University of Windsor

Sample UP artifact influence

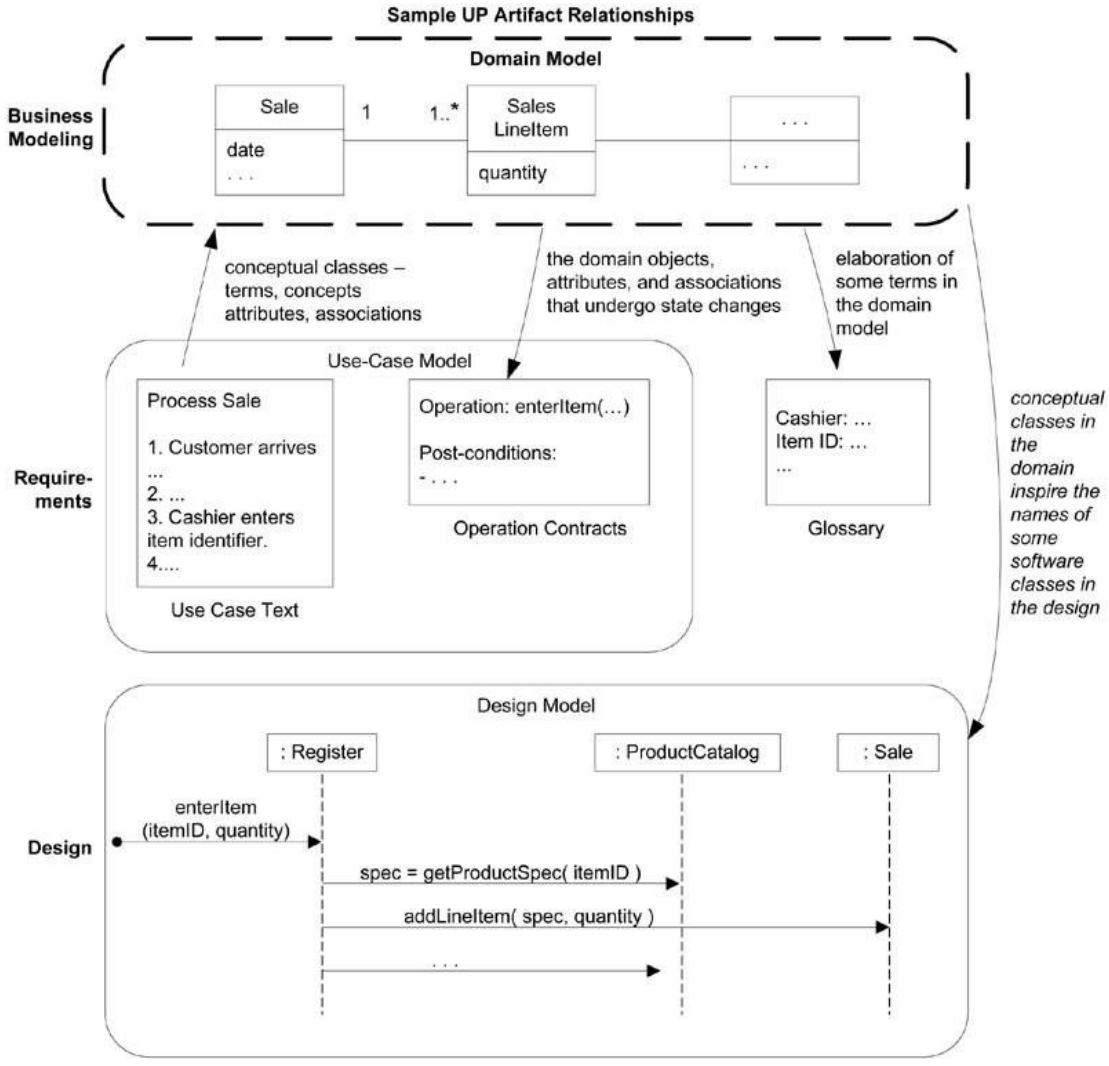


Figure: *Sample UP artifact influence.*

What is a Domain Model?

- A domain model is a **visual representation** of **conceptual classes** or **real-situation objects** in a domain.

- Domain models have also been called
  - conceptual models
  - domain object models, and
  - analysis object models.

- In the UP, the term "Domain Model" means a representation of real-situation conceptual classes, not of software objects.

- The term **does not** mean a set of diagrams describing software classes, the domain layer of a software architecture, or software objects with responsibilities.

What is a Domain Model? (contd.)

