8 Week 27
July
Friday (190-176)

MODULE 2

Week	July 26	27	28	29	30
Monday		4	11	18	25
Tuesday		5	12	19	26
Wednesday		6	13	20	27
Thursday		7	14	21	28
Friday	1	8	15	22	29
Saturday	2	9	16	23	30
Sunday	3	10	17	24	31

Elaboration –

8.00 The primary purpose of this phase is to complete the most essential
8.30 parts of project that are high risk and plan construction phase. Common
9.00 risk factors are addressed to establish and validate the system
9.30 architecture. Processes undertaken during this phase include creation
10.00 of use case diagrams, conceptual diagrams (class diagrams) and package
10.30 diagrams.
11.00
11.30 The architecture is validated primarily through the implementation of
12.00 an executable architecture baseline. By the end of elaboration phase the
12.30 system architecture must have stabilized and executable architecture
13.00 baseline must demonstrate that architecture will support key system
13.30 functionality and exhibit the right behavior in terms of performance,
14.00 scalibility and cost.
14.30
15.00 The final deliverable of elaboration phase is a plan including cost and
15.30 schedule estimates for construction phase.
16.00
16.30 • Steps taken during this phase –
17.00 – Complete project plan with iterations planned and requirements defined.
17.30 – rank use case by priority and risk – define use cases
18.00 – project domain is defined
Evening – begin design & development of use case
– plan iterations for construction phase.

Meetings	✓	Things To Do	✓	Important Calls	✓
	☐		☐		☐
	☐		☐		☐
	☐		☐		☐

- Elaboration Artifacts -

8.00 → domain model: a visualization of domain concepts and entities. It is a no. of class diagrams describing the classes in the problem domain. It is an attempt to understand the problem domain by categorizing its elements.

10.30 → design model: it is a visualization of structure of the software to be created. It often includes
- Class diagrams - describing classes used to implement software, and their relationships.
- Sequence diagrams and collaboration diagrams - describing problem solving process.
- Statechart diagrams - showing how objects evolve (or change state) over time.

15.00 → Data model: Database schemes, including object to relational database mapping.

SUNDAY 10

- Outcomes of Elaboration phase -
- detailed software plan
- Updated risk management
- Management & Staffing plan
- A test plan
- A baseline vision
- Domain analysis model
- Software architecture description stating constraints & limitations
- Executable architectural baseline.

11 Week 28
July
Monday (189-176)

DOMAIN MODEL -

A domain model or Domain Objective Model (DOM) is a conceptual model created for a problem domain describing its various entities, their attributes and relationships, plus the constraints that govern the integrity of model elements comprising that problem domain. It can be effectively used to verify and validate the understanding of problem domain among various stakeholders. It is especially helpful as focusing point & communication tool.
In UML, a class diagram is used to represent domain model.

DATA MODELLING - it is the process of creating a data model for the data to be stored in database. It is conceptual representation of data objects, the associations between diff. data objects, and the rules.

The data model typically evolves thru 3 stages -

1) Conceptual: This stage involves the identification of high level key business and system entities and their relationship that define the scope of the problem to be addressed by the system. These entities are defined using modelling elements for business modeling included in Business Analysis Model.
2) Logical: This stage involves the refinement of the conceptual high level business and system entities into more detailed logical entities. These logical entities and their relationship can be optionally defined in a logical data model.
3) Physical: This stage involves the transformation of logical class designs into detailed and optimized physical database table designs. It also includes the mapping of database table designs to tablespaces & to the database component in the database storage design.

Meetings
Things To Do
Important Calls

Conceptual Classes -

It is defined as an idea, thing, or object. More formally, it may be considered in terms of its symbols, intension, and extension.

→ Strategies to find conceptual classes -

1. Reuse or modify existing models - Published, well-crafted domain & data models are modified into domain models.
2. Use a category list - A standard ~~category~~ conceptual class category table is used to create a list of candidate conceptual class which promotes creation of domain model.
3. Identify Noun Phrases - identify the nouns and noun phrases in textual descriptions of a domain, and consider them as candidate conceptual classes.

Description Classes -

It is defined as information that describes something else. For eg:, a product description that records the price, picture, and text description of an item.

Domain Model Refinement -

It is a process of refining the domain model with generalizations, specializations, associations classes, time intervals, composition & packages, conceptual class hierarchies, etc.

- In generalization process, the common characteristics of classes are combined to form a class in a higher level of hierarchy, i.e. subclasses are combined to form super class. Specialization is reverse process of generalization.

