Use Cases

Software Requirements

- A **software requirement** is:
  1. A **condition or capability** needed by a **user** to **solve a problem** or **achieve an objective**.
  2. A **condition or capability** that must be **satisfied by a system**.
  3. A **documented representation** of a **condition or capability** as described in (1) and (2).
- A **functional requirement** is a requirement that **describes what a system must do** including **processes, interfaces, and data**.
  - **Function requirements** are described in **use case documents**, and are modelled in **OOA** with **use case diagrams, class diagrams, and interaction diagrams**.
- A **non-functional requirement** is a requirement that **specifies how the system must perform** including **response time, security considerations, and the volume of data**.
  - **Non-functional requirements** are documented in a **requirement list**.
- A **usability requirement** is a requirement that is concerned with **matching the system to the way people work**.
  - **Usability requirements** measure objectives, including **characteristics of users, tasks users undertake, situational factors, and the acceptance of criteria for the working system**.
  - **Usability requirements** are documented in the **list of requirements** and may be tested by **prototypes**.

Techniques for Finding Requirements

- There are **several techniques for finding requirements**.
- **Background reading** is a technique for finding **requirements** that aims at **understanding the organization** and it's **business objectives**.
  - Reading material includes **reports, charts, policies, job descriptions, and existing system documentation**.
  - This technique **works best** in the **initial stage of fact finding** and when the **analyst is not familia with the organization**.
- **Interviewing** is a technique for getting an **in-depth understanding** of the **organization's objectives, and user roles**.
  - Interview subjects include **managers, staff, and customers**.
  - This technique **works best** when **in-depth** information is required. The effectiveness of this technique depends on the skill of the interviewer.
- **Observation** is a technique for find out what **really happens**, **not what people say happens**.
  - Items to observe include **what happens to documents, how people carry out processes, quantitative data, a processes from end-to-end**.
- **Document sampling** is a technique for **providing statistical data** about **transaction volumes** and **activity patterns**.

  - Document sampling information includes **copies of empty and completed documents, screenshots of existing systems, numbers of forms filled in, and the lines on the forms**.
  - This technique **works best** when **error rates are high, large volumes of data are being processes**.

- **Questionnaires** are a technique for obtaining the **views of a large amount of people in a way that can be analyzed statistically**.
  - Questionnaires include **postal, web-based, and email questionnaires with open-ended and closed-ended questions**. They also **gather opinions and facts**.
  - This technique **works best** when **staff organizations are geographically dispersed, the system is going to be used by the general public, and when you need to obtain the views of a large amount of people**.

Use Case Descriptions

Use Cases

- A **use case** is primarily an **action of writing text**.
- An **actor** is a **person or thing** that **interacts** with the software.
- A **use case** describes **what happens in the system** when an **actor uses the software**.
- **Use case modeling** may include a **use case diagram**; showing the **name, actors, and relationships** of **use cases**.
- **Use case development is a key characteristic** of the **Unified Process**. It serves to help discover **functional requirements, design construction, test plans, and maintenance to prepare user manuals**.

Types of Use Case Descriptions

- There are **3 types** of **use case descriptions**:
  1. A **brief use case description** consists of a **single paragraph** describing the **main success scenario**.
  2. A **casual use case description** consists of **multiple informal paragraphs** covering both the **main success scenario**, and **various alternatives**.
  3. A **fully dressed use case description** consists of a **detailed description of all steps involved in the main success, alternative, and exception scenarios**. This is usually accompanied by supporting sections, such as pre-conditions and post-conditions.
- The **fully dressed use case description** contains the following sections:
  1. **The primary actor -** The user who interacts with the system during this use case.
  2. **Stakeholders and their interests -** The use case covers the functionality that satisfies all the required stakeholder's interests.
  3. **Pre-conditions -** Conditions that must be true before the main scenario begins without any checking.
  4. **Post-conditions -** Conditions that must be true on the successful completion of a use case.
  5. **The main success scenario (detailed) -** The typical path to a successful outcome (describes what needs to happen not how).

