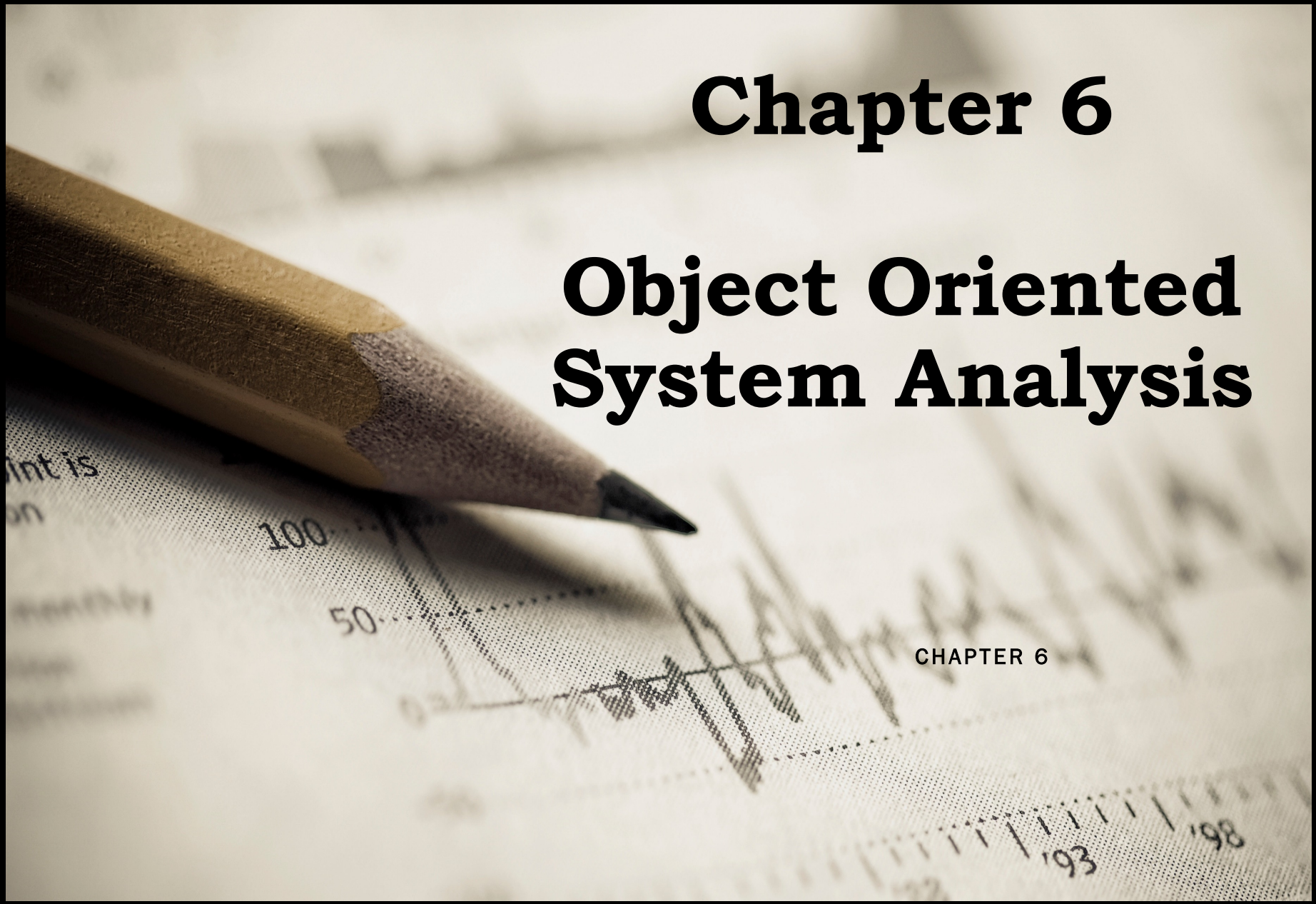


Chapter 6

Object Oriented System Analysis

CHAPTER 6

Contents

- What is object orientation?
- What is object oriented software engineering model
- Principles of Modeling
- Object Oriented Modeling
- Object-oriented perspective
- An overview of UML
- Things in the UML requirement
- Diagrams in the UML
- Use case Diagram

What is Object Orientation?

- **Object Orientation** :- organize software as a collection of discrete objects that incorporate both data structure and behavior.
- **The object-oriented approach,**
  - focuses on objects that represent abstract or concrete things in the real world.
  - These objects are first defined by their character and their properties, which are represented by their internal structure and their attributes (data).
  - The behavior of these objects is described by methods (functions).

…What is object orientation?…

- The OO model is beneficial in the following ways
  - It facilitates changes in the system at low cost.
  - It promotes the reuse of components.
  - It simplifies the problem of integrating components to configure large system.
  - It simplifies the design of distributed systems

....What is object orientation?...

- The **characteristics** of OO System
  - ✓ **Objects** is something that is exists within problem domain and can be identified by data (attribute) or behavior.
    - ✓ All tangible entities (student, patient) and some intangible entities (bank account) are modeled as object.
  - ✓ **Attributes** − They describe information about the object.

...What is object orientation?...

- ✓ **Class** – A class encapsulates the data and its behavior. Objects with similar meaning and purpose grouped together as class.
- ✓ **Methods** – Methods determine the behavior of a class. They are nothing more than an action that an object can perform.
- ✓ **Message** – A message is a function or procedure call from one object to another. They are information sent to objects to trigger methods..

What is object orientation?...

- Features of Object-Oriented System

a) Encapsulation (Information hiding)

- The process of binding both attributes and methods together within a class.
- Through encapsulation, the internal details of a class can be hidden from outside. It permits the elements of the class to be accessed from outside only through the interface provided by the class.
- It allows improvement or modification of methods used by objects without affecting other parts of a system.

What is object orientation?...

- Features of Object-Oriented System

b) Abstraction

- Abstraction lets us focus on essential aspects of an application while ignoring details i.e., focusing on what an object is and does, before deciding how to implement it.
- It's the most important skill required for OO development
- **Example:-** We utilize ATM machines to achieve different functionalities but when we put the card in the ATM, we have no idea what operations are happening within the ATM machine.

What is object orientation?...

- Features of Object-Oriented System

C) Relationships

- All the classes in the system are related with each other.
- The objects do not exist in isolation, they exist in relationship with other objects
- In order to describe a system, both dynamic (behavioral) and static (logical) specification of a system must be provided.
- The **dynamic specification** describes the relationships among objects e.g. message passing.
- And **static specification** describe the relationships among classes, e.g. aggregation, association, and inheritance.

What is object orientation?...

…C) Relationships

- There are three types of object relationships

1. Aggregation – It indicates relationship between a whole and its parts.

- **Example**

  - In the relationship, "**a car has–a motor**", car is the whole object or the aggregate, and the motor is a "part–of" the car.

What is object orientation?...

…C) Relationships

- There are three types of object relationships

1. **Association** –It is a group of links having common structure and common behavior. Association depicts the relationship between objects of one or more classes.

Degree of an Association

- Degree of an association denotes the number of classes involved in a connection. Degree may be unary, binary, or ternary.

