**Use Case Diagrams**

Use case diagrams is a graphical representation that describes the interaction of users with the system and also the functionality the system provides to its users. Actors and Use cases are two important elements.

An actor is defined as One or set of objects who directly interacts with the system. Every actor is associated with defined purpose while interacting with the system. An actor can be a person, device or another system.

A Use case is piece of functionality that a system offers to its users. The set of all use cases defines the entire functionality of the system. Use cases also define the error conditions that may occur while interacting with the system.

Use case diagrams are used to:

* Visualize, specify, construct, and document the (intended) behavior of the system, during requirements capture and analysis.
* Provide a way for developers, domain experts and end-users to Communicate.
* Serve as basis for testing

Use cases specify the desired behavior of the system. It is defined as a description of a set of sequences of actions a system performs to yield an observable result of value to an actor. The name of the use case usually starts with a verb.

An actor represents a set of roles that users of use case play when interacting with these use cases. The name of the actor is usually a noun. An actor can be a human or an automated system. Actors are not a part of the system. They require help from the system to perform their task or are needed to execute the system’s functions by triggering a use case. They give inputs to the system and expect an output from the system.

Each use case may include all or part of the following

* Title or Reference Name - meaningful name of the UC
* Author/Date - the author and creation date
* Modification/Date - last modification and its date
* Purpose - specifies the goal to be achieved
* Overview - short description of the processes
* Cross References - requirements references
* Actors - agents participating
* Pre Conditions - must be true to allow execution
* Post Conditions - will be set when completes normally
* Normal flow of events - regular flow of activities
* Alternative flow of events - other flow of activities
* Exceptional flow of events - unusual situations
* Implementation issues - foreseen implementation problems

