Assignment No. 7

Name - Kuldharan Sumit Dattatraya
Roll No - 200401
Class - TE4
Subject -

Problem statement -

Consider a library, where a member can perform two operations : issue book & return it. A book is issued to a member only after verifying his credentials. Develop a use case diagram for the given library system by identifying the actors & use cases & associate the use cases with the actors by drawing a use case diagram.

Theory -

* What is UML?

It is the general-purpose modeling language used to visualize the system. It is a graphical language that is standard to the software industry for specifying, visualizing, constructing, & documenting the artifacts of the software systems, as well as for business modeling.

* Benifits of UML.

- Simplifies complex software design can also implement OOPs like a concept that is coidely used.
- It reduces thousands of words of explanation in a few graphical diagrams that may reduce time consumption to understand.
- It helps to acquire the entire system in a view.
- It becomes very much easy for the software programmer to implement the actual demand once they have a clear picture of the problem.

* use case diagrams are helpful in the following situations:

① Before starting a project you can create use-case diagrams to model a business so that all participats in the project share an understanding of the coworkers, customers, & activities of the business.

② While gathering requirements, you can create use-case diagrams to capture the system requirements and to present to others what the system should do.

③ During the analysis and design phases, you can use the use cases & actors from ~~user~~ your use-case diagrams to identify the classes that the system requires.

④ During the testing phase, you can use use-case diagrams to identify tests for the system.

* Types of UML-

① Behavioural UML diagrams-

- Activity Diagram
- Sequence Diagram
- use case Diagram
- State Diagram
- Communication Diogram
- Interaction overview Diagram
- Timing Diagram

② Structural UML diagrams are-

- Class Diogram
- Package Diogram
- Object Diagram
- Component Diagram
- Composite Structure Diagram
- Deployment Diogram

A use case diagram is graphical depiction of a user's possible interactions with a system. A use case diagram shows various use cases and different types of users the system has & will often be accompanied by other types of diagrams as well. The use cases are represented by either circles or eclipses. The actors are often shown as stick figures. A use case is not a single scenario but rather a 'class' that specifies a set of releated usuage scenarios, each of which captures a specific course of interactions that take place between one or more actors & the system. Therefore, the description of an individual use case typically can be divided into a basic course & zero or more alternative courses.

The basic course of a use case is the most common or important sequence of transactions that satisfy the use case. The basic course is therefore always developed first. The alternative courses are variants of the basic course & are often used to identify error handling. Within reason, the more alternative courses identified & described, the more complete the description of the use case & the more robust the resulting system. As a user centred analysis technique, the purpose of a use case is to yield a result of measurable value to an actor in response to the initial request of that actor. A use case may involve multiple actors, but only a single actor initiates the use case. Because actors are beyond the scope of the system, use-case modelling ignores direct interactions between actors. A use case may either be an abstract use case or a concrete use case. An abstract use case will not be instantiated on their own, but is only meaningful when used to describe functionality that is common between other use cases. On the other hand, a concrete use case can be instantiated to create a specific scenario.

*What is a use case diagram?

In the unified Modeling Language (UML), a use case diagram can summarize the details of your system's users (also known as actors) & their interactions with the system. To build one, you'll use a set of specialized symbols & connectors. An effective use case diagram can help your team discuss & represent:

- Scenarios in which your system or application interacts with people, organizations or external systems.
- Goals that your system or application helps those entities (known as actors) achieve.
- The scope of your system.

*When to apply use case diagrams?

A use case diagram doesn't go into a lot of detail— for example, don't expect it to model the order in which steps are performed. Instead, a proper use case diagram depicts a high level overview of the relationship between use cases, actors & systems. Experts recommended that use case diagram be used to supplement a more descriptive textual use case. UML is the modelling toolkit that you can use to build your diagrams. Use cases are represented with a labelled oval shape. Stick figures represent actors in the process, and the actor's participation in the system is modelled with a line between the actor & use case. To depict the system boundary, draw a box around the use case itself.

*Purpose of Use case Diagram-

use case diagrams are typically developed in the early stage of development and people often apply use case modelling for the following purposes:

- specify the context of a system.
- capture the requirements of a system.
- Validate a system architecture.
- Drive implementation & generate test cases.
- Developed by together with domain experts.

* How to identify Actor

often, people find it easies to start the requirements elicitation process by identifying the actors. The following questions can help you identify the actors of your system:

- Who uses the system?
- Who installs the system?
- Who starts up the system?
- Who maintaince the system?
- Who shuts dow the system?
- What other systems use this system?
- Who gets information from this system?
- Who provides information to the system?
- Does anything happen automatically at a present time?

* How to identify Use Cases.

Identifying the Use Cases, and then the the scenario-based elicitation process carries on by asking what externally visible, observable value that each actor desires. The following questions can be asked to identify use cases, once your actors have been identified:

- What functions will the actor want from the system?
- Does the system store information? What actors will create, read, update or delete this information?
- Does the system need to notify an actor about changes in the internal state?
- Are there any external events the system must know about? What actor inform the system of those events?

* Use case diagram components.

System - A specific sequence of actions & interactions between actors and the system. A system may also be referred to as a scenario.