- A **unary relationship** connects objects of the same class.
- A **binary relationship** connects objects of two classes.
- A **ternary relationship** connects objects of three or more classes.

What is object orientation?...

……C) Relationships

❑Generalization and Specialization

❑Generalization

- . Subclasses are combined to form a generalized super-class;
- It represents an "**is – a – kind – of"** relationship. For example, "car is a kind of land vehicle", or "ship is a kind of water vehicle".

❑Specialization

- Specialization is the reverse process of generalization.
- It can be said that the subclasses are the specialized versions of the super-class.

…What is object orientation?...

- Features of Object-Oriented System

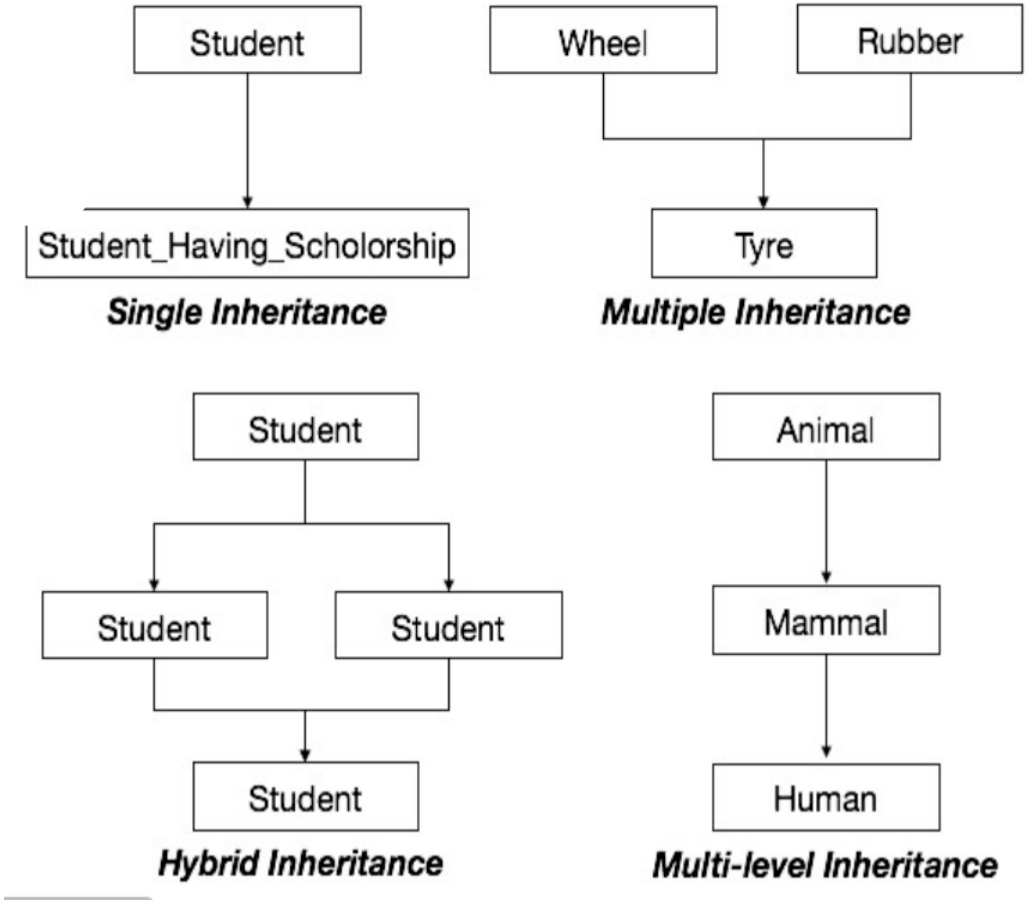
d) Inheritance

- It is a mechanism that permits new classes to be created out of existing classes by extending and refining its capabilities.
- The subclass can inherit or derive the attributes and methods of the super-class (es) provided that the super-class allows so.
- Besides, the subclass may add its own attributes and methods and may modify any of the super-class methods. Inheritance defines a **"is – a"** relationship.

…What is object orientation?…

❑……d) Inheritance

- Examples of different types of inheritance.



...What is object orientation?...

e) Polymorphism and Dynamic Binding

- The ability to take on many different forms.

- It applies to both objects and operations.
- A polymorphic object is one who true type hides within a super or parent class.
- In polymorphic operation, the operation may be carried out differently by different classes of objects.
- It allows us to manipulate objects of different classes by knowing only their common properties.
- Polymorphism is particularly effective while implementing inheritance.

Object Oriented Modelling

- A model is a simplification at some level of abstraction

- Object-oriented modeling and design is a way of thinking about problems using models organized around real world concepts.

- The fundamental construct is **the object**, which combines both data structure and behavior.

- **Purpose of Models:**
  - Testing a physical entity before building it
  - Communication with customers
  - Visualization
  - Reduction of complexity

…Object Oriented Modelling

Types of Models:

There are 3 types of models in the object oriented modeling and design : ***Class Model, State Model, and Interaction Model.***

- **Class Model (Object modelling) :-** The class model shows all the classes present in the system.
- The class model shows the attributes and the behavior associated with the objects.
- The class diagram is used to show the class model.

...Object Oriented Modeling

State Model (Dynamic modelling):-

- State model describes those aspects of objects concerned with time and the sequencing of operations – events that mark changes, states that define the context for events, and the organization of events and states.
- Actions and events in a state diagram become operations on objects in the class model.
- **State diagram** describes the state model.

...Object Oriented Modelling

Interaction Model (Functional modelling) :-

- Interaction model is used to show the various interactions between objects, how the objects collaborate to achieve the behavior of the system as a whole.

- The following diagrams are used to show the interaction model:
  - Use Case Diagram
  - Sequence Diagram
  - Activity Diagram

Principles of Modeling

- **The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped.**
  - Choose your models well. The right models will highlight the most nasty development problems. Wrong models will mislead us, causing us to focus on irrelevant issues.
- **Every model may be expressed at different levels of precision.**
  - Sometimes, a quick and simple executable model of the user interface is exactly what you need. At other times, you have to get down to complex details such as cross-system interfaces or networking issues etc.
  - In any case, the best kinds of models are those that let you choose your degree of detail, depending on who is viewing it. An analyst or an end user will want to focus on issues of what and a developer will want to focus on issues of how.

…Principles of Modeling…

- **The best models are connected to reality**
  - In software, the gap between the analysis model and the system's design model must be less. Failing to bridge this gap causes the system to diverge over time. In object-oriented systems, it is possible to connect all the nearly independent views of a system into one whole.
- **No single model is sufficient. Every non-trivial system is best approached through a small set of nearly independent models.**

Object oriented modeling

- **Object-oriented modeling (OOM)** is the construction of objects using a collection of objects that contain stored values of the instance variables found within an object.
- Unlike models that are record-oriented, object-oriented values are solely objects.
- The object-oriented modeling approach creates the union of the application and database development and transforms it into a unified data model and language environment.
- Object-oriented modeling allows for object identification and communication while supporting data abstraction, inheritance and encapsulation

Object oriented modeling…

- The software life cycle is typically divided up into stages going from abstract descriptions of the problem to designs then to code and testing and finally to deployment.
- Modeling is done at the beginning of the process.
- The reasons to model a system before writing the code are:

A) Communication: Users typically cannot understand programming language or code.