**Sample Usecase Diagrams**

student

register

update

grades

generate output

faculty

place

phone call

cellular

network

user

receive

phone call

place

conference call

receive

additional call

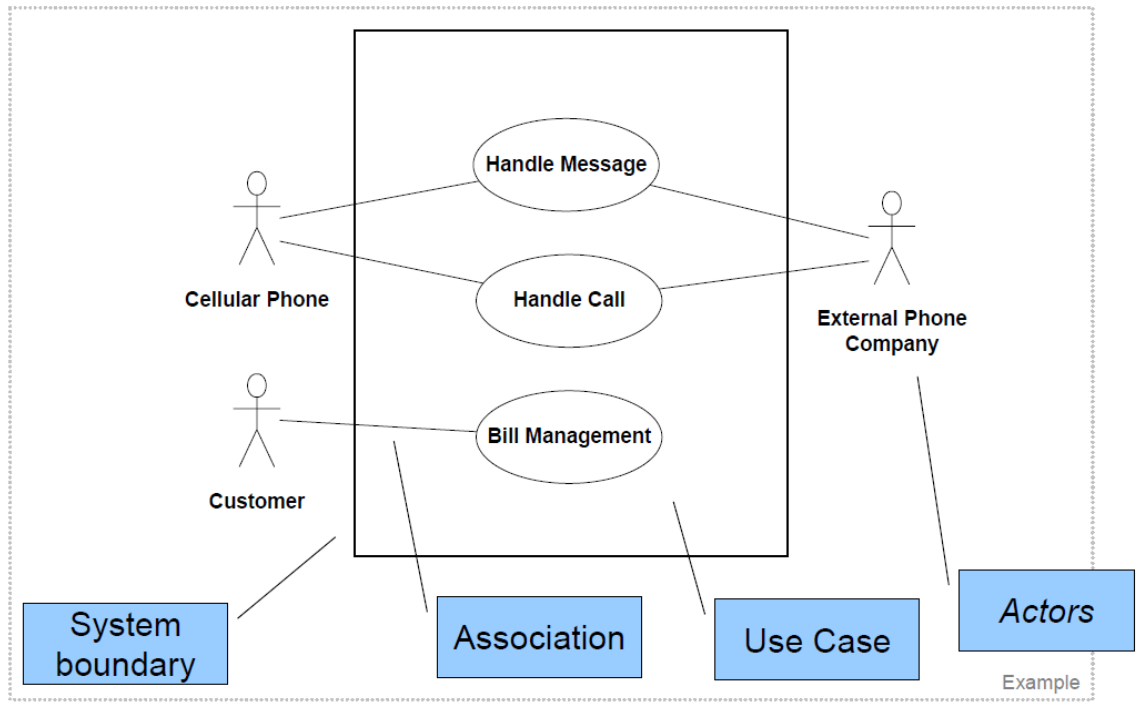
use

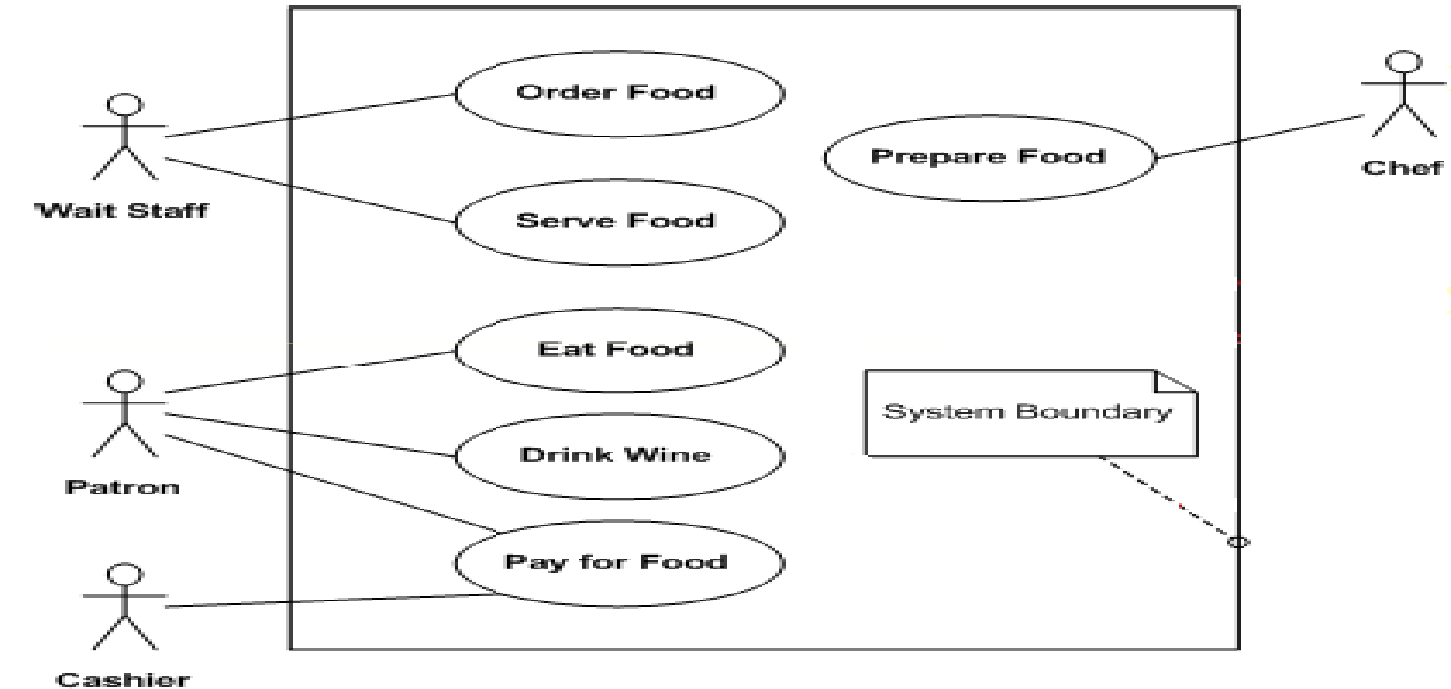
scheduler

<<extend>>

<<extend>>

Cellular Telephone





Actors may be connected to use cases by associations, indicating that the actor and the use case communicate with one another using messages.

update grades

faculty

Use cases are associated by below relationships

* **Generalization** - Use cases that are specialized versions of other use cases.
* **Include** - Use cases that are included as parts of other use cases. Enable to factor common behavior.
* **Extend** - Use cases that extend the behavior of other core use cases. Enable to factor variants.

In Generalizationthe child use case inherits the behavior and meaning of the parent use case. The child may add to or override the behavior of its parent. The usecase generalization relationship is shared with the actor.

parent

child

register student

register graduate student

register non-graduate student

student

non-graduate

student

graduate

student

**Generalization as a Relationships between Actors**

In Include relationship the base use case explicitly incorporates the behavior of another use case. The included use case has no individual existence. It enables avoiding the same flow of events several times by putting the common behavior in a use case of its own.

base

included

<<include>>

Process rental request

Process Sale   
request

Handle Credit Card Payment

<<include>>

<<include>>

In extends relationship the base use case implicitly incorporates the behavior of the another use case at certain points called extension points. The base use case is standalone however its behavior may be extended by the behavior of another use case. It enables modelling of optional behavior or branching under conditions

base

extending

<<extend>>

Request Exam copy

Appeal Exam-grade

<<extend>>

**Sample Use case Specification**

* Name (Must start with a verb)
* Summary
* Actors
* Pre-conditions
  + Conditions that must exist *before* the use case is executed
* Description
  + Textual description (may include steps to execute) and typically is the primary functionality
* Exceptions
  + These are paths which will need to handle exceptions which could be all to provide handling of things which are not provide you with a primary functionality including things like power failure
  + Alternate Flows
  + Handles the other functionality paths for the summary these could be some in the exceptions too
* Post-conditions
  + Conditions that must exist *after* the use case is executed

**Example: ATM**

* Use Case: Withdraw money
* Actors: Customer
* Pre Condition:
  + The ATM must be in a state ready to accept transactions
  + The ATM must have at least some cash on hand that it can dispense
  + The ATM must have enough paper to print a receipt for at least one transaction
* Post Condition:
  + The current amount of cash in the user account is the amount before the withdraw minus the withdraw amount
  + A receipt was printed on the withdraw amount
  + The withdraw transaction was audit in the System log file

|  |  |
| --- | --- |
| **Actor Actions** | **System Actions** |
| 1. Begins when a Customer arrives at ATM |  |
| 2. Customer inserts a Credit card into ATM | 3. System verifies the customer ID and status |
| 5. Customer chooses “Withdraw” operation | 4. System asks for an operation type |
| 7. Customer enters the cash amount | 6. System asks for the withdrawal amount |
|  | 8. System checks if withdrawal amount is legal |
|  | 9. System dispenses the cash |
|  | 10. System deduces the withdraw amount from account |
|  | 11. System prints a receipt |
| 13. Customer takes the cash and the receipt | 12. System ejects the cash card |

**Alternative flow of events:**

**Step 3**: Customer authorization failed. Display an error message, cancel the transaction and eject the card.

**Step 8**: Customer has insufficient funds in its account. Display an error message, and go to step 6.

**Step 8**: Customer exceeds its legal amount. Display an error message, and go to step 6.

**Exceptional flow of events:**

Power failure in the process of the transaction before **step 9**, cancel the transaction and eject the card