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

Domain Model: NextGen Example

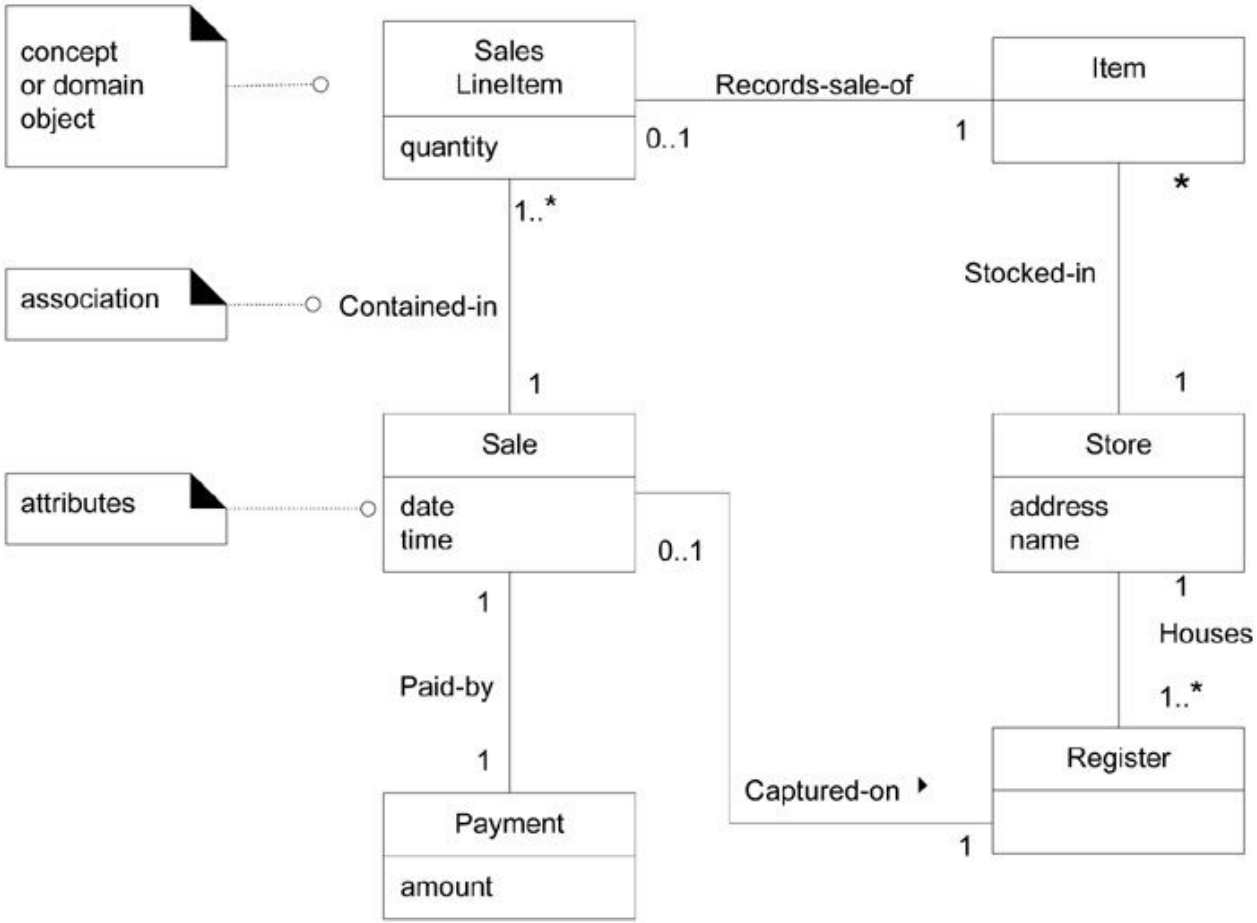


Figure: *Partial domain model a visual dictionary.*

Domain Model: NextGen Example (contd.)

- Figure shows a partial domain model drawn with UML **class diagram** notation.
- Identifying a rich set of **conceptual class**es is at the heart of OO analysis.
- Guideline: Avoid a waterfall-mindset big-modeling effort to make a thorough or "correct" domain model—it won't ever be either, and such over-modeling efforts lead to analysis paralysis, with little or no return on the investment.

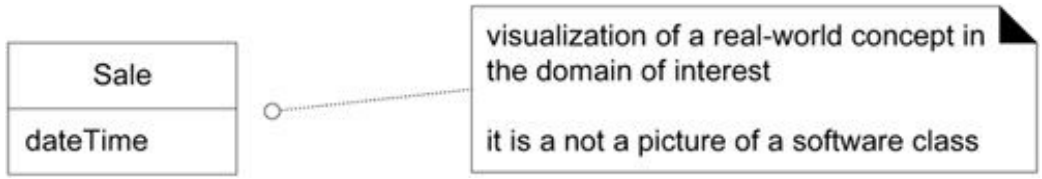
Why Call a Domain Model a "Visual Dictionary"?

- Refer to Figure 2. The information it illustrates (using UML notation) could alternatively have been expressed in plain text (in the UP Glossary).

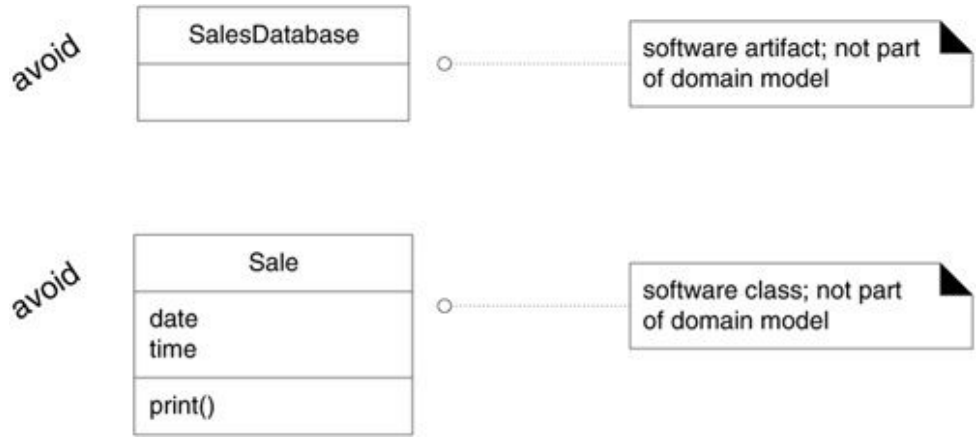
- But it's easy to understand the terms and especially their relationships in a visual language, since our brains are good at understanding visual elements and line connections.

- Therefore, the domain model is a **visual dictionary** of the noteworthy abstractions, domain vocabulary, and information content of the domain.

Is a Domain Model a Picture of Software Business Objects?



(a) A domain model shows real-situation conceptual classes, not software classes.



(b) A domain model does not show software artifacts or classes.

What are Conceptual Classes?

- The domain model illustrates conceptual classes or vocabulary in the domain.
- Informally, 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.

What are Conceptual Classes? (contd.)