- Model diagrams can be more understandable and can allow users to give developers feedback on the appropriate structure of the system.
- A key goal of the Object-Oriented approach is to decrease the "semantic gap" between the system and the real world by using terminology that is the same as the functions that users perform

Object oriented modeling…

- …The reasons to model a system before writing the code are:

B) Abstraction: goal of most software methodologies is to first address "what" questions and then address "how" questions. i.e., first determine the functionality the system is to provide without consideration of implementation constraints and then consider how to take this abstract description and refine it into an implementable design and code given constraints such as technology and budget.

  - Modeling enables this by allowing abstract descriptions of processes and objects that define their essential structure and behavior.

- Object-oriented modeling is typically done via use case and abstract definitions of the most important objects.

- The most common language used to do object-oriented modeling is the Object management Group's Unified Modeling Language (UML)

Overview of UML

- UML was created by the Object Management Group (OMG) and UML 1.0 specification draft was proposed to the OMG in January 1997.
- It is stands for **Unified Modeling Language**.
- different from the other common programming languages such as C++, Java, COBOL, etc.

- A pictorial language used to make software blueprints.
- UML can be described as a general purpose visual modeling language to visualize, specify, construct, and document software system.
- Although UML is generally used to model software systems, it is not limited within this boundary. It is also used to model non-software systems as well. For example, the process flow in a manufacturing unit, etc.

Overview of UML…

Goals of UML

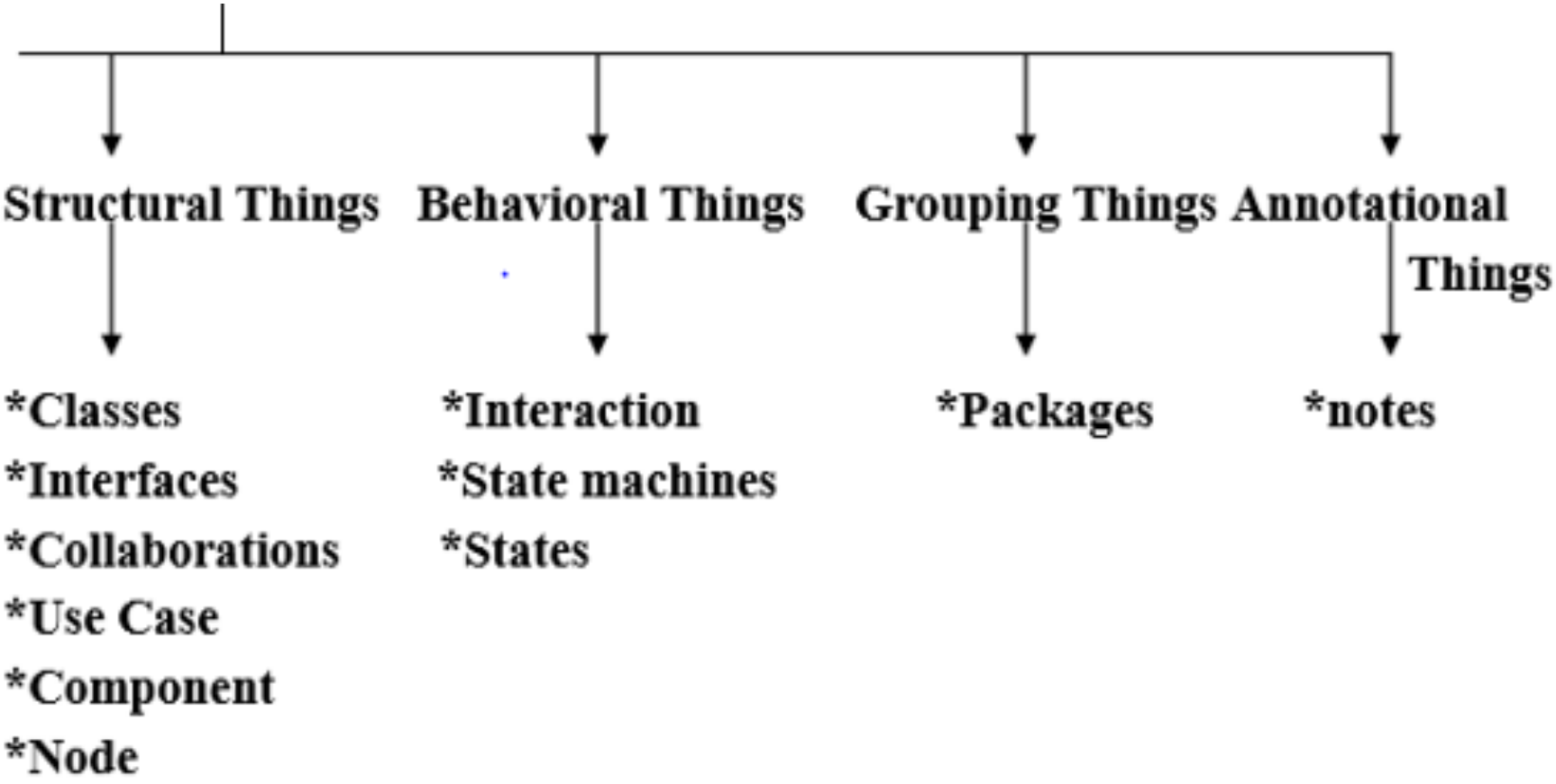
- Provide users with a ready-to-use, expressive visual modeling language so they can develop and exchange meaningful models
- Provide extensibility and specialization mechanisms to extend the core concepts
- Be independent of particular programming languages and development processes
- Provide a formal basis for understanding the modeling language
- Encourage the growth of the OO tools market
- Support higher-level development concepts such as collaborations, frameworks, patterns and components
- Integrate best practices

Overview of UML…

- **Building Blocks of the UML**
- The vocabulary of the UML encompasses three kinds of building blocks
- Building Blocks:
  - **Things**: abstractions, main concepts in a model
  - **Relationships**: tie the things together
  - **Diagrams:** group interesting collections of things

…Overview of UML…

- **Things**
  - Four kinds of Things in UML. These things are the basic object-oriented building blocks of the UML. We use them to write well-formed models.



…Overview of UML…

- Things in the UML
  - **Structural Things**
    - Static part of a model, conceptual or physical elements
    - nouns of UML Models
    - Collectively, the structural things are called classifiers.
  - **Behavioral Things**
    - dynamic parts of UML models
    - verb, representing behavior over time and space

Overview of UML…

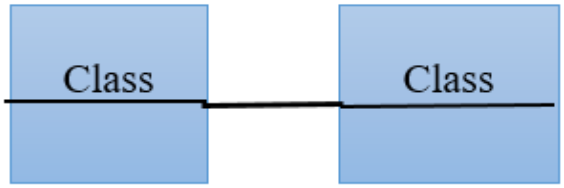
- **Grouping Things**
  - Organizational parts of UML models, These are the boxes into which a model can be decomposed. There is one primary kind of grouping thing, namely, packages.
- **Annotational Things**
  - Explanatory parts of UML models,
  - These are the comments we may apply to describe, light up, and remark about any element in a model.
  - There is one primary kind of annotational thing, called a note.

