Modeling:

Modeling is the fundamental concept in Software Engineering. The term "Model" is derived from the Latin word ***Modulus*** which means measure, rule, pattern, or example to be followed. Modeling describe the system at an abstract level to understand its complexity. It is used for the requirements and specification purpose. Modeling of the system is necessary in order to manage the complexity of the system and quick understanding as well. Modeling process reduces the conflicting views between end-user and system designers. For the modeling of software Object Management Group (OMG) created a modeling language i.e. UML in January 1997. This language simplifies the software design process

UML: (What is UML?)

Unified Modeling Language (UML) is the pictorial/visual language to visualize, specify, construct and document the artifacts of the software system. The term "language" create confusion for some peoples, but it is not like human or computer language. Though, it is a formal specification language, and it has a set of rules to determine how it can be used. It is an industry standard graphical language, which is independent of implementation language and technology, also supportive for diverse application areas. This language can be used for general initial design (Fragmentation) to very specific design (Standardizaiton) across the entire software development life cycle. Moreover, it increases the communication and understanding of the product to the customers and developers.

Purpose of UML (Why use UML?)

UML uses the graphical notations for modeling of software/ product. As the famous proverb says "A pictures is worth a thousand words". A person who designs artifacts use pictures or diagram to assist in the design process. For example, fashion designers, engineers, architects and system analysts all uses diagrams to visualize their products. Graphical notations are more useful to make communication clearer than natural language which is imprecise and code as code in too detailed. *Moreover, it helps us to acquire an overall view of the system.* There are two main advantages of using UML in the software development life cycle. One is to abstract features of the design and other is to show the relationship between elements of the design.

Types of UML Diagrams:

UML has different types of diagram to be used in the software development life cycle. Here is a set of UML modeling diagrams which are most widely used.

- **Use Case Diagram**
- Class Diagram
- Sequence Diagram
- Collaboration Diagram
- State Diagram

Use Case:

Use case diagrams works like a contract between end user and software developers. It is mainly used for capturing user requirements that indicate the functional behavior . The primary purpose of the use case diagrams is to enable the software designers to visualize the different types of roles in a system and how these roles interact with the system. They give us a high level view of the system which becomes helpful while presenting the product to the stakeholders. By use of these diagrams we can extract the roles interacting with the system and functionality provided by the system without going into the inner working of the system. Use cases provide the basis for defining development iteration without revealing the structure of the system. Similarly, they collectively outline the boundaries of the system to be implemented. As described earlier a use case represent functionality provided by the system as an event flow, so a use case consists of following elements:

- Unique name
- Participating actors
- Entry condition
- Flow of events
- Exit conditions
- Special requirements

Components of Use Cases and Notations:

- Actors

An entity that can play a role in the given system. Actor can be a person, organization, or an external system. They have goals and system should respond to actor's goals. In a use case diagram an actor is drawn as stick person with the role name written below as given in the Figure 1.

Figure 1

- Use case

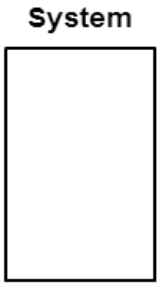
Use case is an action or function within the system. They are basically a set of scenarios which describes the interaction between the user and the system. It may contain additional responsibilities. It is drawn as an oval and named with the function as shown in the Figure 2.

Figure 2

In addition, use case model is the set of all use cases which is the complete description of system functionality and its environment.

- System boundary

It is a rectangular diagram (Figure: 3) represent the boundary between actor and the system. It is useful to visualize the large systems or it can be used to show the different areas covered in different modules of the system to be implemented.



- Figure 3

- Package

Packages are used to group together use cases. It is an option element in use case diagrams which is used for drawing the use case scenarios of complex systems. Its representation is similar to class diagram as shown in Figure 4.

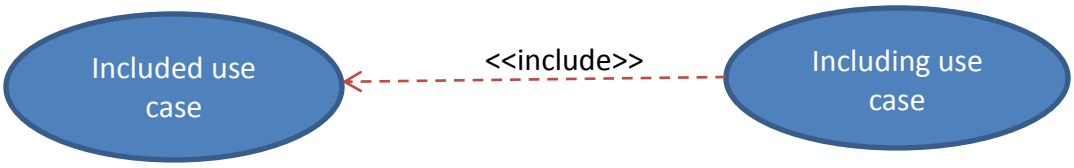
Figure 4

Relationships:

- Include

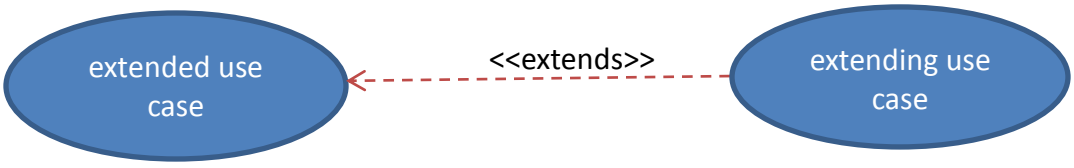
Include is the mandatory behavior of use case diagram i.e. one use case always include other. To represent include relationship, a dashed line is used with arrow head pointing towards the included use

case. Along the line keyword *include* is also added such as <<include>>. Inclusion propery enables he reusability of one use case into another usecase.



- Extend

Extend is the optional behavior of use case i.e. if a use case has an additional property than a new use case can be added as a extended use case. This relationship enables to add new use into an existing use case. Similar to include relationship , a dashed line is used with arrow head pointing towards extended use case. The keyword added in extension is *extend* written as <<extend>>.



- Generalization

In generalization relationship of use case inheratince between use cases occurs. In inheritance process, child use case which is known as specialized usecase inherits the behavior and meaning from parent use case which is known as general use case. This relationship is represented by a line with a triangular arrow head pointing towards the parent use case.

- Association

Association is the communication between actor and usecase and it is represented by a solid line.

Usefulness / Importance of Use Case:

Use case diagrams are basically used for communication with clients and to determine the requirements of the product being developed. The common use of UC diagrams is to model the context of the system, actors, and their interaction with the system. ***They model the requirements of the system that what should the system do independent of how it is done.*** Their simplicity in notation makes use case diagarms a good way for product developers to speak with clients. In stage of design and system analysis , new use cases often generate new requirements. By using UC diagrams test cases can be generated , and collection of scenarios for a use case may suggest a group of test cases for those scenarios.

UML Modeling Tools:

- Rational Rose (www.rational.com) by IBM
- TogetherSoft Control Center, Borland (http://www.borland.com/together/index.html)
- **ArgoUML** (free software) (http://argouml.tigris.org/) OpenSource; written in java
- **Microsoft visio** https://products.office.com/en/visio/flowchart-software
- Others (http://www.objectsbydesign.com/tools/umltools_byCompany.html)

References:

- Ludewig, Jochen , Models in software engineering – an introduction, Springer-Verlag, 2003
- http://www.tutorialspoint.com/uml/uml_overview.htm
- http://creately.com/blog/diagrams/use-case-diagram-tutorial/