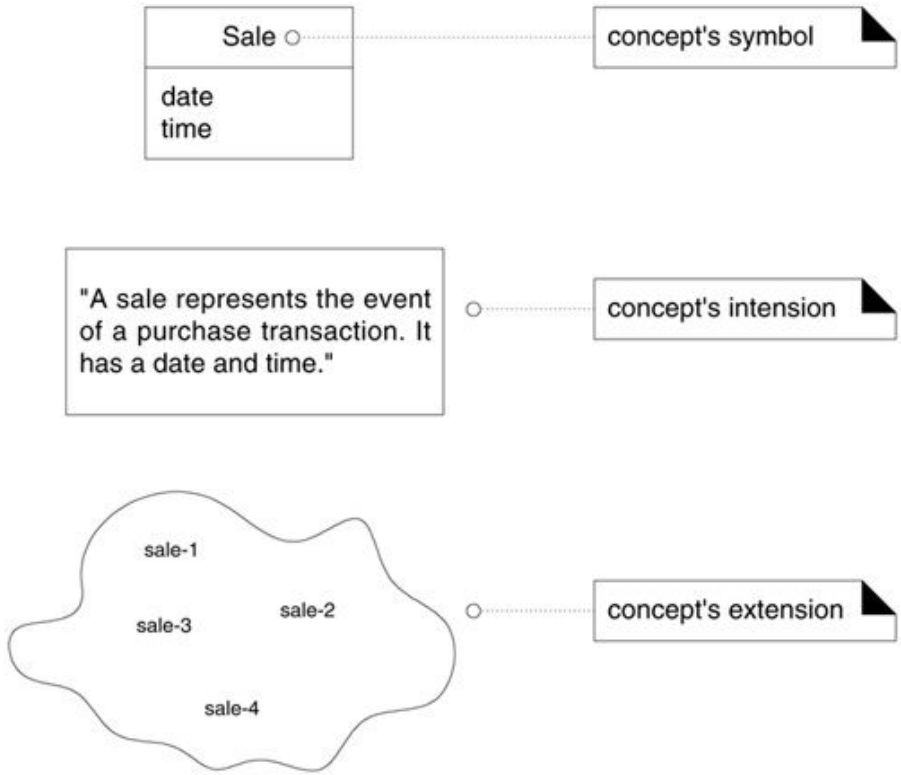


Figure: *A conceptual class has a symbol, intension, and extension.*

Why Create a Domain Model?

- Domain model to help coding?
- Domain model to help understanding?
- Domain model to help understanding and more
  - To understand the key concepts and vocabulary in a domain
  - To support a lower gap between the software representation and our mental model of the domain.

Why Create a Domain Model? (contd.)

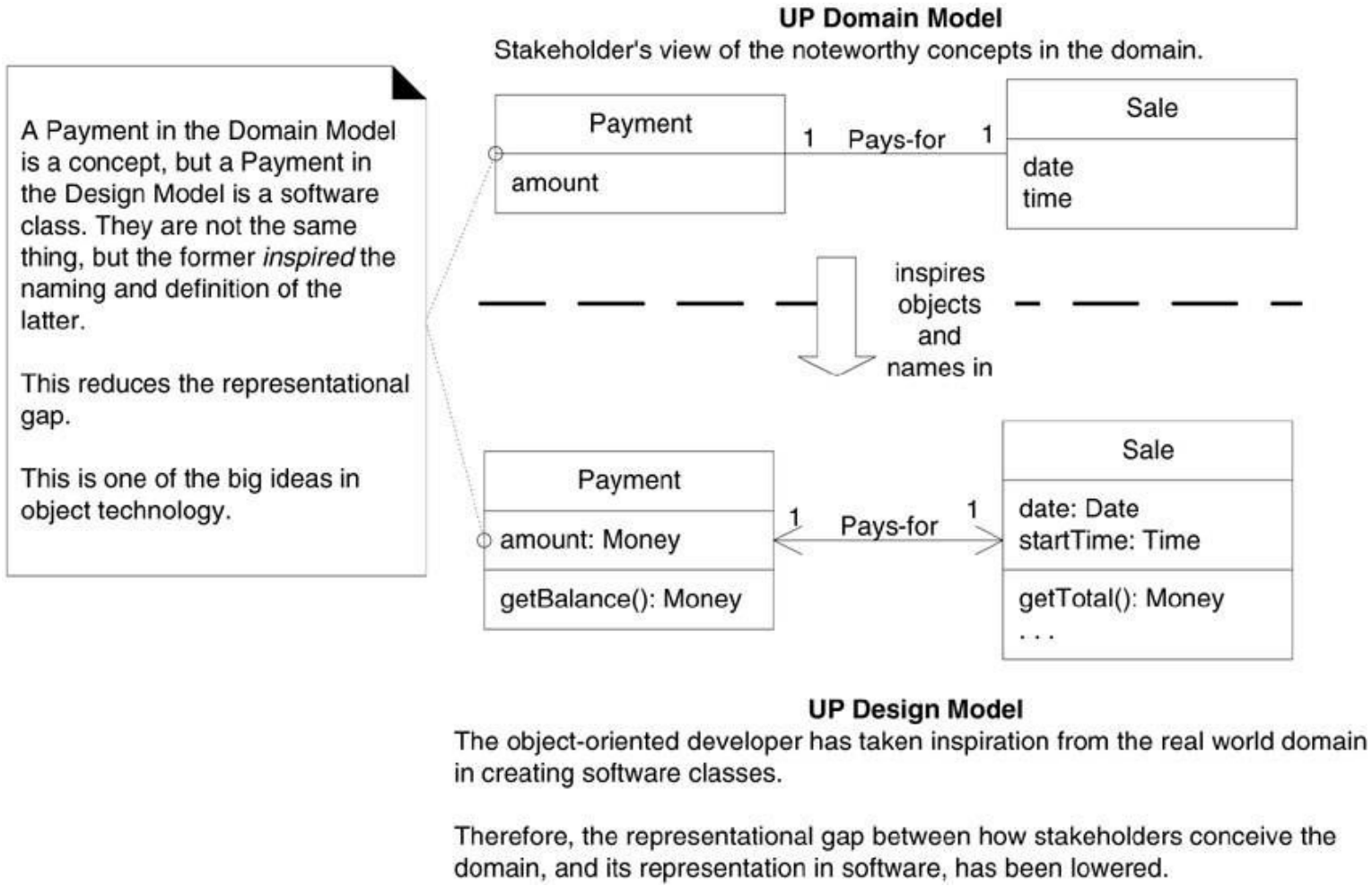


Figure: *Lower representational gap with OO modeling.*

Guideline: How to Create a Domain Model?

- Bounded by the current iteration requirements under design:
  - Find the conceptual classes
  - Draw them as classes in a UML class diagram.
  - Add associations and attributes.

Find the conceptual classes

Guideline: How to Find Conceptual Classes?

- Three Strategies to find conceptual classes
  - Reuse or modify existing models.
  - Use a category list.
  - Identify noun phrases.

Guideline: How to Find Conceptual Classes—Use a Category List

Table 9.1. Conceptual Class 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.	*SalesLineItem*
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 MonopolyPlayer 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 Flight*

Guideline: How to Find Conceptual Classes—Use a Category List (contd.)

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*

Guideline: How to Find Conceptual Classes—Noun Phrase Identification

- Identify the **nouns and noun phrases** in textual descriptions of a domain, and consider them as candidate conceptual classes or attributes.

- The **fully dressed use cases** are an excellent description to draw from for this analysis.

Guideline: How to Find Conceptual Classes—Noun Phrase Identification (contd.)

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).

