Tutorial 6: UML Use Case Diagram

QUICK REVIEW

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

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, and systems. Experts recommend that use case diagrams 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 and use case. To depict the system boundary, draw a box around the use case itself.

UML use case diagrams are ideal for:

- Representing the goals of system-user interactions
- Defining and organizing functional requirements in a system
- Specifying the context and requirements of a system
- Modelling the basic flow of events in a use case

The notation for a use case diagram is pretty straightforward and doesn't involve as many types of symbols as other UML diagrams.

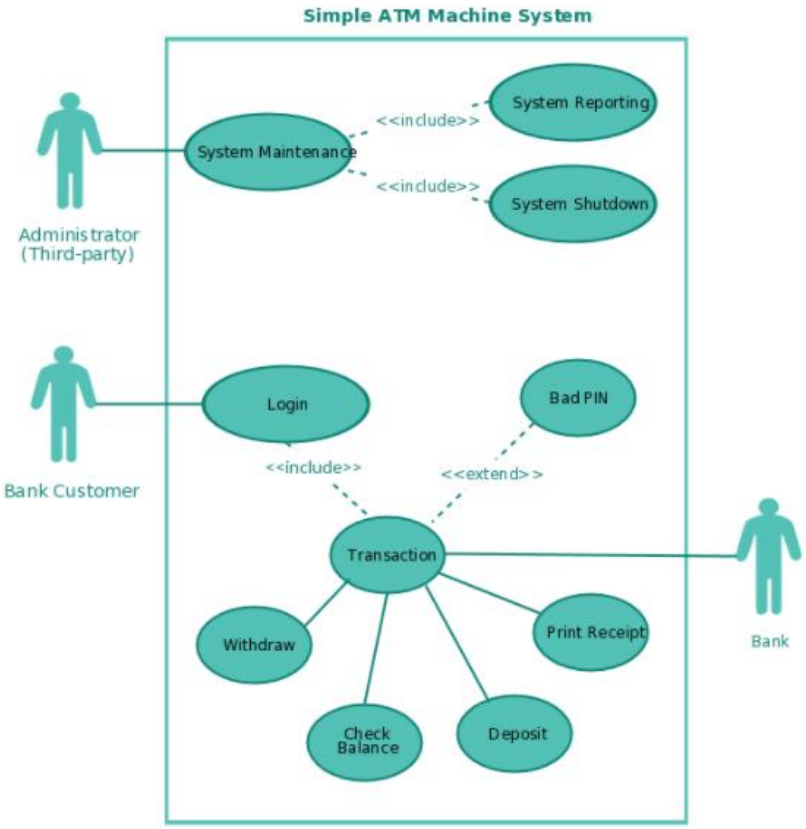
- **Use cases**: Horizontally shaped ovals that represent the different uses that a user might have.
- **Actors**: Stick figures that represent the people actually employing the use cases.
- **Associations**: A line between actors and use cases. In complex diagrams, it is important to know which actors are associated with which use cases.
- **System boundary 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.

LAB PRACTICES

With Astah Professional you can create Use Case diagram. An introduction to creation of Use Case diagram with Astah Professional is available here: Use Case Diagrams in Astah.

T6.1. Simple ATM Machine System - warm-up ~ 10 min.

An automated teller machine (ATM) is an electronic telecommunications device that enables customers of financial institutions to perform financial transactions, such as cash withdrawals, deposits, funds transfers, or account information inquiries, at any time and without the need for direct interaction with bank staff.. The figure below shows a simplified use case diagram of ATM system.
As a warm-up, use Astah Professional to create the use case diagram shown in the figure below.



T6.2. Vending Machine ~ 15 min.

Read the description below:

A vending machine sells small, packaged, ready to eat items (chocolate bars, cookies, candies, etc.). Each item has a price and a name. A customer can buy an item, using a smart card (issued by the vending machine company) to pay for it. No other payment forms (i.e. cash, credit card) are allowed. The smart card records on it the amount of money available.

The functions supported by the system are:

- Sell an item (choose from a list of items, pay item, distribute item)
- Recharge the machine
- Set up the machine (define items sold and price of items)
- Monitor the machine (number of items sold, number of items sold per type, total revenue)

The system can be used by a customer, a maintenance employee (who recharges items in the machines), an administrator (who sets up the machine).

T6.2.1. Class Diagram

Based on the description, draw a class diagram for the system.

T6.2.2. Use Case Diagram

Based on the description, draw a use case diagram for the system.

T6.3. Wireless Restaurant Management ~ 20 min.

Read the description below:

In the past, restaurants were managed with paper, except for payment, where often a register was used. Now, many restaurants are managed using wireless devices. Each waiter has a wireless device (a PDA with wi-fi connection), the kitchen has one or more PCs, the reception desk has a PC, too, all are connected.

Key functions to be considered are:

- **Order from waiter to kitchen**: the waiter takes orders from a table and sends them to the kitchen.
- **Modify order from waiter to kitchen**: the waiter takes changes from a table and sends them to the kitchen.
- **Inquiry**: the waiter asks about the status of an order
- **Dish from kitchen to waiter**: the waiter should be alarmed to collect a dish ready to be dispatched to the right table
- **Checkout**: compute the amount due by a table, manage payment, issue receipt. Payment by the customer could be made at the table (the waiter handles the credit card or cash and the receipt) or at a register.

Consider also that usually restaurants, unless very small, are divided in zones allocated to one or more waiters. Besides, large restaurants could have many exits and many registers for payment.

T6.3.1. Requirements

Based on the description, develop a list of requirements. You may add some arbitrary non-functional requirements, too.

T6.3.2. Class Diagram

Based on the description, draw a class diagram for the system.

T6.3.3. Use Case Diagram

Based on the description, draw a use case diagram for order management system.