Domain Modeling

Topic # 7

Chapter 9 – Craig Larman

What is a Domain Model?

- ***Problem domain :*** *area (**scope**)of application that needed to be investigated to solve the problem*
- ***Domain Model*** *: Illustrates **meaningful conceptual objects** in problem domain.*
- Represents real-world concepts, not software components
- Software-oriented class diagrams will be developed later, during design
- Bridges the representational gap between Use-Case Model and Software Design Model (such as Class and Interaction Diagrams)

Domain Modelling

- The end goal of object-oriented analysis and design is the construction of the classes that will be used to implement the desired software system.
- Domain modeling is a first step in that direction.
- A domain is a collection of related concepts, relationships, and workflows.
- Examples of domains include: science, engineering, medicine, business, government, military

Domain Model Representation

- Captures the most important types of objects in a system.
- *A domain model is a visual representation of **real world concepts** (real-situation objects), that could be : **idea, thing , event** or **object**…..etc .*
  - ***Business objects*** - represent things that are manipulated in the business e.g. ***Order***.
  - ***Real world objects*** – things that the business *keeps track of* e.g. ***Contact , book***.
  - ***Events*** *that come to light* - e.g. ***sale***.

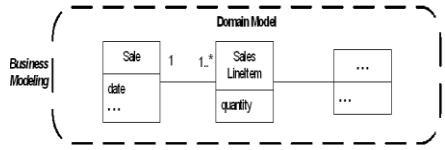
Domain Model Representation

Domain Model may contain :

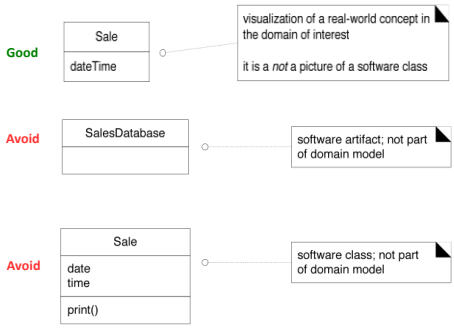
- Domain **objects** (conceptual classes)
  - **Attributes** of domain objects
- **Associations** between domain objects
  - **Multiplicity**

Domain Model - UML Notation

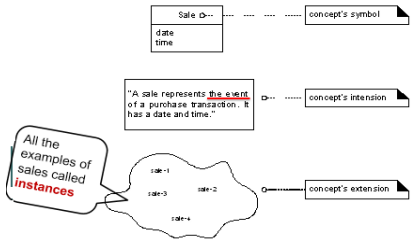
- Illustrated using a set of domain objects (conceptual classes) ***with no operations*** *(no **responsibility** assigned yet , **this will be assigned during design**).*



Domain Model is conceptual not a software artifact

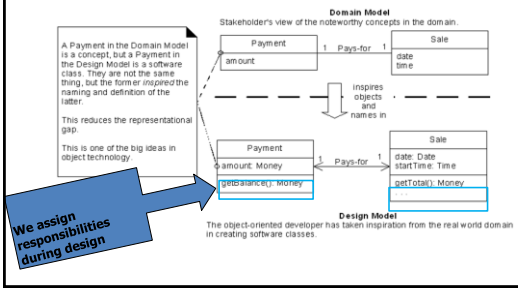


Symbol, Intension and Extension.



Why Create Domain model?

Answer : Get inspiration to create software classes



Characteristics of Domain Modeling

- Visual representation of conceptual classes.
- Associations and relationships between concepts (e.g Payment PAYS-FOR Sales).
- Attributes for information content (e.g. Sale records DATE and TIME).
- Does not include operations / functions.
- Does not describe software classes.
- Does not describe software responsibilities.

How to Create Domain Model

- Find conceptual classes
- Draw them as classes in a UML class diagram
- Add associations and attributes

Finding Conceptual Classes

- Reuse or modify the existing model if one exists
  - There are many published, well crafted domain models.
- Use a category list
  - Make a list of all candidate conceptual classes
- Identify noun phrases in your use-cases

Conceptual Class Category List

- Physical or tangible objects
  - Register, Airplane
- Specifications, or descriptions of things
  - ProductSpecification, FlightDescription
- Places
  - Store, Airport
- Transactions
  - Sale, Payment, Reservation
- Transaction items
  - SalesLineItem
- Roles of people
  - Cashier, Pilot
- Containers of other things
  - Store, Hangar, Airplane
- Things in a container
  - Item, Passenger
- Computer or electro mechanical systems
  - CreditPaymentAuthorizationSystem, AirTrafficControl
- Catalogs
  - ProductCatalog, PartsCatalog
- Organizations
  - SalesDepartment, Airline