8.00 - By defining superclass (a conceptual superclass is more general than subclass; i.e. all members of subclass must be members of superclass), or subclass or abstract conceptual class (if every member of class C is also member of same subclass)

8.30 - If a class C can simultaneously have many values for the same kind of attribute A, create an association class with that attribute and associate it with C.

9.00 - define aggregation or composition by defining relationship among classes.

9.30 - Include packages and group elements by subject area, in same hierarchy, or in same use case or elements who have strong associations.

10.00

10.30

11.00

11.30

12.00

12.30

13.00

13.30

14.00

14.30

15.00 # ASSOCIATIONS -

15.30 it defines the relationship between two or more classes in the system. These generally relates to the one object having instance or reference of another.

16.00

16.30

17.00

17.30 in UML
Associations are implemented in 3 ways:

18.00

Evening 1) Multiplicity : It indicates the no. of instance of one class is linked to one instance of another class. (⟶)

2) Aggregation : it is a special form of association where all object have their own lifecycle but there is an ownership like parent & child. It implies a relationship where child can exist independently of the parent. It is "has-a" relationship.

(⟶◇)

Meetings

Things To Do

Important Calls

3) Composition : It is a special form of association. It is a strong type of aggregation. Here, the parent & child objects have co-incident lifetimes. Child object does not have its own lifecycle & if parent gets deleted, then all of its child object also gets deleted. (⟶)

ASSOCIATION	AGGREGATION	COMPOSITION.
1) it is denoted by ⟶	it is denoted by ⟶◇	it is denoted by ⟶◆
2) it can exist b/w two or more classes in UML	its part of association relationship	its part of association relationship.
3) There can be one-one, one-many, many-many association present b/w classes	it is considered as weak type of association	it is strong type of association
4) One or more objects can be associated with each other.	Objects that are associated with each other can remain in scope of system without each other.	objects that are associated with each other cannot remain in scope without each other.
5) Objects are linked with each other.	linked objects are not dependent upon each other.	objects are highly dependent on each other.

SUNDAY 17

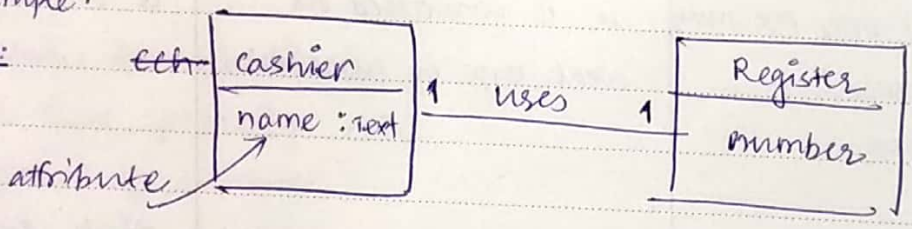
President's Day (Botswana)

6) Deleting one element may or may not affect another associated element.	Deleting one element does not affect another associated element.	Deleting one element affects another associated element.
7) eg: A teacher is associated with multiple students. or a teacher provides instruction to the students.	A car needs a wheel but does not need the very same wheel, it can function adequately with another wheel.	A file is placed inside the folder, if folder is deleted, all the files contained within are also gone.

ATTRIBUTES -

it is a logical data value of an object. In UML, they are shown in second component of class box. Attributes and its data type must be simple.

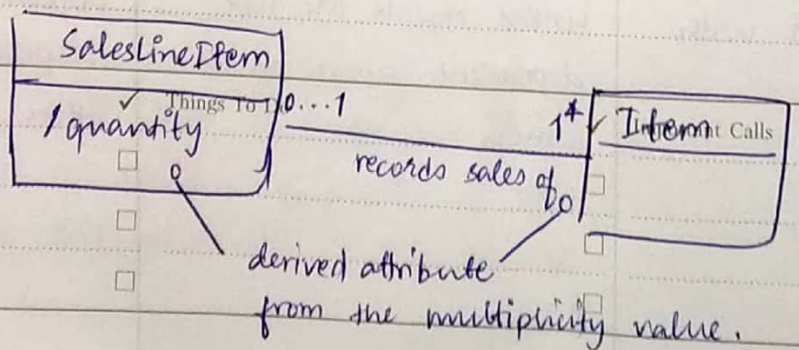
eg:



- Derived Attributes -

A quantity that can be calculated from other values, such as role multiplicities, is a derived attribute, designed in UML by leading slash symbol.

eg:



Meetings

Things To Do

Important Calls

ACTIVITY DIAGRAM -

It is a behavioral diagram, i.e. it depicts the behavior of the system. It potrays the control flow from a start point to finish point showing the various decision paths that exist while the activity is being executed. It is used to depict the dynamic aspects of the system.

