University of Tripoli
Faculty of Information Technology

Department of Software Engineering

Software Requirements Analysis ITSE311 -- F2023

Use Cases

INTRODUCTION

- Among the different techniques for capturing system behavior, use cases are of major importance because their approach ensures expressing requirements in <u>a way understandable to both users and developers</u>.

Understanding Use Cases

- Use cases is a valuable approach to discovering and presenting functional requirements.
- use case is expressed with a flow of events presenting scenarios of system use understandable to both users and developers.
- use case is considered as a practical technique for documenting requirements where use cases emphasize system users' goals and demonstrate how the system works with users to achieve these goals .
- Example: A user (actor) logs on to (goal) the automated teller machine (ATM) (system) using an ATM card and entering personal identification number (user–system interactions)

Understanding Use Cases

- A use case model consists of use case diagrams depicted in UML and use case description providing detailed information about the requirements.
- the description is considered as the most important part because it documents the flow of events between the actors and the system in question.
- Use case diagrams provide a sort of road map of relationships between the use cases in a model and help system stakeholders better understand the user–system interaction process.

Use Case Benefits

- Emphasizes user goals and objectives.
- Users and other stakeholders become more involved in the process of requirements engineering
- Provides common notation understandable to both professionals and nonprofessionals.
- Use cases are action-oriented and focused on system functionality—system functionality is decomposed into a set of discrete tasks.
- Helps developers derive test cases and some design insights.
- It may be reused for user documentation (both text and diagrams).
- It may be used as a basis for planning iterative work.

Use Case Actors

- An actor represents a class of system users playing a specific role in use cases.
- Note that several users may play the same role in a system and an actor represents a single role, not a single user. But a single user may take one or more roles.
- The roles presented by actors may be taken not only by humans but also by organizations, machines, or machine components, and other software systems or software components.

Kinds of Use Case Actors

- **Primary actors** have goals that can be met by using the services (or functions) of the system. Note that identifying all the primary actors is of major importance because by knowing these actors we may identify their goals driving the use cases.

Example: A client (primary actor) deposits money (goal) in the bank (system).

- **Supporting actors** provide supporting services to the system. Such an actor is often a computerized system or machine, but it can also be another system, organization, or person. It is important to identify the supporting actors because they help the design team discover external interfaces and protocols needed by the system in question.

Example: Any money transfer operation requires data exchange with the ATM server (supporting actor).

Kinds of Use Case Actors

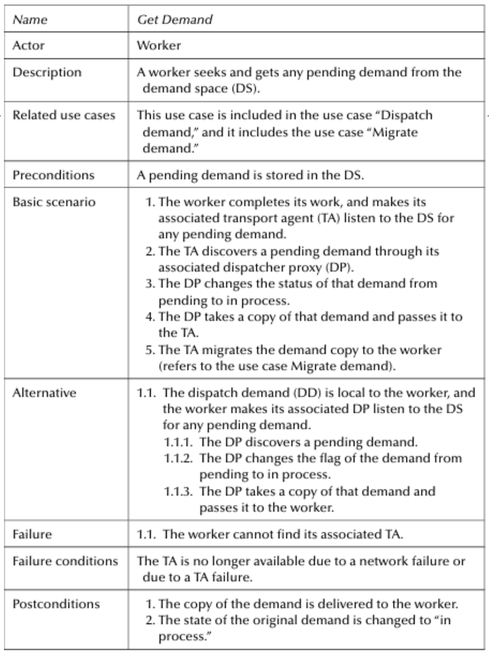
- **Offstage actors** are not directly involved in any use cases, but have an interest in some aspects of the system behavior; that is, in some of the use cases. Often, offstage actors are not presented in the use case model unless they are explicitly named by other system stakeholders.

Example: When pressed, the emergency button turns on the hidden camera and calls the police department (offstage actor).

Use Cases

- Use cases can be written in different formats:
  - Brief—the use case in question is summarized in one paragraph by emphasizing only the basic scenario.
  - Casual—informal text usually expressed in multiple paragraphs that cover various use case scenarios (cf. the Use Case Scenarios section).
  - Fully dressed—all steps of all the possible scenarios are explained in detail; supporting sections are used to add more information, such as actors, related use cases, preconditions, postconditions, and failure conditions.