Overview of UML…

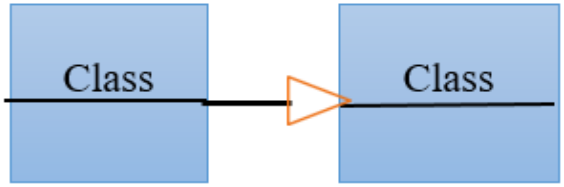
Relationships

- There are 4 kinds of relationships in the UML:
  - Dependency
  - Association
  - Generalization
  - Realization
- These relationships are the basic relational building blocks of the UML

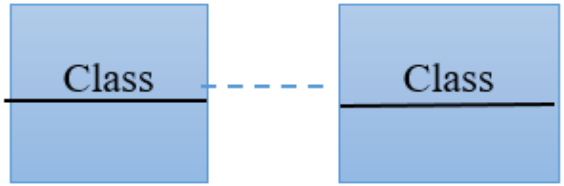
Overview of UML…



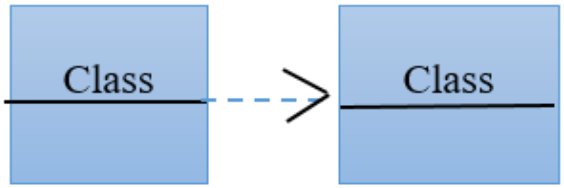
Association: - represented by a straight line connecting two classes. It simply demonstrates that the classes are aware of their relationship with each other.



Inheritance: - Indicates a 'child – parent' relationship between classes. The child class is a spatialized sub-class of the parent.

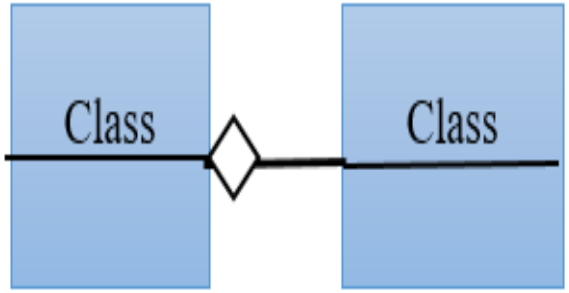


Realization: - One class implements the behaviour specified by other class.

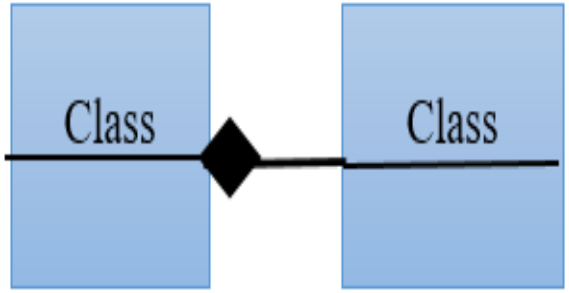


Dependency: - As the name suggests, one class depends on another. A dashed arrow shows this.

Overview of UML…



Aggregation: -this represents a unilateral relationship between classes. On class is part of or subordinate to another. In this instance, the parent and child classes can exist independently.



Composition: - It is a form of aggregation where on class is independent on another. One class is a part of the other. In this instance, the child classes and the patent classes cannot exist independently.

Overview of UML…

Diagrams

- It is the graphical presentation of a set of elements. It is rendered as a connected graph of vertices (things) and arcs (relationships).
- In theory, a diagram may contain any **combination of things and relationships**.
  - Class diagram , Object diagram , Use case diagram Sequence diagram Collaboration diagram, State chart diagram, Activity diagram Component diagram, Deployment diagram

Overview of UML…

❑Rules:

The UML has a number of rules that specify what a well-formed model should look like.

❑ A well-formed model is one that is semantically self-consistent and in harmony with all its related models.

❑The UML has semantic rules for:

- **Names** – What you can call things, relationships, and diagrams.
- **Scope** – The context that gives specific meaning to a name.
- **Visibility** – How those names can be seen and used by others.
- **Integrity** – How things properly and consistently relate to one another.

Use Case

What is a Use Case

- A formal way of representing **how a business system interacts with its environment**
- Illustrates the activities that are performed by the users of the system
- Use Cases describe scenarios that describe the interaction between users of the system (the actor) and the system itself.

➢ A scenario-based technique in the UML

➢ Functional requirement

➢ Verb + Noun, e.g., place order, display patents

….Use Cases …

- A **use case** is a summary of scenarios for a single task or goal.

- The emphasis is on *what* a system does rather than *how.*
- Use case diagrams are closely connected to scenarios.
- **Use case diagrams** describe what a system does from the standpoint of an external observer.
- A **scenario** is an example of what happens when someone interacts with the system.

….Use Cases …

- Here is a scenario for a medical clinic.

❑ *A patient calls the clinic to make an appointment for a yearly checkup. The receptionist finds the nearest empty time slot in the appointment book and schedules the appointment for that time slot. "*

- We want to write a use case for this scenario.

❑ Remember: A **use case** is a summary of scenarios for a single task or goal.

.....Use Cases ...

Step 1 Identify the actors

- Define those people or systems that are going to interact with the scenario.
- A *patient calls the clinic to make an appointment for a yearly checkup. The receptionist finds the nearest empty time slot in the appointment book and schedules the appointment for that time slot. "*

Questions for Identifying People Actors

- Who is interested in the scenario/system?
- Where in the organization is the scenario/system be used?
- Who will benefit from the use of the scenario/system?
- Who will supply the scenario/system with this information, use this information, and remove this information?
- Does one person play several different roles?
- Do several people play the same role?

..Use Cases …

….Questions for Identifying People Actors

- What other entity is interested in the scenario/system?
- What other entity will supply the scenario/system with this information, use this information, and remove this information?
- Does the system use an external resource?
- Does the system interact with a legacy system?

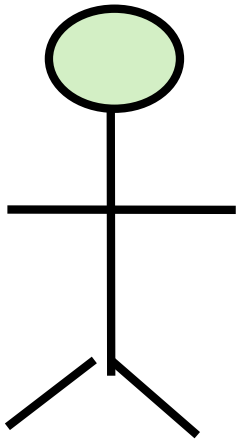
..Use Cases …

Actors

- A **user** that interacts with the system being designed in order to obtain some value from that interaction
- An **actor** is who or what initiates the events involved in the task of the use case. Actors are simply roles that people or objects play.

