M3 - Domain Model, SSD and Operation contracts

17 September 2021 16:48

Elaboration is the initial series of iterations during which the team does serious investigation , implements the core architecture, clarifies most requirements and tackles the high risk issues.

Domain Model:

- Its classic model in OO Analysis
- Helps in designing some software objects
- A domain model is the visual representation of conceptual classes or real situation objects in domain.
- Domain model artefact created in business modelling discipline.
- A domain model does not show software artefacts or classes.

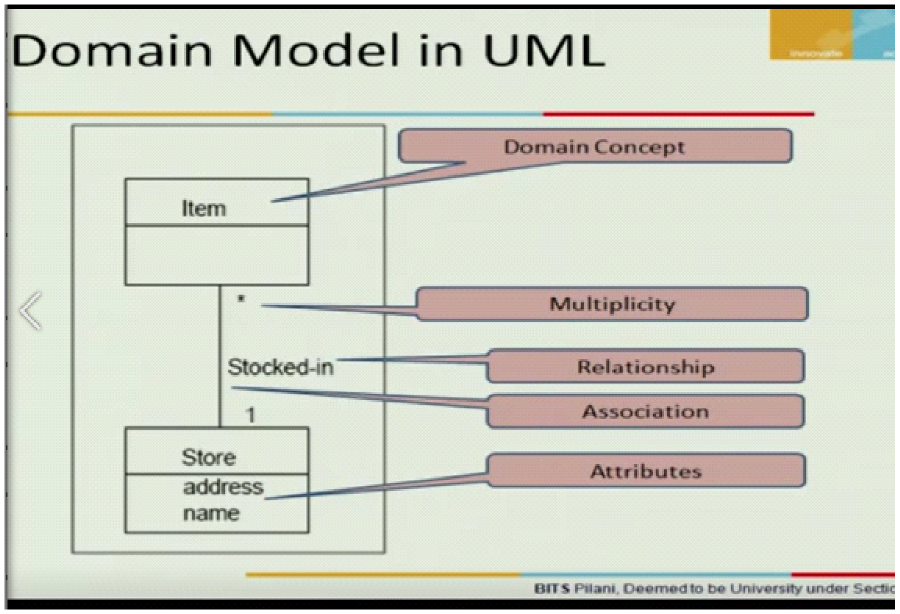
  Analyst has written fully dressed use cases,
- Use case diagram is behavioural diagram, it shows interaction between actors and the use cases.
- Domain model indicates relationship among domain concepts
- Not behavioural but static
- Entities: Bar code scanner, Bill, Payment.
- All domain concepts will be identified by analyst.
- It may show :
  - Concepts
  - Association between concepts
  - Attributes of concepts. (Bill date, Bill amount)
- In Domain we will include binary relationship (How two domains are related)
- It is different than ER diagram, For design we use class diagram.
- Objects are appearing in form of domain concepts.
- Useful for designer to draw artifact.

How to create domain model,

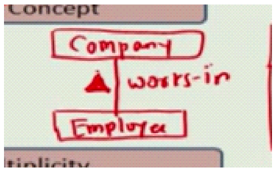
- Find conceptual class
- Draw them as classes in UML class diagram
- Add associations and attributes

1. Find conceptual classes
   a. Reuse or modify existing models
   b. Use a category list
   c. Identify noun phrases

How Domain model is represented in UML



- Item is the domain concept and represented in rectangular form without attributes.
- We can add attributes also, like item id, price of item.
- The line coming from one domain concept to other domain concept indicates association
- Association b/n two domain concepts indicates item and store are related to each other.
- Stocked in is the relationship between both domain concepts
- Domain concepts are appearing in noun and relationship is in verb
- By default reading direction is top to down / Left to right. (Items are stocked in Store)
- If there is direction it has to be like below.



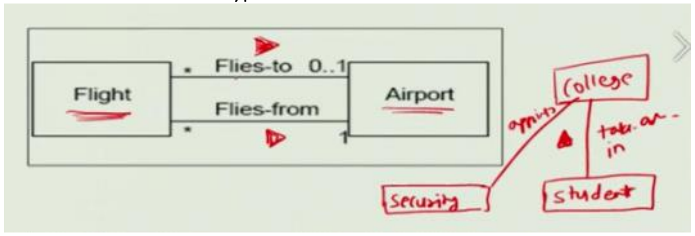
- Multiplicity / cardinality : how many relationships of one domain concept is related to other
- (*) means 0 or more,

Identification of Domain concepts from Use case

- All items are nouns (Item) or noun phrases (Item identifier)
- Identify conceptual classes
  - Use noun phrase identification
  - Get the nouns from fully dressed Use case text.
- Some of the nouns are domain concepts and some of them are attributes.
- Guideline: think like a map maker, user terminologies
- Guideline: 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.
- When to model with descriptor class ?
  - A description class contains information that describes something else.