- Notations -

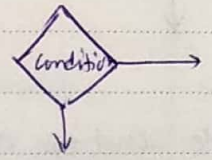
1). Initial State - the starting state before an activity takes place is depicted using the initial state. It is denoted by a black filled circle ' ● '

2) Action or activity state - it represents execution of actions on objects or by objects. It is represented using a rectangle with rounded corners.

' Activity. '

3) Action flow or control flow - they are also known as paths & edges. they are used to show the transition from one activity state to another. ' ⟶ '

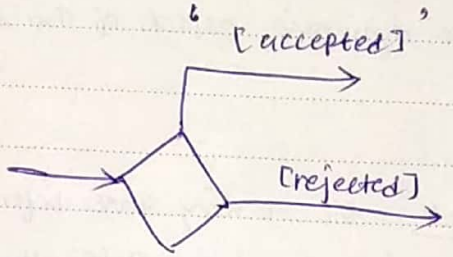
4) Decision node & Branching - When we need to make a decision before deciding the flow of control, we use decision node

'  '

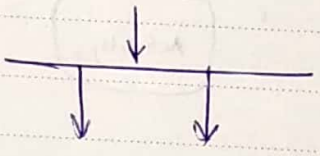
Meetings

✓	Things To Do	✓	Important Calls	✓
☐		☐		☐
☐		☐		☐
☐		☐		☐

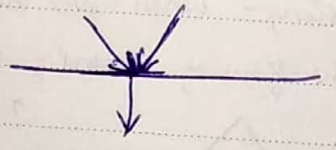
5) Guards – it refers to a statement written next to a decision node on an arrow sometimes with square brackets. This statement must be true for control to shift along a particular direction



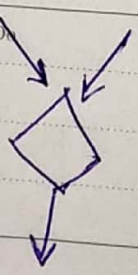
6) Fork – it is used to support concurrent activities, i.e. no decision is made before splitting the activity in two parts. Both parts need to be executed in fork statement



7) Join – join nodes are used to support concurrent activities converging into one. for join notations we have two or more incoming edges and one outgoing edge



8) Merge – Merge is used to combine activities that does not occur concurrently.

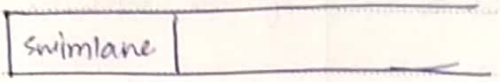


Meetings

✓ Things To Do

✓ Important Calls ✓

9) Swimlanes – it groups related activities into one column or one row. They are used to add modularity to the activity diagram. It's similar to creating a function in a program.



10) Time Event – We can have a scenario where an event takes some time to complete. We use an hourglass to represent time event. 

11) Final State or End State – it's the state which the system reaches when a particular process or activity ends. It is denoted by filled circle within a circle 

→ Uses of activity diagram:

- dynamic modelling of the system or process
- illustrates various steps involved in UML use case
- model software elements like methods, operations & functions
- to depict concurrent activities easily.
- show constraints, condition and logic behind algos.

#	USE CASE DIAGRAM	ACTIVITY DIAGRAM
1)	it is a model that represents the user's interaction with the system.	its a model that represents the series of actions or flow control in a system similar to flowchart.
2)	Helps to model the system, user interactions.	Helps to model the workflow of the system.

# SEQUENCE DIAGRAM	ACTIVITY DIAGRAM
1) it is used to visualize the sequence of calls in a system that is used to perform a specific functionality.	it is used to model the workflow of the system.
2) shows the msg flow from one object to another.	shows the msg flow from one activity to another.
3) it is used for dynamic modelling	it is used for functional modelling
4) it is used to describe the behavior of several objects in single use case.	it is used to describe general sequence of actions for several objects & use cases.
5) it is used to represent the time order of process	it is used to represent execution of process.

# GENERALIZATION	SPECIALIZATION
1) it proceeds in bottom up manner	top down manner
2) it reduces the size of schemas	increases the size of schemas.
3) it is applied on group of entities.	applied on single entity.
4) it extracts the common feature of multiple entities to form a new entity.	it splits an entity to form multiple new entities that inherit some feature of splitting entity
5) it results in forming a single entity from multiple entity	it results in forming multiple entity from single entity.