❑ **It can be a:**

- Human
- Peripheral device (hardware)
- External system or subsystem
- Time or time-based event

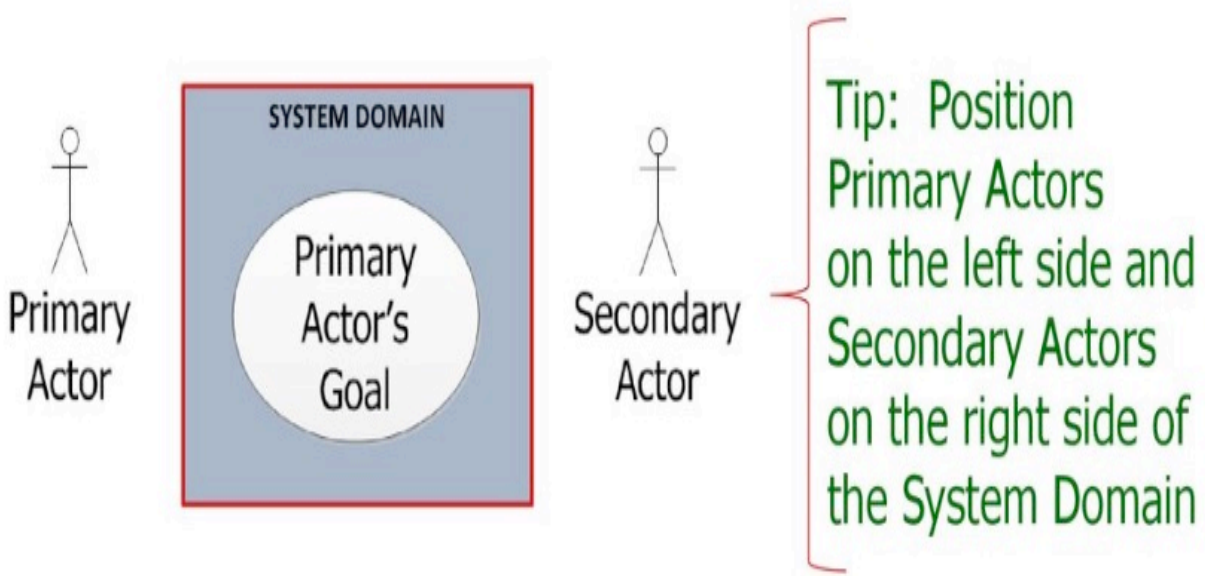


..Use Cases …

Types of Actors

❑Primary Actors

- initiates interaction with the system domain to achieve a goal
- ‘’Who knocks?’’



❑Secondary Actor

- called by the system domain for assistance to achieve the primary actor’s goal

Use Cases

- So as we read our scenario, what or who is the actor????
- *A patient calls the clinic to make an appointment for a yearly checkup. The receptionist finds the nearest empty time slot in the appointment book and schedules the appointment for that time slot. "*

- The actor is a **Patient**.
- The Use Case is **Make Appointment.**
- **Actors**:-
  - Labelled using a descriptive noun or phrase
  - Represented by a stick character

…Use Cases

- The picture below is a **Make Appointment** use case for the medical clinic.

- The actor is a **Patient**. The connection between actor and use case is a **communication association** (or **communication** for short).

- **Actors** are **stick figures**.
- **Use cases** are **ovals**.
- **Communications** are lines that **link actors to use cases.**

Use Case Components

- The use case has three components.
  - The **use case** task referred to as the use case that represents a feature needed in a software system.
  - The **actor(s)** who trigger the use case to activate.
  - The **communication** line to show how the actors communicate with the use case.

…Use Case

Use Case Diagram

- A major process performed by the system that benefits an actor(s) in some way
- Models a dialogue between an actor and the system
- Represents the functionality provided by the system

…Use Case

- Each use case in a use case diagram describes one and only one *function* in which users interact with the system
  - May contain several "paths" that a user can take while interacting with the system
  - Each path is referred to as a scenario
- Labelled using a descriptive **verb-noun** phrase
- Represented by an oval

…Use Case - Relationships

Relationships

- Represent communication between actor and use case 
- Depicted by line or double-headed arrow line
- Also called association relationship

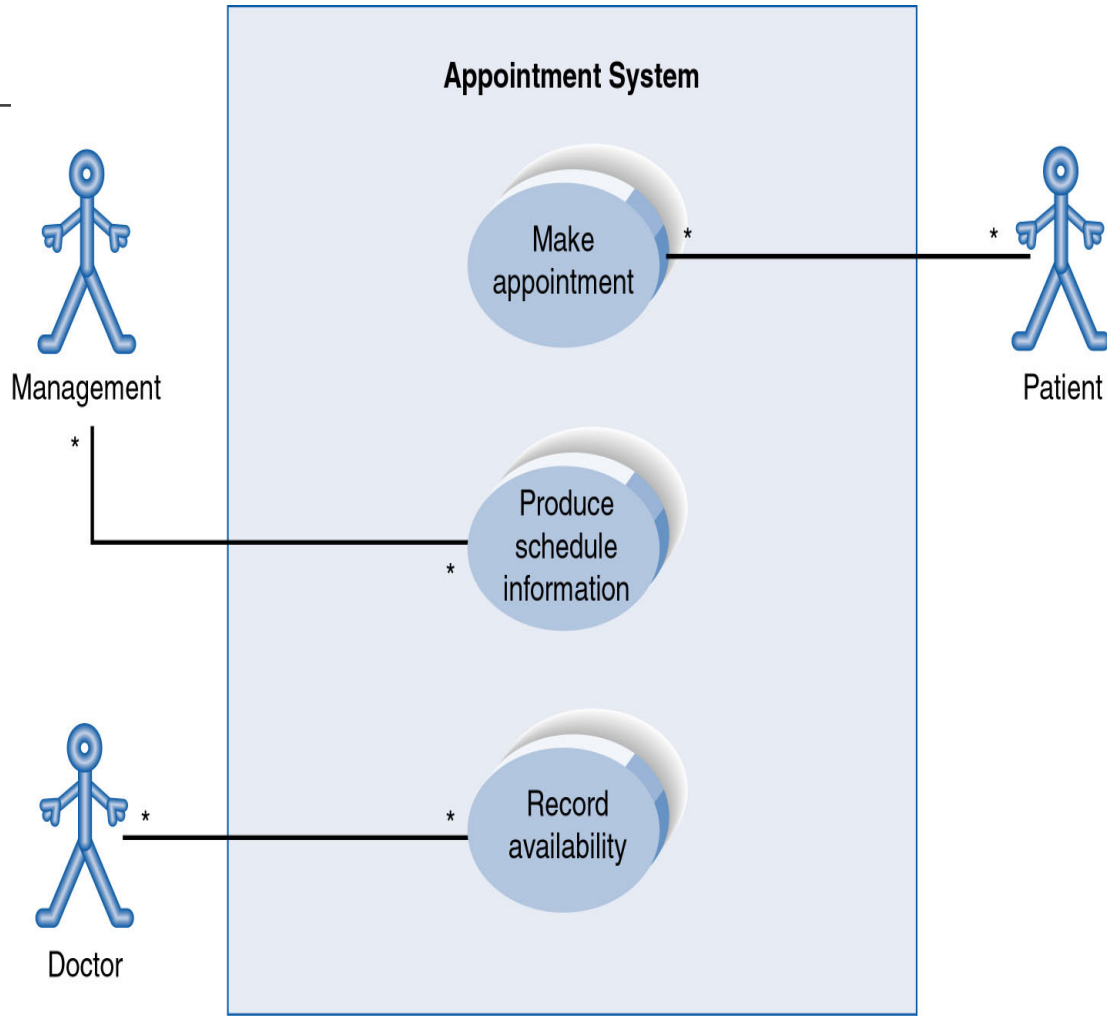
Use Case - Boundary

Boundary

- defines the scope of what a system will be
- A boundary rectangle is placed around the perimeter of the system to show how the actors communicate with the system.
- Boundary b/n the software and the actors of the system.