Identification of Relationship among domain concepts.

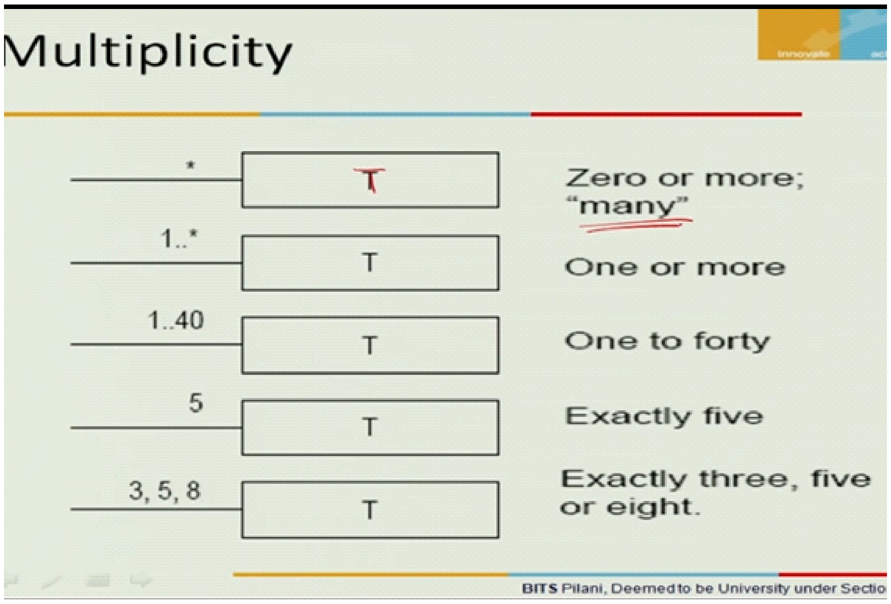
- A triangle is shown as direction
- Relationship is always bi directional
- Identify the verb for associations
- Store contains POS - association is contains which is verb
- Teacher guides student - association is guides which is verb.
- There can be multiple association between two types.



- There can be no relationship/association between two domain concepts.
- Guideline:
  - Associations worth noting usually imply knowledge of a relationship that needs to be preserved for some duration.
  - Between what objects do we need some memory of a relationship.
  - Ex: do we need to remember what salelineitem instances are associated with a sale instance ?
  - Avoid adding many associations

Finding Multiplicity / Cardinality:

- Multiplicity defines how many instances of type A can be associated with one instance of type B, at particular moment in time
- One to one
- One to many
- Many to one
- Many to Many



Multiplicity

*	T	Zero or more; "many"
1..*	T	One or more
1..40	T	One to forty
5	T	Exactly five
3, 5, 8	T	Exactly three, five or eight.