Steps To Create A Domain Model

1. Create User Stories
2. Identify candidate conceptual classes
3. Draw them in a UML domain model
4. Add associations necessary to record the relationships that must be retained
5. Add attributes necessary for information to be preserved
6. Use existing names for things, the vocabulary of the domain

User Stories

- A User Story is one or more sentences in the everyday or business language of the end user or user of a system that captures what a user does or needs to do as part of his or her job function.
- It captures the 'who', 'what' and 'why' of a requirement in a simple, concise way, often limited in detail by what can be hand-written on a small paper note card.
- User stories are the descriptions of the domain that could be :
  - The problem definition.
  - The Scope.
  - The vision.

Identify Objects: Noun Phrase Identification

- *Identify Nouns and Noun Phrases in textual descriptions of the domain.*
- *However, Words may be ambiguous (such as : System)*
- *Different phrases may represent the same concepts.*
- *Noun phrases may also be attributes or parameters rather than classes:*
  - *If it stores state information or it has multiple behaviors, then it's a class*
  - *If it's just a number or a string, then it's probably an attribute*

Noun Phrase Identification

- Consider the following problem description, analyzed for Subjects, Verbs, Objects:

The ATM verifies whether the customer's card number and PIN are correct.
S V O O O
If it is, then the customer can check the account balance, deposit cash, and withdraw cash.
S V O V O V O
Checking the balance simply displays the account balance.
S O V O
Depositing asks the customer to enter the amount, then updates the account balance.
S V O V O V O
Withdraw cash asks the customer for the amount to withdraw; if the account has enough cash,
S O V O O V S V O
the account balance is updated. The ATM prints the customer's account balance on a receipt.
O V S V O O

From Noun Phrases to Classes and attributes

Consider the following problem description, analyzed for Subjects, Verbs, Objects:

The ATM verifies whether the customer's card number and PIN are correct.
SC V RO O A OA
If it is, then the customer can check the account balance, deposit cash, and withdraw cash.
S R V O A V O A V OA
Checking the balance simply displays the account balance.
SM OA V OA
Depositing asks the customer to enter the amount, then updates the account balance.
SM V OR V OA V OA
Withdraw cash asks the customer for the amount to withdraw; if the account has enough cash,
SM OA V OR OA V SC V OA
the account balance is updated. The ATM prints the customer's account balance on a receipt.
OA V SC V OA O

Analyze each **subject** and **object** as follows:

- Does it represent a person performing an action? Then it's an actor, '**R**'.
- Is it also a verb (such as 'deposit')? Then it may be a method, '**M**'.
- Is it a simple value, such as 'color' (string) or 'money' (number)? Then it is probably an attribute, '**A**'.
- Which NPs are unmarked? Make it '**C**' for class.
- Verbs can also be classes, for example:
  Deposit is a class if it retains state information

Identification of conceptual classes

- *Identify candidate conceptual classes*
- *Go through them and :*
  - ***Exclude irrelevant features and duplications***
  - ***Do not add things that are outside the scope*** *(outside the application area of investigation)*

Attributes

- *A logical data value of an object.*
- *Imply a need to remember information.*
  - Sale needs a **dateTime** attributte
  - Store needs a **name** and **address**
  - Cashier needs an **ID**

A Common Mistake when modeling the domain- Classes or Attributes?

Rule

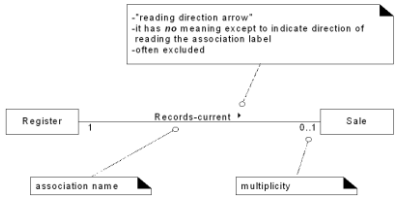
- ***If we do not think of a thing as a number or text in the real world, then it is probably a conceptual class.***
- ***If it takes up space, then it is likely a conceptual class.***

Examples:

- ***Is a store an attribute of a Sale ?***
- ***Is a destination an attribute of a flight ?***

Identifying Object Relationships- Associations

- Relationship between classes (more precisely, between instances of those classes)indicating some meaningful and interesting connection

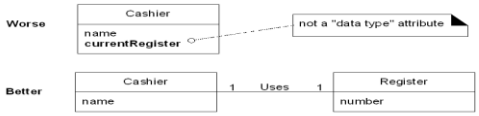


High priority association

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

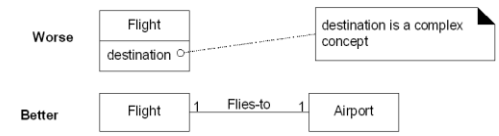
- To avoid:
  - *Avoid showing **redundant** or **derivable** associations*
  - ***Do not overwhelm the domain model with associations not strongly required***

Association or attribute ?