Use-Case Diagram-Example

- A use case diagram is a collection of **actors**, **use cases**, and their **communications**.



Components of Use Case Diagram

- Other Types of Relationships for Use Cases

Generalization Relationship

- Represented by a line and a hollow arrow
- From child to parent

Include Relationship

- Represents the inclusion of the functionality of one use case within another. It happens every
- Arrow is drawn from the base use case to the used use case
- It is represented by a dashed lined with an arrow. The arrow points to the "included" use case,

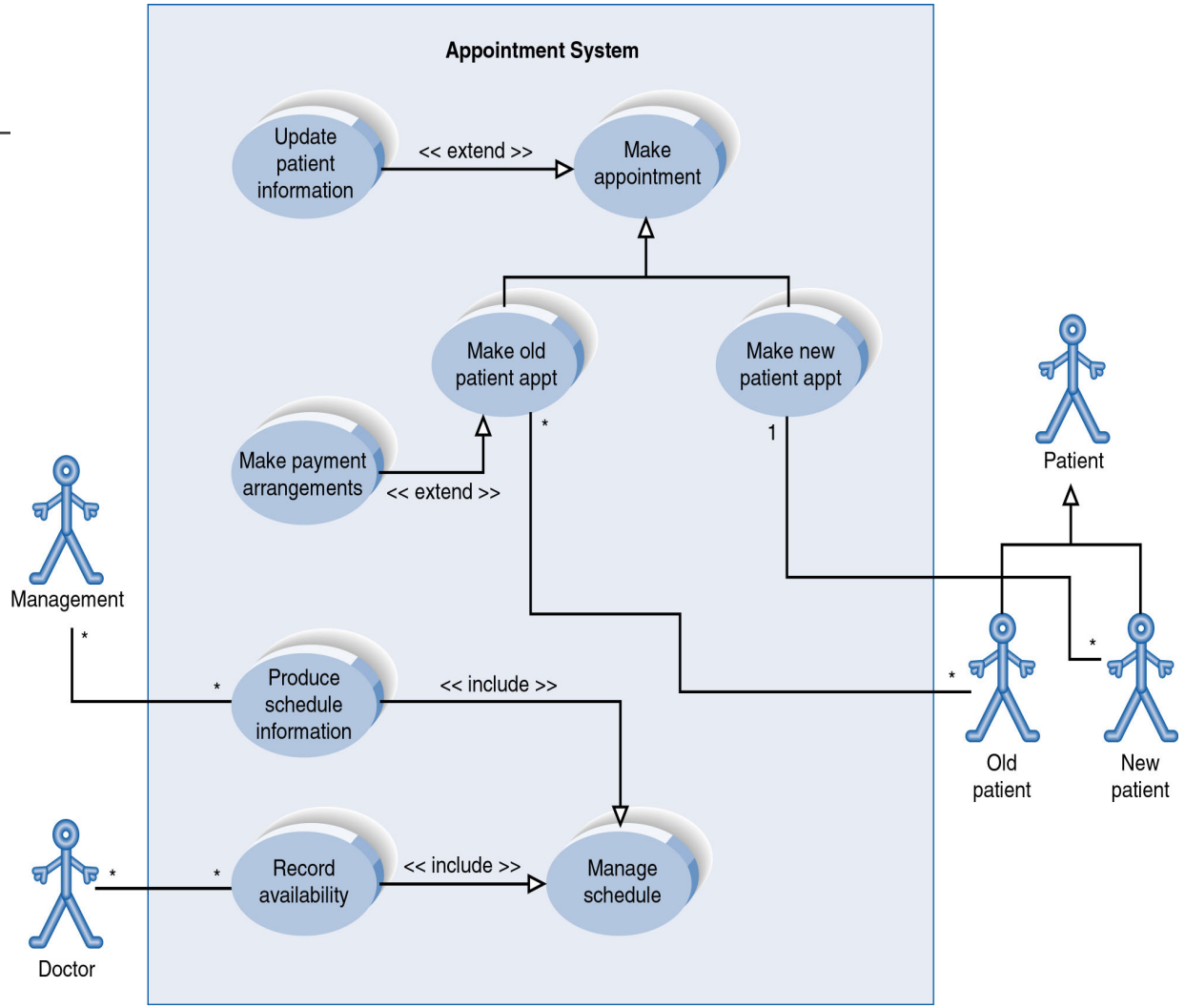
Components of Use Case Diagram

❑ Extend relationship

- Represents the extension of the use case to include optional functionality that could happed just sometime
- Arrow is drawn from the extension use case to the base use case
- Write << **extend** >> above arrowhead line