Extensions (or Alternative Flows):

. . .

Guideline: How to Find Conceptual Classes—Noun Phrase Identification (contd.)

- Where are those terms found?
  - Some are in the use cases.
  - Others are in other documents, or the minds of experts.
- In any event, **use cases** are one rich source to mine for noun phrase identification.
- Some of these noun phrases are candidate conceptual classes
  - Some may refer to conceptual classes that are ignored in this iteration (for example, "Accounting" and "commissions"), and
  - Some may be simply attributes of conceptual classes.

Guideline: How to Find Conceptual Classes—Noun Phrase Identification (contd.)

- A **weakness** of this approach is the imprecision of natural language;
- Different noun phrases may represent the same conceptual class or attribute, among other ambiguities.
- Nevertheless, it is recommended in combination with the Conceptual Class Category List technique.

Example: Find and Draw Conceptual Classes: Case Study: POS Domain

Example: Case Study: POS Domain (contd.)

- Is this list correct?
- The good news/or bad news is
  - There is no such thing as a "correct" list.
  - It is a somewhat "arbitrary" collection of abstractions and domain vocabulary that the modelers consider noteworthy.
  - Nevertheless, by following the identification strategies, different modelers will produce similar lists.

Guideline: Agile Modeling Maintain the Model in a Tool?

- It's normal to miss significant conceptual classes during early domain modeling, and to discover them later during design sketching or programming.
- If you are taking an agile modeling approach, the purpose of creating a domain model is to quickly understand and communicate a rough approximation of the key concepts.
- Perfection is not the goal, and agile models are usually discarded shortly after creation (although if you've used a whiteboard, I recommend taking a digital snapshot).

Guideline: Report Objects - Include 'Receipt' in the Model?

- Are we missing something in the figure on domain model
- Should we include "Receipt" in domain model?
  - **No**. In general, showing a report of other information in a domain model is not useful since all its information is derived or duplicated from other sources. This is a reason to exclude it.
  - **Yes**. On the other hand, it has a special role in terms of the business rules: It usually confers the right to the bearer of the (paper) receipt to return bought items. This is a reason to show it in the model.
- The decision:
  - Since item returns are not being considered in this iteration, Receipt will be excluded.
  - During the iteration that tackles the Handle Returns use case, we would be justified to include it.

Guideline: Think Like a Mapmaker; Use Domain Terms

- 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.
- Do not add things that are not there.

Guideline: A Common Mistake with Attributes vs. Classes

- Perhaps the most common mistake when creating a domain model is to represent something as an attribute when it should have been a conceptual class.

- A rule of thumb to help prevent this mistake is:
  - **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**

Guideline: A Common Mistake with Attributes vs. Classes (contd.)

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.

Guideline: A Common Mistake with Attributes vs. Classes (contd.)

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.

Domain Model

- How to create a domain model?
- Bounded by the current iteration requirements under design:
  - Find the conceptual classes
  - Draw them as classes in a UML class diagram.
  - Add associations and attributes.

Associations and attributes

Associations

- An **association** is a relationship between classes that indicates some meaningful and interesting connection.

- Finding out association and visually representing them are part of your domain modeling effort.

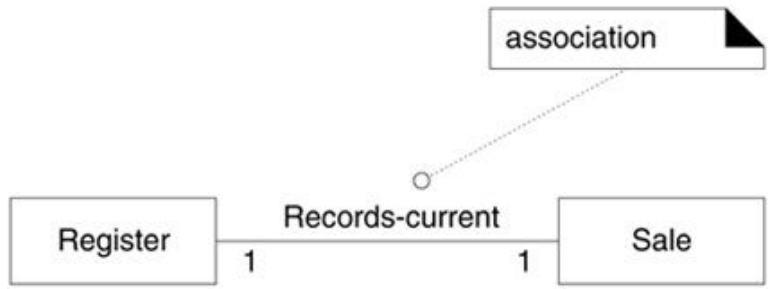


Figure: *Associations.*

- Avoid adding many associations.

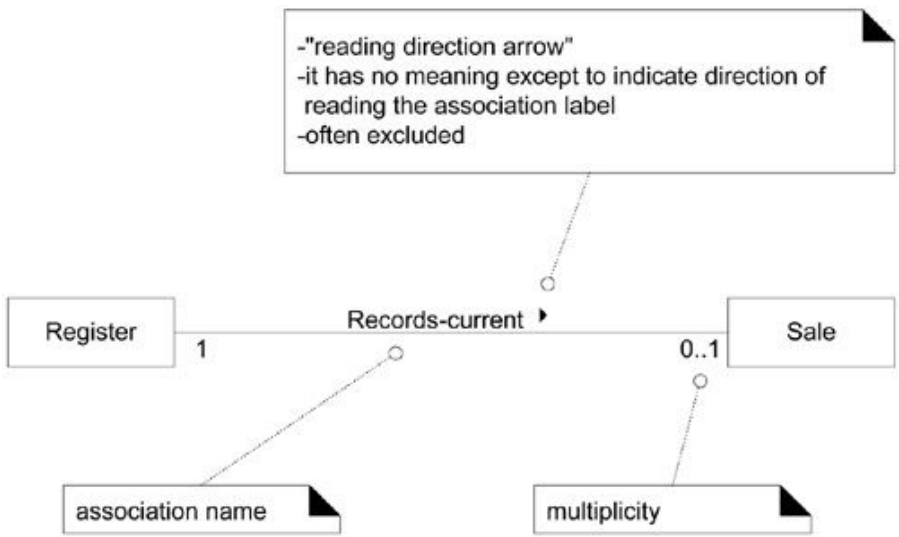
Associations

- During domain modeling, an association is **not** a statement about data flows, database foreign key relationships, instance variables, or object connections in a software solution;
- It is a statement that a relationship is **meaningful** in a purely conceptual perspective in the real domain.
- That said, many of these relationships will be implemented in software as paths of navigation and visibility.

Applying UML: Association Notation

- An association is represented as a line between classes with a capitalized association name.

- The ends of an association may contain a multiplicity expression indicating the numerical relationship between instances of the classes.



Guideline: How to Name an Association in UML?

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

  For example,

  - "Sale Paid-by CashPayment"
    - bad example (doesn't enhance meaning): Sale Uses CashPayment

Guideline: How to Name an Association in UML? (contd.)