6. **Alternative flows (detailed)**. - All other paths that may lead to a success or failure.
7. **Exceptions (detailed)**. Exceptions that may occur.
8. **Special requirements -** Non-functional requirements for the use case.
9. **Open issues -** Anything that hs an effect on the functionality of the use case.

Use Case Diagrams

Models

- A **model** is a **representation of an entity**.
- **Models** can be used in simulations, evolve as we learn, and are quicker to build than the real thing.
- A **useful model** has the **right level of detail** and represents only what is **important** for the task.

Diagrams

- A **diagram** is a **simplified drawing** showing the **appearance, structure, or workings** of something.
- There are **rules / standards** for drawing **diagrams**.

Unified Modeling Language Diagrams

- The **Unified Modeling Language (UML)** is a general-purpose **developmental modeling language** that is intended to provide a standard way to **visualize a system**.
- To model systems based on the **UML**, we use **UML diagrams**.
- **UML diagrams** consist of **icons, 2D symbols, paths, and strings**.
- It is more important to **fully and correctly communicate ideas** than it is to completely adhere to UML notation standards.

UML Use Case Diagrams

- A **use case diagram** shows the **names of actors and use cases** along with **their relationships**.
- There are **four elements** in use case diagrams: **actors, use cases, subsystem boundaries, and relationships**.
- A **top-level diagram** includes **top-level use cases** that **interact directly** with one or more **actors**.
- A **sub-level diagram** includes a few of the **top-level** use cases and **other related use cases**.
- **Actors** in a use case diagram are **external entities** who **use the system**.
  - **Primary actors** achieve their goals by using the system.
  - **Supporting actors** provide services to the system.
- To **identify actors** you need to look at **who will be using the system, and what will they be doing with it**.

Name and Size of Use Cases

- A use case describes **what happens** in the **system** when it is **used by an actor**.
- The **name** of a **use case** is typically a **verb and a noun**.
- The **size of use cases** should be **adequate** (not too big or too small).
- **Use cases** focus on **what, not how**.

System Boundary

- A **use case model** usually consists of **multiple diagrams**.
- The **boundary separates top-level use cases from actors**. It does not include use cases for external behaviors.

Entity Relationships

- To connect entities we use relationships.
- The **include relationship** indicates that **an entity always uses one or more instances of another entity**.
- The **extend relationship** indicates that **one use case flows directly from another**.
- To draw the **include and extend relationships in a UML diagram** you use a **dotted line with arrows pointing towards the entity being included or extended**. The arrow should say ”include” or ”extend”.

Creating Use Case Diagrams

- To create a **use case diagram**, you do the following:
  1. **Identify actors and uses cases** by reviewing the **vision document** and the **list of requirements**.
  2. **Add elements to high/low level diagrams** by showing the **system boundaries** as boxes, **placing primary actors outside of the boxes** and **primary use cases inside the boxes**.
  3. **Refine the diagram** by **adding use case and actor relationships**, **adjusting the placement of elements**, and **linking the use cases with important scenarios**.

Activity Diagrams

- An **activity diagram** can be used to **model tasks, describe use-case functionality, describe the logic of an operation, and model the activities that make up the life cycle in the unified process**.
- Before drawing you should ask yourself:
  1. What is the purpose of the diagram?
  2. What is the name of the use case?
  3. What level of detail is required?
- To create an **activity diagram** you do the following:
  1. Identify actions and their order of flow.
  2. Work on the main flow of actions by
     - Creating a start node.

   - Placing main actions in the order of flow.
   - Adding a final node at the flow end.
   - Linking actions with necessary decisions.
   - Identifying and creating alternative flows.
   - Introduction fork/joint nodes.
   - Continuing with other use cases (optional).

3. Refine the diagram by adding objects and object flows, as well as control flows with IO pins.
4. To view the **diagram notation** visit https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/.