Actors - The users that interact with a system. An actor can be a person, an organization, or an outside system that interacts with your application or system. They must be external objects that produce or consume data. Stick figures that represent the people actually employing the use cases.

Goals - The end result of most use cases. A successful diagram should describe the activities & variants used to reach the goal.

Use cases - Horizontally shaped ovals that represent the different uses that a user might have.

Associations - A line between actors & use cases. In complex diagrams, it is important to know which actors are associated with which use cases

System boundar boxes - A box that sets a system scope to use cases. All use cases outside the box would be considered outside the scope of that system. For example, Psycho killer is outside the scope of occupations in the Chainsaw example found below.

Packages - A UML shape that allows you to put different elements into groups. Just as with component diagram, these groupings are represented as file folders.

* Where to use a use case diagram &

As we have already discussed there are five diagrams in UML to model the dynamic view of ~~a view~~ system. Now each & every model has some specific purpose to use. Actually these specific purposes are different

angles of a running system. To understand the dynamics of a system, we need to use different types of diagrams. Use case diagram is one of them & its specific purposes is to gather system requirements & actors. Use case diagrams specify the events of a system & their flows.

But use case diagram never describes how they are implemented. Use case diagram can be imagined as a black box where only the input, output & the function of the black box is known. These diagrams are used at a very high level of design. This high level design is refined again & again to get a complete & pratical picture of the system. As well structured use case also describes the pre-condition post condition, & exceptions. These extra elements are used to make test cases when performing the testing.

But use case diagram never describes how they are implemented. Use case diagram can be imagined as a black box where only the input, output & the function of the black box is known. These diagrams are used at very high level of design. This high level design is refined again & again

- A use case diagram should be as simple as possible.
- A use case diagram should be complete.
- A use case diagram should represent all interactions with the use case.
- If there are too many use cases or actors then only the essential use cases should be represented.
- A use case diagram should describe at least a single module of a system.
- If the use case diagram is large then it should be generalized.

Although use case is not a good candidate for forward & reverse engineering still they are used in a slightly diffn way to make forward & reverse engineering. The same is true for reverse engineering. Use case diagram is used differently to make it suitable for reverse engineering.

In general use case diagrams are used for:
- Analyzing the requirements of a system.
- High-level visual software designing.
- Capturing the functionalities of a system.
- Modelling the basic idea behind the system.
- Forward & reverse engineering of a system using various test cases.

UML use case diagrams are ideal for
- Representing the goals of system-user interactions.
- Defining & organizing functional requirements in a system.
- Specifying the context & requirements of a system.
- Modelling the basic flow of events in a use case.

* Relationships in Use Case Diagrams -
There are five types of relationships in a use case diagram. They are,
- Association between an actor & a use case.
- Generalization of an actor.
- Extend relationship between two use cases.
- Include relationship between two use cases.
- Generlization of a use case.

Use case Relationship
① Communication Association -
- Between an actor & a use case.
- Unidirectional or bidirectional
- Unidirectional association shown by an arrow head to the destination.

(c) Relationship between use cases -
- Include or extend.
- Stick arrow head is put nearest to the use case that is being used.
- Stick arrow head is put next to the base use case that is being extended.
- Include & extend are stereotype associations between use cases.

* Advantages of use case diagram.
- places a requirements statement "in context".
- Acquiring requirements by exploring "what if" questions.
- Scoping required system by exploring system actions.
- Validating requirements by exploring alternative courses.
- provide a basis for system & acceptance testing.
- Reduce system complexity.
- Excellent for communicating/agreeing requirements.
- Encourage inter-disciplinary learning.
- Little training & experience needed to use them.
- Scenarios can evolve over time.

* Disadvantages of UML diagram:
- poor identification of structure & flow.
- Geometric & temporal information hard to describe.
- Unsystematic craft.
- Time-consuming to generate.
- Limited softwar tool support.
- Handling unforeseen combinations of abnormal events.
- Require the co-existance of prototype.
- Still poor integration coith established methods.
- Difficult to generalise from scenarios - abstract use cases.
- Scenario management is difficult.

* Star UML -

Star UML is a UML tool by MK Lab. The software was lincensed under a modified version of GNU GPL until 2014, when a recoritten version 2.00 was released under a proprietary license. After being abandoned for some time, the project had a reterival to move from Delphi to Java, before stopping again. Meancohile, an unaffiliated community fork was created under the name of White star UML. In 2014, the original developers rewrote Star UML & released it as shareware under a proprietary license. The stated goal of the projes was to replace larger, commercial applicaitons such as Rational Rose & Borland Together. Star UML supports most of the diagrams type specified in UML 2.0. Since version 4.0.0 (October 29 2020) it includes timming & interaction overview diagrams. UML supports most of the diagram types specified in UML 2.0. Star UML 2.0 uses its own file format with the .mdj extension. can import files from star UML 1.0 using format .uml Also can import a fragment from a .mfj file

Conclusion -

Use cases should be only one of several ways of capturing user requirements. The model of objects, classes & their semantic relationships should be consistent with but not totally driven by the use cases. Designers should exercise great care to avoid the creation of partial, redundant variants of classes, especially on large projects involving multiple builds & releases.