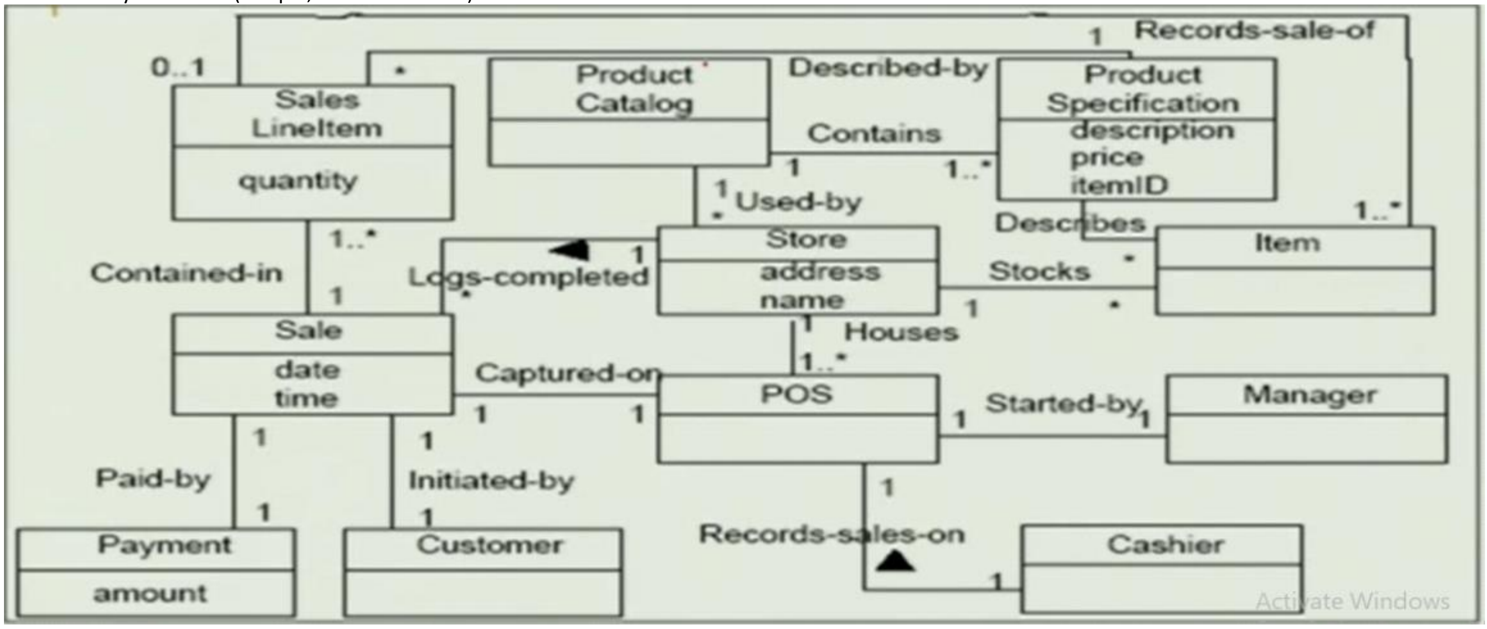
BITS Pilani, Deemed to be University under Sectio

Adding Attributes

- ➢ Logical data value of an object
- ➢ Don't think in programmers way
- ➢ Keep attributes simple, should not normally be a complex domain object
- ➢ Preferably be pure data values: bool, data, number, string
- ➢ Simple attributes like colour, phone number, zip code, universal product code.

Concluding Domain model (Steps for creating domain model)

1. Identification of domain (noun phrase analysis) avoiding attributes
2. Add associations (Relationships), Add directions
3. Multiplicity (Cardinality)
4. Identify attributes (simple, but not derived)



Significance of Domain Model:

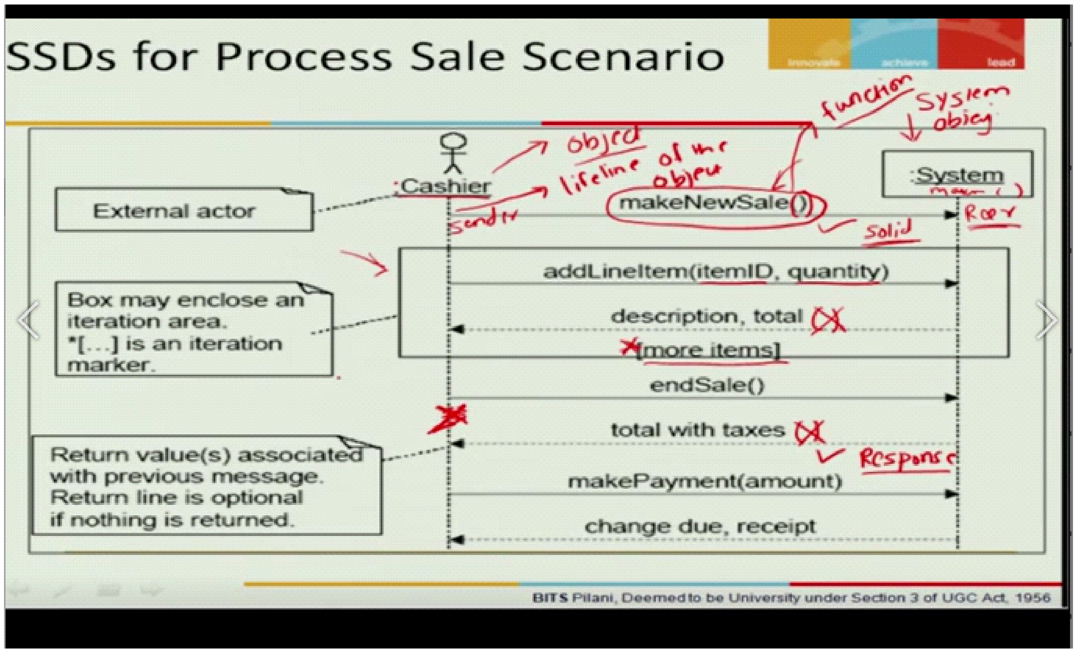
- ➢ Domain model is base for designer to draw class diagram
- ➢ Not necessarily all domain concepts will be carried forward.