Fully Dressed Use Case Description Example

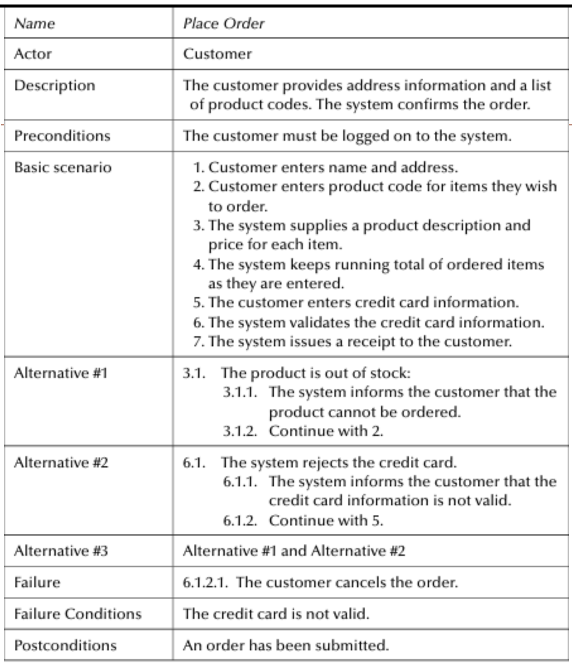


Name	*Get Demand*
Actor	Worker
Description	A worker seeks and gets any pending demand from the demand space (DS).
Related use cases	This use case is included in the use case "Dispatch demand," and it includes the use case "Migrate demand."
Preconditions	A pending demand is stored in the DS.
Basic scenario	1. The worker completes its work, and makes its associated transport agent (TA) listen to the DS for any pending demand. 2. The TA discovers a pending demand through its associated dispatcher proxy (DP). 3. The DP changes the status of that demand from pending to in process. 4. The DP takes a copy of that demand and passes it to the TA. 5. The TA migrates the demand copy to the worker (refers to the use case Migrate demand).
Alternative	1.1. The dispatch demand (DD) is local to the worker, and the worker makes its associated DP listen to the DS for any pending demand. 1.1.1. The DP discovers a pending demand. 1.1.2. The DP changes the flag of the demand from pending to in process. 1.1.3. The DP takes a copy of that demand and passes it to the worker.
Failure	1.1. The worker cannot find its associated TA.
Failure conditions	The TA is no longer available due to a network failure or due to a TA failure.
Postconditions	1. The copy of the demand is delivered to the worker. 2. The state of the original demand is changed to "in process."

Use Case Scenarios

- Often a use case defines a collection of possible scenarios each presenting a distinct flow of events that may occur during the performance of a use case by an actor.
- A use case scenario is a sequence of actions and interactions between actors and the system in question.
- A use case scenario is usually described in text and depicted visually with a sequence diagram
- The description of a use case may encompass three different types of scenarios (paths):
  - basic scenario,
  - alternative scenario
  - failure scenario.

Use Case "Place Order"



Name	*Place Order*
Actor	Customer
Description	The customer provides address information and a list of product codes. The system confirms the order.
Preconditions	The customer must be logged on to the system.
Basic scenario	1. Customer enters name and address. 2. Customer enters product code for items they wish to order. 3. The system supplies a product description and price for each item. 4. The system keeps running total of ordered items as they are entered. 5. The customer enters credit card information. 6. The system validates the credit card information. 7. The system issues a receipt to the customer.
Alternative #1	3.1. The product is out of stock: 3.1.1. The system informs the customer that the product cannot be ordered. 3.1.2. Continue with 2.
Alternative #2	6.1. The system rejects the credit card. 6.1.1. The system informs the customer that the credit card information is not valid. 6.1.2. Continue with 5.
Alternative #3	Alternative #1 and Alternative #2
Failure	6.1.2.1. The customer cancels the order.
Failure Conditions	The credit card is not valid.
Postconditions	An order has been submitted.

Use Case Preconditions and Postconditions

- Use case preconditions is when a use case may be defined with the required states of the actors and the system itself at the time a use case is to be started.
- Note that the preconditions are not a description of the events that start the use case in question, but a description of the specific conditions under which the use case is applicable.
- A use case may also be defined with the so called postconditions used to define the required states of the system at the end of a use case.

Use Case Diagram

- A use case diagram visualizes and conveys the structure of the use case model of a system.
- It shows interactions between a system and the use case actors.
- A use case diagram is a graph where the nodes represent both actors and use cases, and the lines represent relationships among use cases and actors.
- the use case diagrams have low information content and they are complementary documents meant to assist the use case textual descriptions
- A use case is represented by an ellipse that can be labeled.
- An actor is usually drawn as a named stick figure or, alternatively, as a class rectangle with the «actor» keyword.

Use Case Diagram

- A use case diagram may also include an optional system boundary box, which is denoted as a rectangle around all the use cases. A system boundary box indicates the scope of the system in question. Anything within the box represents functionality that is in the scope of the system and anything outside the box is not.
- UML introduces notations for four different types of relationship found in use case models:
  - Association (UML notation: solid line). An association determines the communication path between an actor and the associated use case. In general, an association exists whenever an actor is involved in an interaction described by a use case; that is, an actor is involved in a use case.

Use Case Diagram

  - Generalization (UML notation: solid line with closed arrowhead). The generalization relationship denotes a relationship between a more generic use case (or actor) and a more specific use case (or actor), where the latter inherits features from the former.
  - Extend (UML notation: dashed line with open arrowhead and the «extend» keyword). The extend relationship denotes the insertion of additional behavior (use case scenarios) into a base use case where the latter does not know about it.
  - Include (UML notation: dashed line with open arrowhead and the «include» keyword). The include relation- ship denotes the additional behavior included in a base use case that explicitly defines the inclusion.

Algorithm for Determining Use Cases

1. Determine the system boundary.
2. Identify the primary actors.
3. For each primary actor identify the user goals.
4. Define use cases that will help primary actors achieve their user goals.
5. Identify the sub-goal use cases
6. Identify the supporting actors providing supporting services to the system.
7. Identify the offstage actors.

Use Case Traps to Avoid

- Too many use cases.
- Scenario duplication across use cases. To avoid duplication, define supplementary use cases and use the include
- Use case name duplication. Use cases must be named with unique names across a use case model.
- User interface design included in use cases. A use case should not emphasize the system user interface, but the system functionality.
- Data definitions included in use cases. Note that data requirements are nonfunctional and should be tackled by other requirements engineering approaches.

Example

ATM Use Case Model:

A Case Study in the text-book

The End……