Object Oriented Analysis: Basics
Creating Use Case Diagram

A benefit of the Object-Oriented approach is that the same concepts appear in all stages of development. OO techniques view software systems as *systems of communicating objects*. Each object is an instance of a class. All objects of a class share similar features (Attributes and Operations). Classes can be specialized by subclasses. Objects communicate by *sending messages*.

Essence of Object-Oriented Analysis
Consider a problem domain from the perspective of objects (real world things, concepts). Finding and describing the objects (or concepts) in the problem domain. During analysis we model "real world" objects. For example, for a library information system, concepts like book, library, members are considered. We consider what business activities are performed in a library:

- lending books
- keeping track of due dates
- buying new books

In Object Oriented Requirements Analysis represent the business processes in textual narration called *Use Cases*.

Object-Oriented Analysis Activities

- Use Case Analysis (Use Case Model)
- Structural Analysis (Domain Model)
- Behavioral Analysis

Use Case Analysis: Mostly focused on writing text - one overview use case diagram is often enough. Use cases are just a part of functional requirements (only the interactive ones).

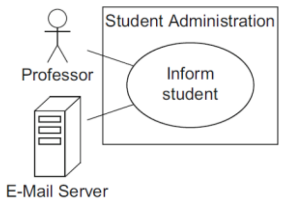
Structural Analysis: Finding the "real-world" objects involved in the use cases and creating basic class diagram(s) to represent them.

Behavioral Analysis: Creating activity diagram and system sequence diagrams to capture use case details. Activity diagrams to depict business workflow. Sequence diagrams for reactive behavior (also with timing).

Use Case Model
A view of a system that emphasizes the behavior as it appears to outside users. A use case model partitions system functionality into transactions '*use cases'* that are meaningful to users '*actors*'. In use case model we create Use Case Diagram(s) and Use Case Descriptions. "Use cases and use case diagram(s) support the gathering and analysis of user-centric requirements by starting with your user's goals".

Use Case: describes functionality expected from the system to be developed. It encompasses a number of functions that are executed when using this system. A use case provides a tangible benefit for one or more actors that communicate with this use case. Use cases describe business processes and requirements in a textual descriptive form. A use case usually represents a user-recognizable "end-to-end" process rather than an individual step or activity within a process (e.g., “rent a video” instead of “pay for video rental”). It is a normal practice to start the name of a use case with a transitive verb followed by an object (e.g., Pay Cash, Update Database, and Prepare Summary Report). Use cases describe what the software system is expected to do (what responsibilities the system is expected to discharge) rather than how it does it. Use cases define functionality from a particular point-of-view. Use cases describe a system from an external usage viewpoint. The set of use case descriptions specifies the complete functional requirements of a system.

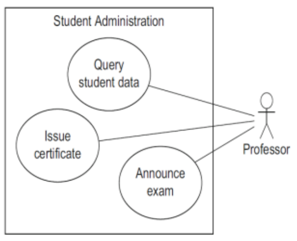
Actor: An actor actually works and interacts with the system. An actor is always clearly outside the system. An actor is connected with the use cases via associations which express that the actor uses a certain functionality of the system. In the use case diagram, actors always interact with the system in the context of their use cases, that is, the use cases with which they are associated. An association is always binary (specified between one use case and one actor). Multiplicities may be specified for the association ends.
Active Actor: Initiates the execution of use cases.
Passive Actor: Provide functionality for the execution of use cases.
In the example use case diagram shown below, both active and passive actors are required for the execution of the use case Inform student.



A primary actor takes an actual benefit from the execution of the use case (in our example this is the Professor), secondary actor E-Mail Server receives no direct benefit from the execution of the use case. If E-mail Server is an internal activity within the system, then we can remove this actor from the use case diagram.

An actor is a user of a system in a particular role e.g. *Book Borrower.* An actor can be human or an external system (e.g. Pressure Sensor).
A *use case* is a task that an actor needs to perform with the help of the system, like Borrow book, or a sequence of transactions performed by the system, or Record Pressure.

Use Case Diagram
A use case diagram for a system shows, in a pictorial form using UML notations, use cases, actors, and their relations (Association). The boundary that separates the system from its environment is shown by a rectangle. Use cases inside its boundary and actors outside. Solid lines are drawn between a use case and the actors that take part in the use case. An actor can initiate more than one use case in the system. Example use case diagram shown below is of a Student Administration system, it has three use cases: (1) Query student data, (2) Issue certificate, and (3) Announce exam.
And has one actor ‘Professor’



UML: Use Case Notation

Actor	A role played by a person, other system or other objects	
Use case	A start-to-finish feature of the system	◯
Association	The communication path between an actor and a use case that it participates in	———
Extend	The insertion of additional behavior into a base use case that does not know about it	<<extend>> - - - - - ->
Use case Generalization	A relationship between a general use case and a more specific use case that inherits and adds features to it	———▷
Include	The insertion of additional behavior into a base use case that explicitly describes the insertion	<<include>> - - - - - ->
Boundary	The boundary of the information system	▭

Use Case Diagram: Creation

Created of use case diagram is top-down approach. First, we define the system boundary, next we identify Actors, and Use cases. Then for each actor, we determine what fundamentally different processes does it participate in? What initial event does an actor perform to start a process? First a highly aggregative view of the system is created, where a use-case diagram showing only actors and the major system functions are spelt out.