- Association names should start with a **capital letter**, since an association represents a classifier of links between instances; in the UML, classifiers should start with a capital letter. Two common and equally legal formats for a compound association name are:
  - `Records-current`
  - `RecordsCurrent`

Applying UML: Roles

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

Applying UML: Multiplicity

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

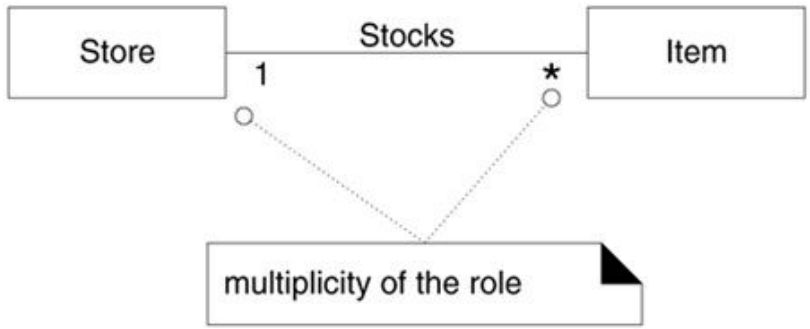
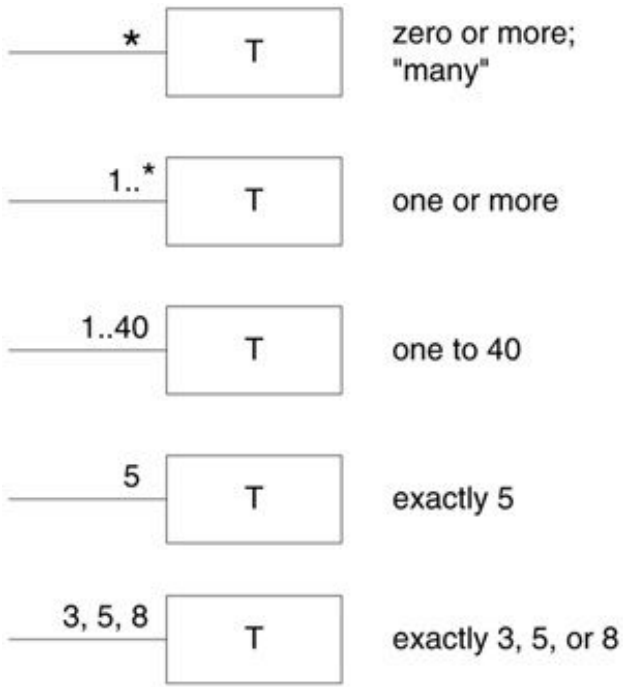


Figure: Multiplicity on an association.

For example,

a single instance of a `Store` can be associated with "many" (zero or more, indicated by the *) `Item` instances.

Applying UML: Multiplicity (contd.)



Applying UML: Multiplicity (contd.)

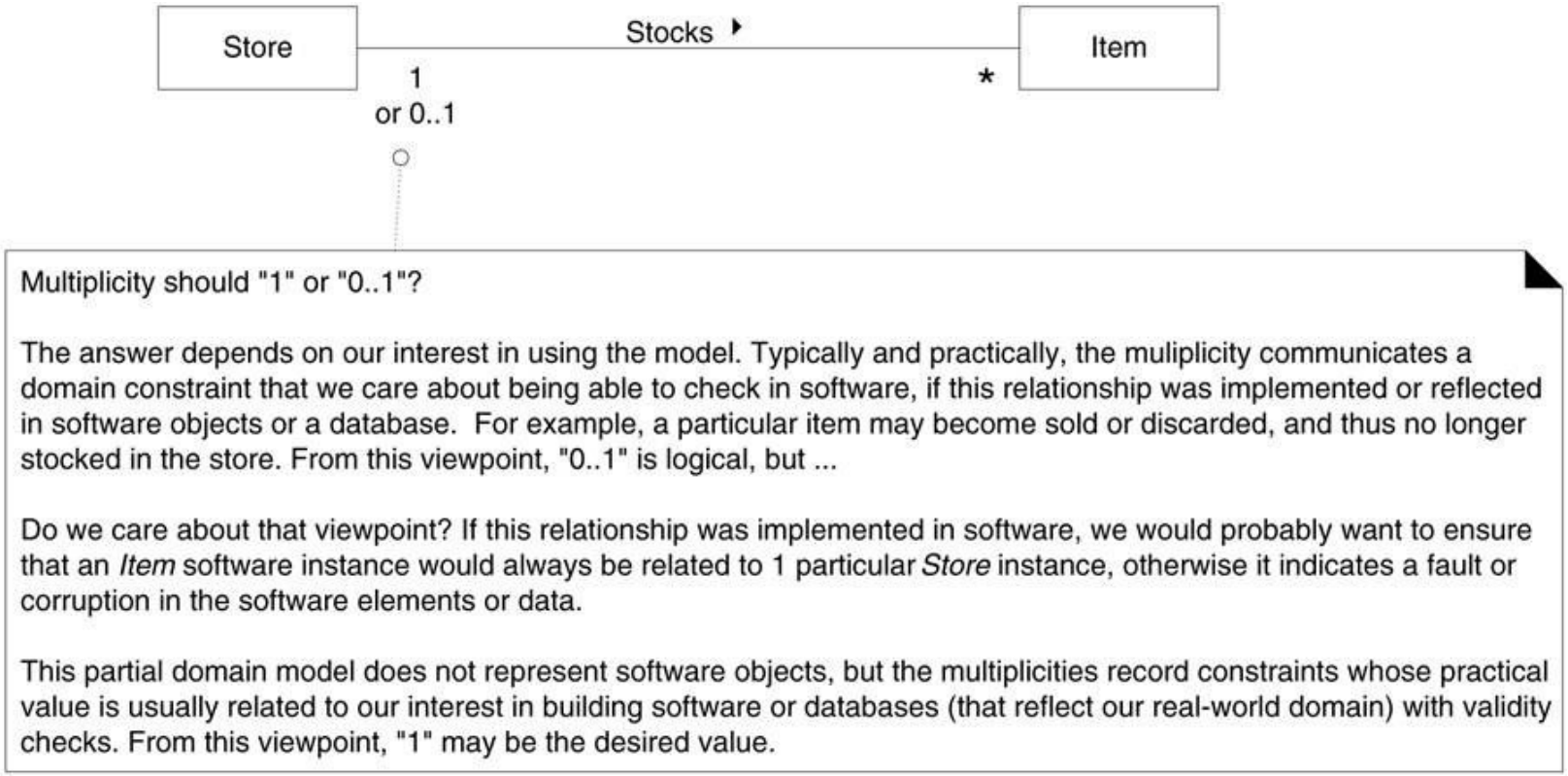


Figure: Multiplicity is context dependent.