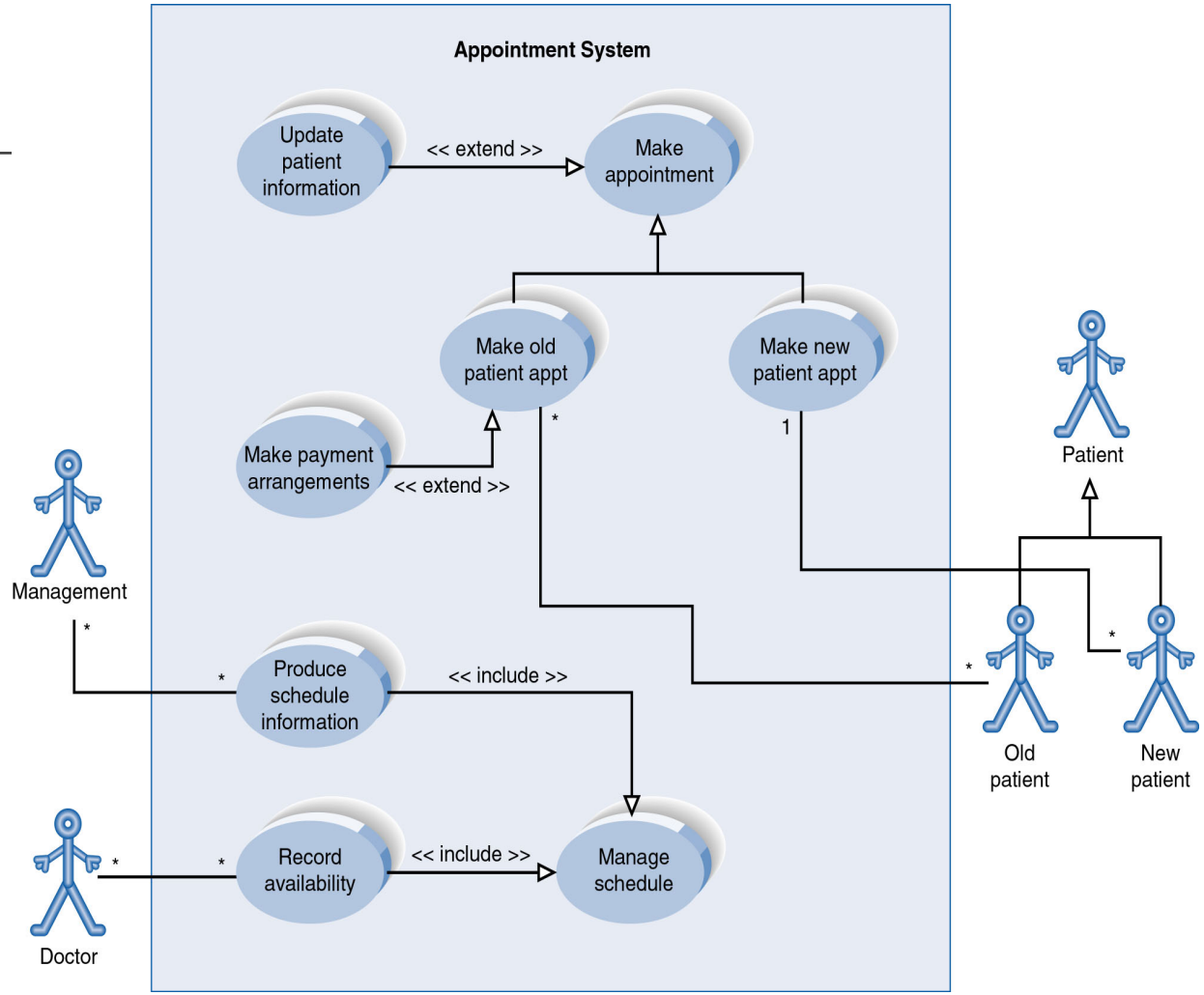
Use Case Diagram:- Examples

Example :- types of relationships in use case diagram



…Use Case Diagram:- Examples

Example :-R types of relationships in use case diagram



Use Cases Identification Techniques

- Three techniques for Identifying use cases

A) User goal technique
B) Event decomposition technique
C) CRUD

- Remember, effective use cases must have understandable actors and goals

….Use Cases Identification Techniques

A) User Goal Technique

- Ask users what they need the system to do .
- The analyst would typically list all the stakeholders who are likely to interact with the system, think through their roles and identify what they would need to get their work done.
- Interview and ask them to describe the tasks the system can help them with
- Identify users by their functionality and organizational level
  - E.g., "I need to *Ship items*, *Track a shipment*, *Create a return*"

…Use Cases Identification Techniques

Example:-

User	User goal and resulting use case
Potential customer	Search for item Fill shopping cart View product rating and comments
Marketing manager	Add/update product information Add/update promotion Produce sales history report
Shipping personnel	Ship items Track shipment Create item return

Use Cases Identification Techniques

❑ User Goal Technique: Specific Steps

1. Identify all the potential users for the new system
2. Classify the potential users in terms of their functional role (e.g., shipping, marketing, sales)
3. Further classify potential users by organizational level (e.g., operational, management, executive)
4. For each type of user, interview them to find a list of specific goals they will have when using the new system (current goals and innovative functions to add value)

…Use Cases Identification Techniques

…User Goal Technique Specific Steps …

5. Create a list of preliminary use cases organized by type of user
6. Look for duplicates with similar use case names and resolve inconsistencies
7. Identify where different types of users need the same use cases
8. Review the completed list with each type of user and then with interested stakeholders

…Use Cases Identification Techniques

B) Event Decomposition Technique

- **More Comprehensive and Complete Technique**
  - Identify the events that occur to which the system must respond.
  - For each event, name a use case (verb-noun) that describes what the system does when the event occurs
- **Event**– something that occurs at a specific time and place, can be described, and should be remembered by the system
- **Use case** – What the **system does** when the even occurs

…Use Cases Identification Techniques

Types of Events

External Event

- an event that occurs outside the system, usually initiated by an external agent or actor

Temporal Event

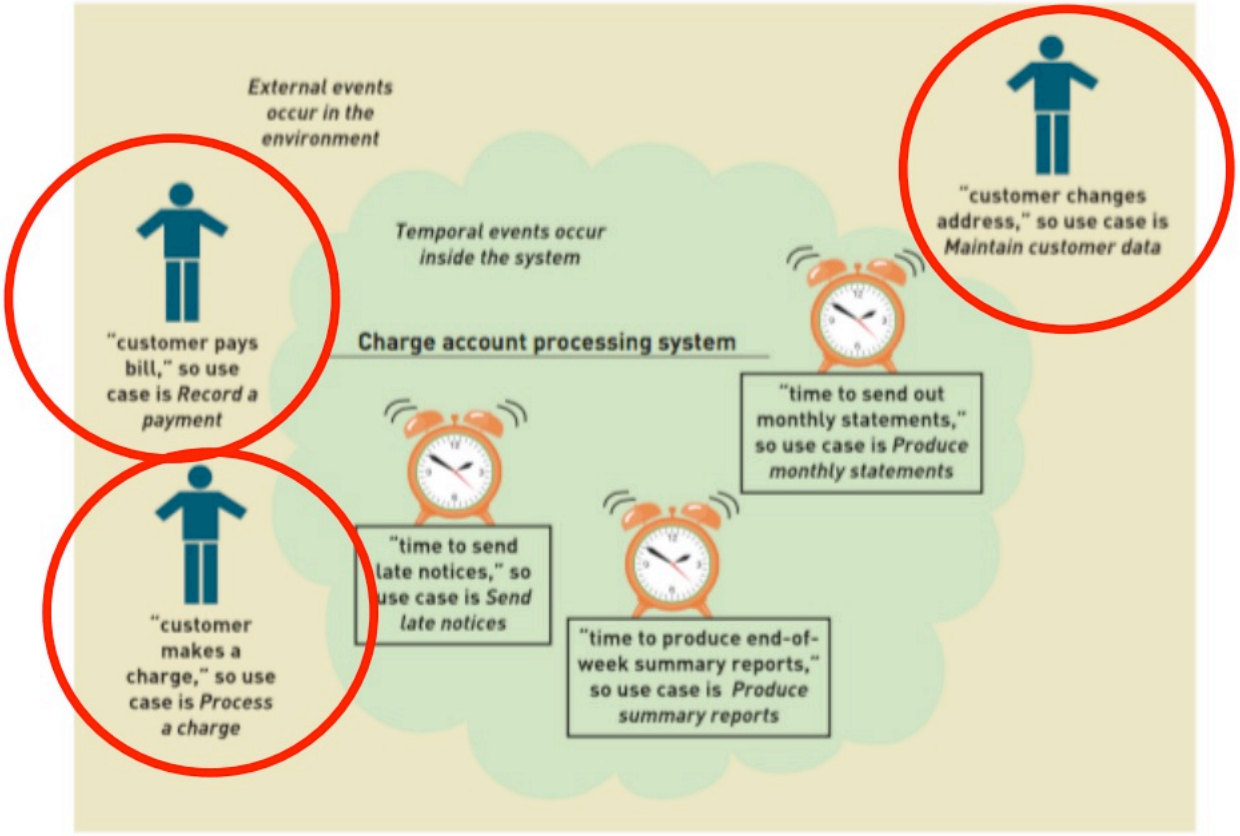
- an event that occurs as a result of reaching a point in time

State Event

- an event that occurs when something happens inside the system that triggers some process
  - For example a state event can occur when an inventory level reaches a pre-order point. So we can have a use case ‘’Reorder Inventory’