Use Case Identification Approaches

- **Actor Based***:* Identify actors related to the system. Identify scenarios these actors initiate or participate in.
- ***Event Based:*** Identify external events that a system must respond to. Relate the events to actors and use cases.
- ***Goal Based:*** [Actors have goals] Find user goals [Prepare actor-goal list]. Define a use case for each goal.

Defining Use Case

Finding Actors

To define use cases, we can determine actors by asking following questions:

- Who or what will use the main functionality of the system?
- Who or what will provide input to this system?
- Who or what will use output from this system?
- Who will need support from the system to do their work?
- Are there any other software systems with which this one needs to interact
- Are there any hardware devices used or controlled by this system?

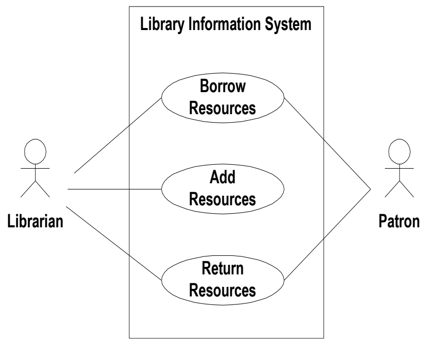
Defining Use Cases

High level use cases briefly describe major processes in the system.
For example, a library contains resources (books and videos). Patrons may borrow these resources, for which information is maintain by the librarian.
From the above information, we can easily identity actors and use cases.

Actors:	Patron, Librarian
Use Cases:	Borrow Resources, Return Resources, Add/Maintain Resources

We can also easily determine which actor interacts with which use case. Having the required information about actors, use cases, and their relationships we can create use case diagram as show:

.



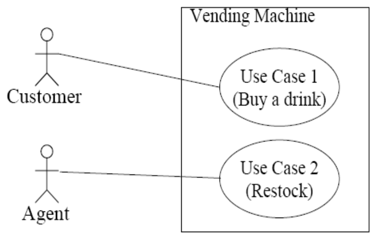
Now we consider a simple example of a vending machine. Sometime when we simply look into the system description as listed below, we may miss important activity being performed for the overall functionality of the system.

> Consider a simplified vending machine, which is used to dispense soft drinks. The vending machine consists of a coin slot for inserting coins, a return tray for returning the customer's money and three buttons used to select Pepsi (tm), Coke (tm) or Sprite (tm). If the customer inserts coins and does not press a button within one minute, the coins will be returned automatically (no coin-return lever). If the customer selects a beverage which is out-of-stock (none left), the coins will also be returned. This vending machine does not provide change - it is up to the customer to insert only enough money to purchase a drink. If sufficient coins are inserted and a button with available beverages is pressed, the appropriate drink is dispensed, the corresponding button is illuminated for five seconds, and the coin slot moves the coins to the general storage area (i.e. the purchase has finished)

First we Identify Actors, then Use Cases. To identify Actors, and Use Cases, simply look for Who, Does what and Why ? for that we must have a Boundary. Next we draw a Boundary, and we place actors outside boundary, and Use-Cases inside boundary. Next we draw associations between Actors and Use-Cases.
For our example vending machine we identify

Boundary: Vending Machine
Actors: Customer, Agent/Operator
Use Cases: Buy Drink, Restock

Very clear relationships between actors and use cases. We create use case diagram:



Using another following example, we create use case diagram.

> A university wishes to increase security in its car park. It has been decided to issue an identity card to all employees. The cards record the employee's name, department and identity number
>
> A barrier, a card reader and a sensor are placed at the entrance of the car park. The driver inserts the numbered card into the card reader. The card reader checks the card number. If the number is valid, the reader sends a signal to raise the barrier and the vehicle can enter

the car park. The sensor sends a signal to the barrier to lower when the vehicle has entered. There is a system at the exit with a barrier which is raised when a car wishes to leave the car park.

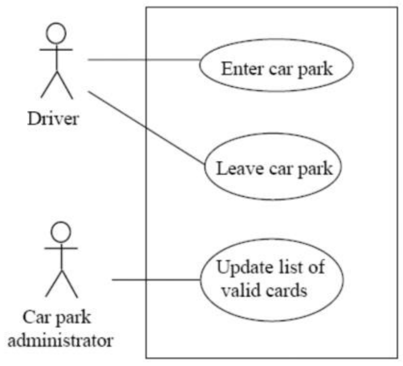
When there are no spaces in the car park a sign at the entrance displays “Full” and is only switched off when a vehicle leaves the park.

Special visitors’ cards, which record a number and the current date, also permit access to the car park. Visitors’ cards may be sent out in advance or collected from reception. All visitors’ cards must be returned to the reception when the visitor leaves the site so that they can be deleted from the list of valid cards.

From the above system description, we can analysis the situation, and can identify the system boundary. The system is about car park activities. With this context in focus, we can identify main activities are entry to and exit from the car park, which is controlled by a barrier. There are certain conditions for entry, i.e. valid card.
By considering overall activities and the conditions we identify following actors and use cases:

Actors: Driver, Car-park-administrator
Use Cases: Enter car park, Leave car park, Update list of valid cards

Very clear from actors and use cases identification, which actor is interacting with which use case. Using this information, we create following use case diagram:


Driver
Enter car park
Leave car park
Car park administrator
Update list of valid cards