Applying UML: Multiple Associations Between Two Classes

- Two classes may have multiple associations between them in a UML class diagram; this is not uncommon.

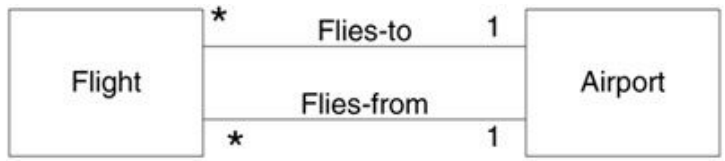


Figure: Multiple associations.

Example: Associations in the Domain Models Case Study: NextGen POS

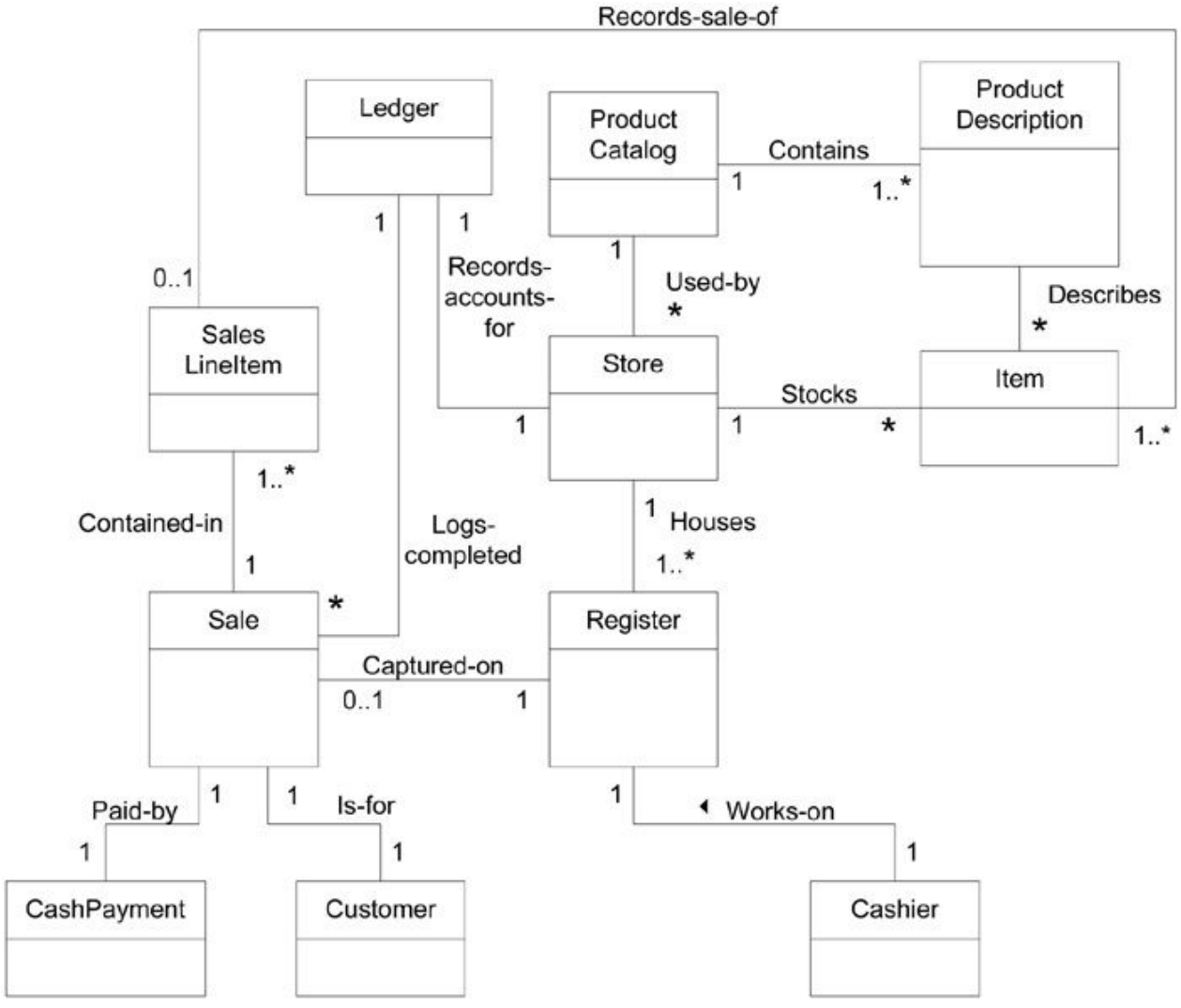


Figure: *NextGen POS partial domain model.*

Attributes

- Conceptual classes have been identified, drawn using UML. ✓
- Associations have been identified, drawn using UML. ✓
- It is now time to find out **attributes** for each conceptual class.
- An **attribute** is a logical data value of an object.
- So when do I know that I need an attribute for an conceptual class?

Guideline: When to Show Attributes?

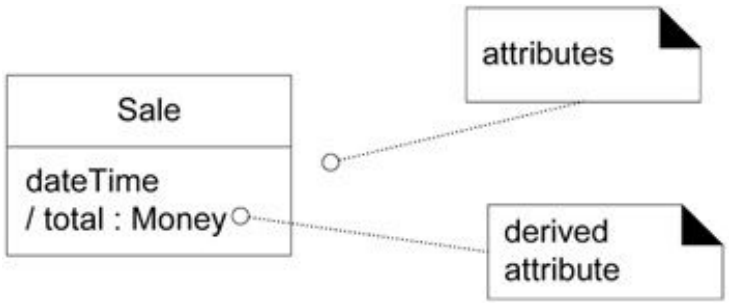
- Include attributes that the requirements (for example, use cases) suggest or imply a need to remember information.

  For example,

  a `receipt` (which reports the information of a `sale`) in the **Process Sale use case** normally includes a `date` and `time`, the `store name` and `address`, and the `cashier ID`, among many other things. Therefore,

  - `Sale` needs a `dateTime` attribute.
  - `Store` needs a `name` and `address`.
  - `Cashier` needs an `ID`.