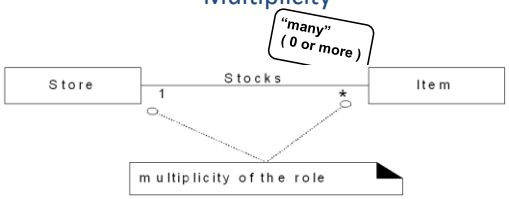
- *Most attribute type should be "**primitive**" data type, such as: numbers , string or boolean (true or false)*
- *Attribute should not be a complex domain concept(Sale , Airport)*
- *CurrentRegister is of type "Register", so expressed with an association*

Association or attribute ?



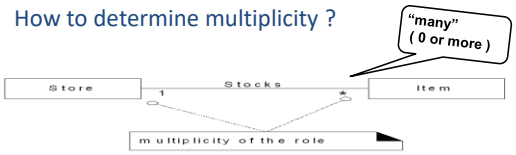
A destination airport is ***not a string***, it is a complex thing that occupies many square kilometers of space. So "Airport" should be related to "Flight" via an association , not with attribute

Multiplicity



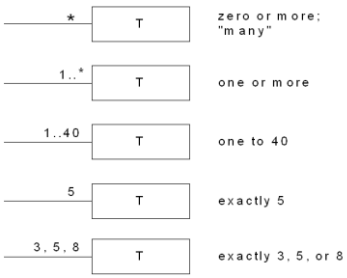
- ***Multiplicity*** *indicates how* ***many instances*** *can be validly associated with another instance, at* ***a particular moment****, rather than over a span of time.*

How to determine multiplicity ?



- ***Ask these 2 questions :***
  - ***1 store may stock how many item ?***
  - ***1 item may be stocked in how many stores ?***

Multiplicity



Identify Noun Phrase in Use Cases

Basic Flow for a POS System

1. **Customer** arrives at a **POS checkout** with **goods** and/or **services** to purchase
2. **Cashier** start a new **sale**
3. **Cashier** enters **item identifier**
4. System records **sale line item** and presents **item description**, **price**, and **running total**
5. …

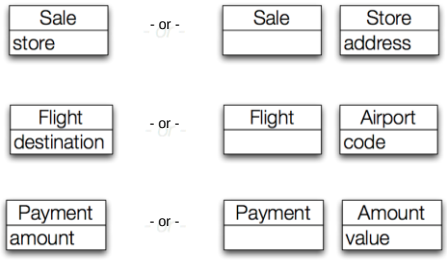
List and Draw UML Diagram

- Sale
- Cash Payment
- SaleLineItem
- Item
- Register
- Amount
- Cashier
- …

1. Decide which ones are classes and which ones are attributes
2. Add attributes and relation to the identified domain classes

Part of POS System

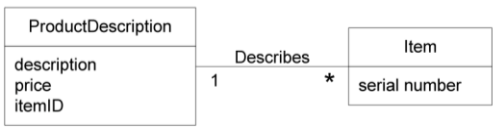
Attributes vs. Classes



Q3

Description Classes

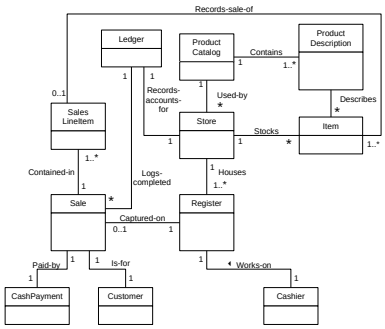
- A description class contains information that describes something else
- E.g. Item and ProductDescription



Point of Sale System (Continued)

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).

POS Domain Model



But remember

- *There is no such thing as a single correct domain model. All models* ***are approximations of the domain*** *we are attempting to understand.*
- *We* ***incrementally evolve a domain model*** *over several iterations on attempts to capture all possible conceptual classes and relationships.*

Book Reference and Reading

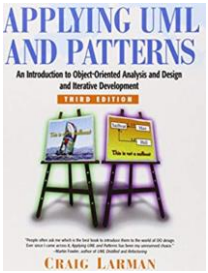
- Chapter 9 – Applying UML & Patterns – Craig Larman

Chapter 9. Domain Models

It's all very well in practice, but it will never work in theory.
anonymous management maxim

Objectives

- Identify conceptual classes related to the current iteration.
- Create an initial domain model.
- Model appropriate attributes and associations.



END OF TOPIC 7

-COMING UP!!!!!!
-Class Diagrams
- Midterm Examination 1

39