…Use Cases Identification Techniques

……B) Event Decomposition Technique :
Example for External event:-



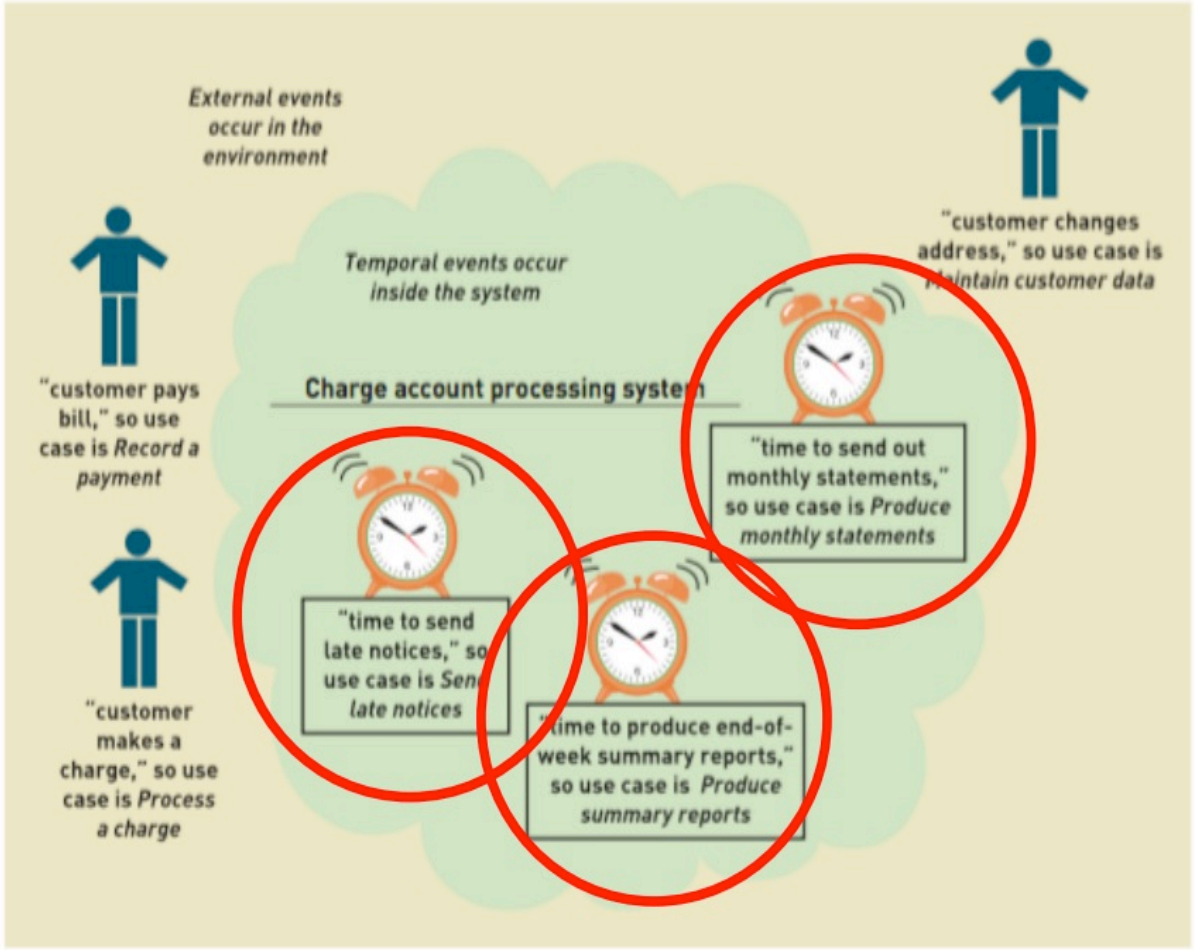
Three Events:
- Customer pays a bill
- Customer makes a charge
- Customer changes their address

Three Use Cases:
- Record a payment
- Process a change
- Maintain customer

…Use Cases Identification Techniques

……B) Event Decomposition Technique :

Example for External event:-



Three Temporal Events:

- Time to send late notices
- Time to send out monthly statements
- Time to produce end of week summary reports

Three Use Cases:

- Send late notices
- Produce monthly statements
- Produce summary reports

…Use Cases Identification Techniques

External Event Checklist

- External agent or actor wants something resulting in a transaction
  - Customer buys a product
- External agent or actor wants some information
  - Customer wants to know product details
- External data changed and needs to be updated
  - Customer has new address and phone
- Management wants some information
  - Sales manager wants update on production plans

…Use Cases Identification Techniques

Temporal Event Checklist

- Internal outputs needed at points in time
  - Management reports (summary or exception)
  - Operational reports (detailed transactions)
  - Internal statements and documents (including payroll)
- External outputs needed at points of time
  - Statements, status reports, bills, reminders
  - Both Temporal and State are internal events, but a state event is triggered by a change in the "state" of the system (or data in the system, and a temporal event is trigger purely by the passage of time.

…Use Cases Identification Techniques

Event Decomposition Technique: Specific Steps

1. Consider the external events in the system environment that require a response from the system by using the checklist shown.
2. For each external event, identify and name the use case that the system requires
3. Consider the temporal events that require a response from the system by using the checklist
4. For each temporal event, identify and name the use case that the system requires and then establish the point of time that will trigger the use case

…Use Cases Identification Techniques

…Event Decomposition Technique: Specific Steps …

5. Consider the state events that the system might respond to, particularly if it is a real-time system in which devices or internal state changes trigger use cases.
6. For each state event, identify and name the use case that the system requires and then define the state change.
7. When events and use cases are defined, check to see if they are required by using the perfect technology assumption.
8. When events and use cases are defined, check to see if they are required by using the perfect technology assumption. Do not include events that involve such system controls as login, logout, change password, and backup or restore the database, as these are put in as system controls.

…Use Cases Identification Techniques

Event Decomposition Technique: Benefits

- Events are broader than user goal: Capture temporal and state events
- Help decompose at the right level of analysis: an elementary business process (EBP)
- EBP is a fundamental business process performed by one person, in one place, in response to a business event
- Uses perfect technology assumption to make sure **functions that support the users work are identified** and not additional functions for security and system controls

Use Cases Identification Techniques

C) CRUD Technique

- CRUD is Create, Read/Report, Update, and Delete (archive)
- Often introduced in database context
- For Customer domain class, verify that there are use cases that create, read/report, update, and delete (archive) the domain class

Data entity/domain class	CRUD	Verified use case
Customer	Create	Create customer account
	Read/report	Look up customer Produce customer usage report
	Update	Process account adjustment Update customer account
	Delete	Update customer account (to archive)

…Use Cases Identification Techniques

CRUD Technique Steps

1. Identify all the data entities or domain classes involved in the new system.
2. For each type of data (data entity or domain class), verify that a use case has been identified that creates a new instance, updates existing instances, reads or reports values of instances, and deletes (archives) an instance.
3. If a needed use case has been overlooked, add a new use case and then identify the stakeholders.
4. With integrated applications, make sure it is clear which application is responsible for adding and maintaining the data and which system merely uses the data.

The END!