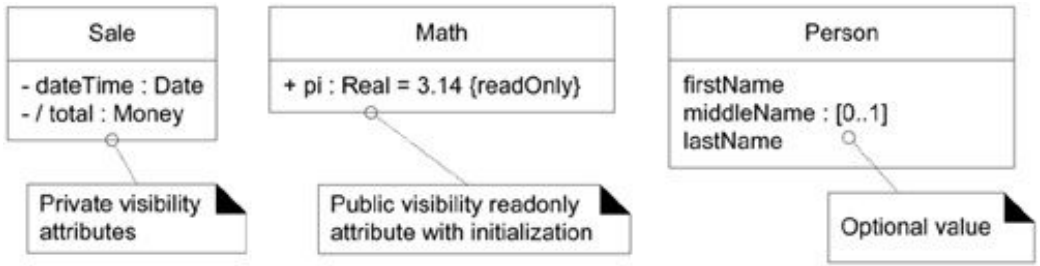
Applying UML: Attribute Notation



- Attributes are shown in the second compartment of the class box. Their type and other information may optionally be shown.

More Attribute Notation

- The full syntax for an attribute in the UML is: `visibility name : type multiplicity = default property-string`



- As a convention, most modelers will assume attributes have **private** visibility (-) unless shown otherwise
- {**readOnly**} is probably the most common property string for attributes.
- **Multiplicity** can be used to indicate the optional presence of a value, or the number of objects that can fill a (collection) attribute.

Attributes: Where to document attributes?

- Didn't we just record attributes in each class in UML format? So We have documented attributes in domain model, in those conceptual classes we drew?

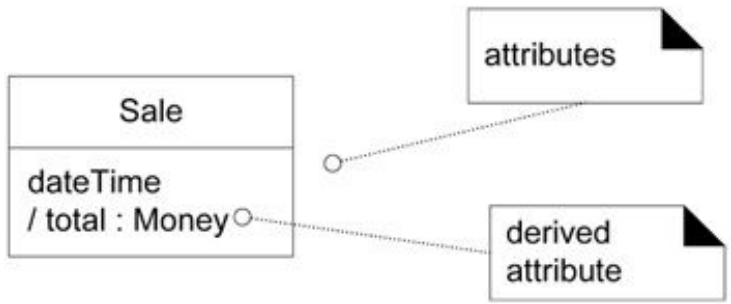
- Some modelers accept leaving such specifications only in the domain model, This is certainly one option or

- Place all such attribute requirements in the UP Glossary, which serves as a data dictionary.

Attributes: Derived Attribute

- The `total` attribute in the `Sale` can be calculated or derived from the information in the `SalesLineItems`.

- When we want to communicate that



  1. this is a noteworthy attribute, but
  2. it is derivable,

- We use the UML convention: a "/" symbol before the attribute name.

Attributes: Derived Attribute (contd.)

As another example,

a `cashier` can receive a group of like items (for example, six tofu packages),

- enter the `itemID` once, and
- then enter a `quantity` (for example, six).

Attributes: Derived Attribute (contd.)

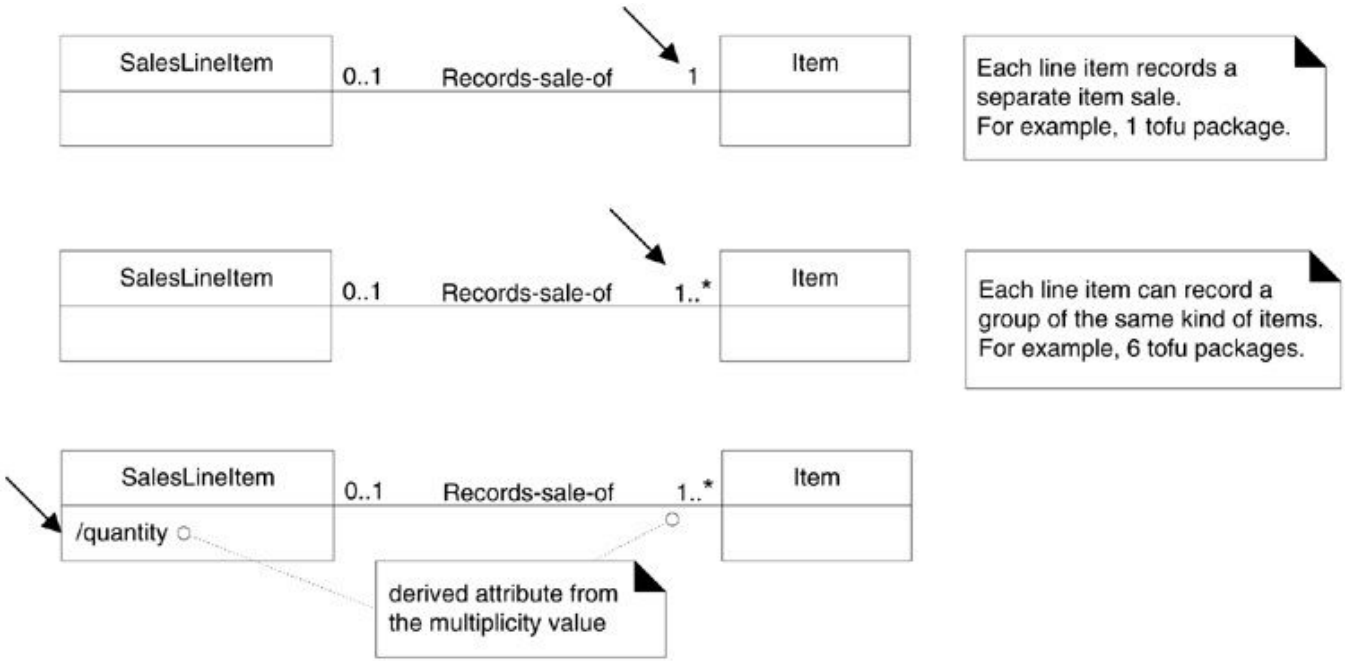


Figure: *Recording the quantity of items sold in a line item.*

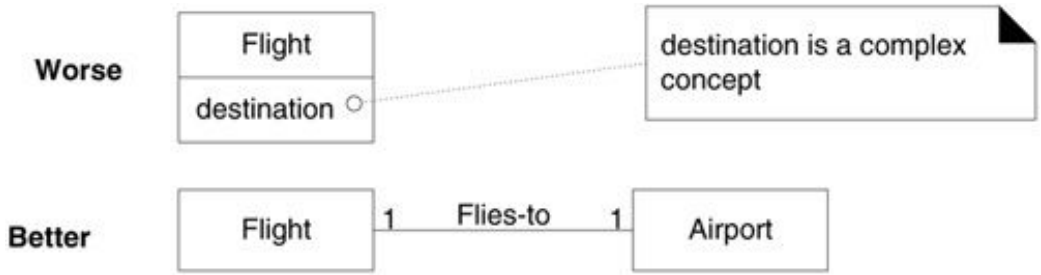
Guideline: What are Suitable Attribute Types?