System sequence diagrams

- Investigate and define the behaviour of software of black box.
- These are inputs to operation contracts and mostly object design.
- System is appearing like black box. We don't know internal details of it.
- SSD is drawn for a certain use case.
- External actors (cashier) interact with system box by means of sequence of messages
- Messages are to and from
- System behaviour is a description of what the system does (without explanation how it)
- How actors interact with the software system, During this actor generate events.
- Events such as.
    - Start a new sale. (MakeNewSale())

syntax:

:cashier : object representation in UML
Dotted lines under cashier and system is lifeline of the object. Cashier Is alive in the whole scenario
Cross (X) symbol is used to break the lifeline of object
Arrows: Message interaction between cashier, system.
Dotted arrow : Return values
Solid arrows : Requests
Message name : like a function makeNewSale()
***[more items]** : used in loops.


SSDs for Process Sale Scenario

External actor

Box may enclose an iteration area. *[...] is an iteration marker.

Return value(s) associated with previous message. Return line is optional if nothing is returned.

BITS Pilani, Deemed to be University under Section 3 of UGC Act, 1956

Significance of SSD:

- Plays key role in GUI design.
- Interaction of system with outside world.
- How system responses to external events.

Operation Contract

- Contract : It is an agreement between 2 parties
- There may be rules in the agreement and 2 parties follow that
- Operation contract may be defined for **system operations**, that the system as a black box component offers in its public interface.
- Operations : Every class contains operations.
- Why operation contracts :
    - Details missing from system sequence diagram.
    - What an analyst reflect in operation contract.
    - Will have logics in operation contract.
- For ex in SSD we only have a function makeNewSale(), where logic is missing, in operation contract we add logic
- It is an artifact which indicates what should be the operation of a function()
- Ex: MakeNewSale() contains sale objects, and AddSaleItem()

What is operation contract ?

- Operation contracts are documents that describe system behaviour
- OC may be defined for system operations
- The entire set of system operations across all use cases, defines the public system interface.
- Contracts are written for each system operation to describe its behaviour.

OC Representation in UML

Syntax:

1. Unique id of each operation contract

   Ex: Contract CO2: addLineItem
2. **Operation:** addLineItem(itemId: integer, quantity: integer) - operation signature
3. **Cross Reference:** Use cases: Process Sale. - Use case reference
4. **Pre conditions:** There is a sale underway. - Pre requisite.
5. **Post Conditions:** (In past tense)
   - **a.** A salesLineItem sli instance was created (instance creation)
   - **b.** Sli was associated with the sale. (association formed)
   - **c.** Sli.quantity was set to quantity (attribute modification)
   - **d.** Sli was associated with a productSpecification, based on itemId match (Association formed)

Writing operation contract for POS

Contract CO1: MakeNewSale
Operation: MakeNewSale()
Cross reference: Use case: Process sale
Pre-Condition: Cashier is authenticated and logged in.
Post Condition: (past tense) :

- A Sale sl was created (Instance creation)
- Sale date and time was updated with system date and time.

PostConditions:
The post conditions describe changes in the state of objects in the domain model, domain model state changes include instances created, associations formed or broken and attributes changed.

- Post conditions support fine grained detail and precision in declaring what the outcome of the operation must be.

How to create ?

- Identify system operations from the SSD.
- For complex system operations that are complex and perhaps subtle in their results, or which are not clear in the use case, construct a contract
- For post conditions, use following categories
  - Instance creation and deletion.
  - Attribute modification
  - Associations formed and broken.

GUIDELINES:

1. How to name system events
   - a. EnterItem(itemId) is better than Scan(ItemId)
2. SSD are part of use case model.
3. SSD are created during elobaration.