- The attributes in a domain model should preferably be data types. Very common data types include: Boolean, Date (or DateTime), Number, Character, String (Text), Time.

- Other common types include: Address, Color, Geometrics (Point, Rectangle), Phone Number, Social Security Number, Universal Product Code (UPC), SKU, ZIP or postal codes, enumerated types.

Guideline: What are Suitable Attribute Types? (contd.)

- Relate conceptual classes with an association, not with an attribute.



Guideline: When to Define New Data Type Classes?

- Represent what may initially be considered a number or string as a new data type class in the domain model if:
  - 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 date
  - It is a quantity with a unit.
    - payment amount has a unit of currency
  - It is an abstraction of one or more types with some of these qualities.
    - item identifier in the sales domain is a generalization of types such as Universal Product Code (UPC) and European Article Number (EAN)

Example: Attributes in the Domain Models Case Study: NextGen POS

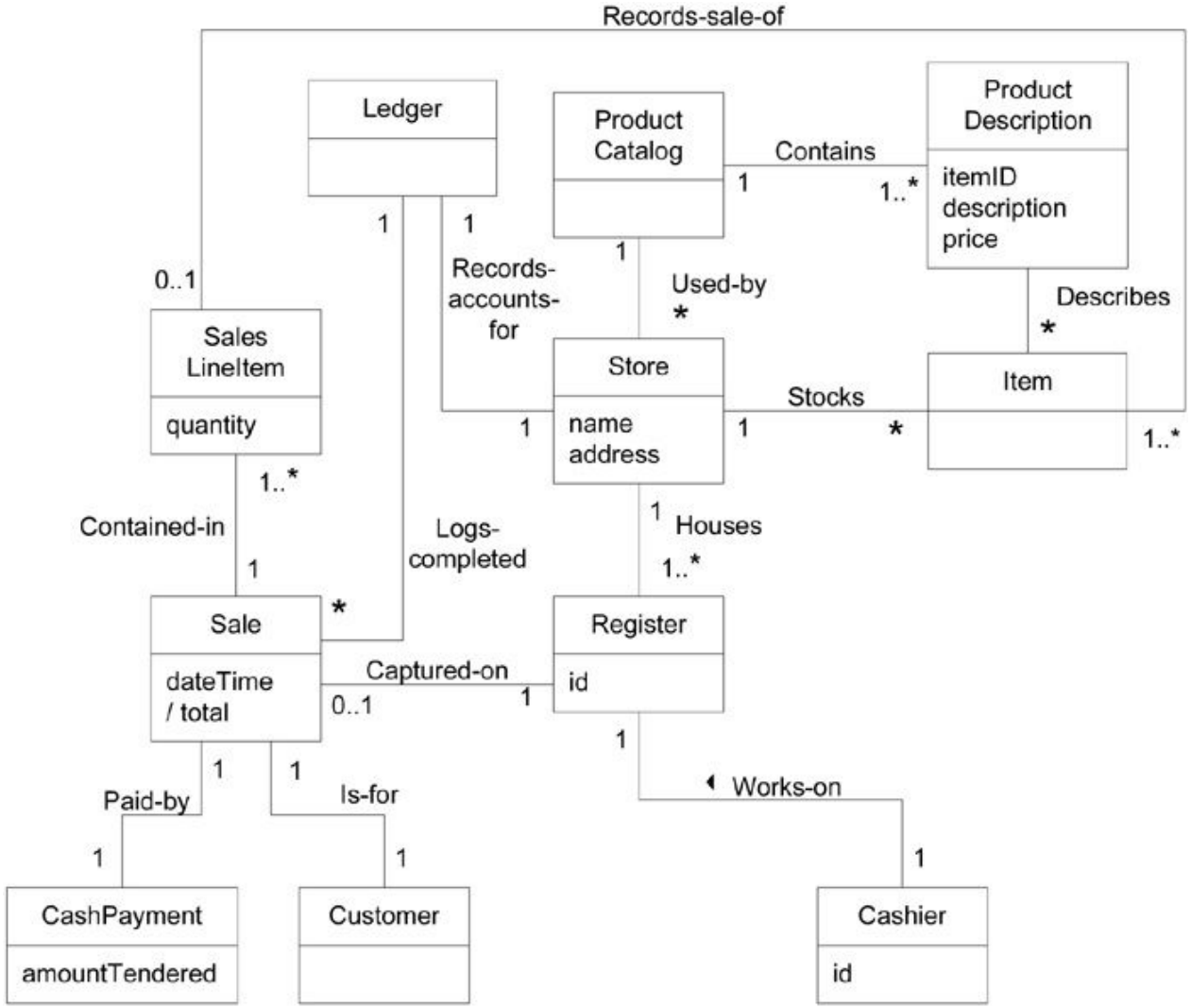


Figure: *NextGen POS partial domain model.*

Is the Domain Model Correct?

- There is no such thing as a single correct domain model.
- All models are approximations of the domain we are attempting to understand;
- The domain model is primarily a tool of understanding and communication among a particular group.
- A useful domain model captures the essential abstractions and information required to understand the domain in the context of the current requirements, and aids people in understanding the domain - its concepts, terminology, and relationships.

Guideline: Iterative and Evolutionary Domain Modeling

- Avoid a waterfall-mindset big-modeling effort to make a thorough or "correct" domain model—it won't ever be either, and such over-modeling efforts lead to analysis paralysis, with little or no return on the investment.

- Limit domain modeling to no more than a few hours per iteration.

Domain Models Within the UP

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	

It's Quiz Time

1. Domain models have also been called conceptual models (True or False)

2. The ends of an association may contain a multiplicity expression indicating the numerical relationship between instances of the classes. (True or False)

3. In ..... model, we use the UML convention: a / symbol before the derived attribute name.