**Dalhousie University**

**Faculty of Computer Science**

# CSCI 3132 – Object Orientation and Generic Programming

## Week 1 – Expressing Client Requirements for a Software System

# Creating Software

- Software Development Life Cycle
  - **Requirements Analysis**
  - Design
  - Implementation
  - Testing
  - Evolution

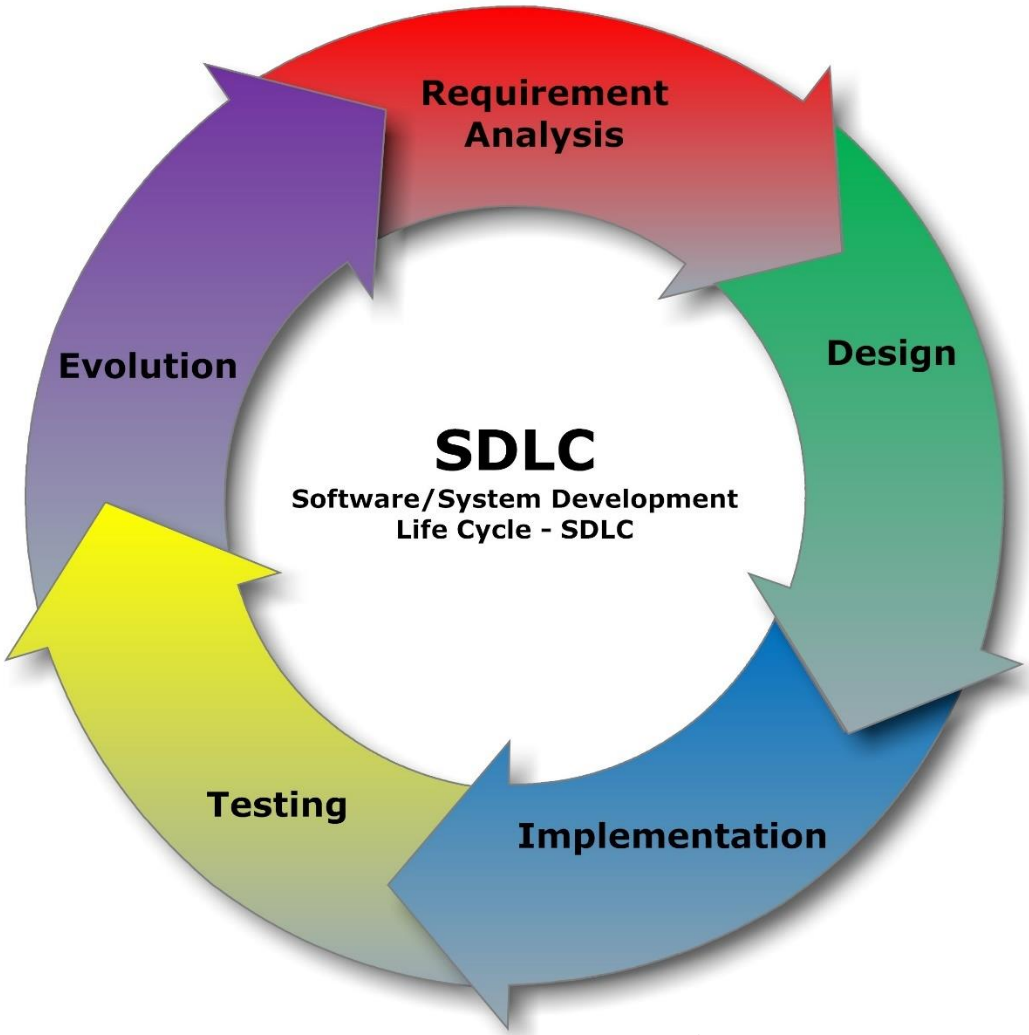


Image ref: Wikimedia Commons

# Requirements Analysis

## What is a Requirement?

"(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 met or possessed by a system or system component to satisfy a contract, standard, specification, or other formally imposed documents.

(3) A documented representation of a condition or capability as in (1) or (2)"

(IEEE, 1990)

## Requirements Analysis

- RA is the process of determining quantifiable, relevant and detailed user requirements
- Often a mismatch between the required and the delivered project/product
  - This mismatch results from poor requirements analysis

## Requirements Activities

- Eliciting Requirements
- Expressing Requirements
- Prioritizing Requirements
- Analyzing Requirements
- Managing Requirements

## Eliciting Requirements

- An investigative and interactive process to help the client differentiate between their wants and needs
  - "Want" is a feature the client would like to have, but is not core to the product/project itself
  - "Need" is a core functionality of the product/project required to fulfill its purpose
- Done through client interviews and meetings

# Expressing Requirements

- Involves framing the requirements after these have been elicited
- Typical representations include:
  - Use Cases
  - User Stories

# Prioritizing Requirements

- Prioritize clients needs
- "MoSCoW[1]" method of prioritization or other similar techniques
  - Acronym for "Must have", "Should have", "Could have", and "Would like but won't get"

[1] developed by Dai Clegg

## Analyzing Requirements

- Examining the listed requirements to ensure that they are:
  - Clear
  - Complete
  - Consistent
- This helps:
  - Identify potential conflicting requirements
  - Ensure identified requirements truly reflect the needs of the product/project

# Managing Requirements

- A continuous process that involves:
  - Re-organizing and reusing subsets of the requirements
  - Keeping track of priorities, analyses, and changes in requirements
- Note that a change in one requirement may affect other requirements and subsequently the product/project

## Example of Requirements

- Bad requirements:
  - “We need several users to be able to log in simultaneously in our system”
  - The system should allow users to search for a book in the library using several parameters
  - The client needs to reduce orders processing errors significantly within a short time to increase their revenue
  - System should require a strong password containing a mix of alpha-numeric characters

## Example of Requirements

- Bad requirements:
  - "We need **several users** to be able to log in simultaneously in our system"
  - The system should allow users to search for a book in the library **using several parameters**
  - The client needs to reduce orders processing errors **significantly** **within a short time** to **increase their revenue**
  - System should require a strong password containing a **mix of alpha-numeric characters**

## Example of Requirements

- Good requirements:
  - "We need to allow **100 concurrent users** in our system"
  - The system should allow users to search for a book in the library **by ISBN, Author, or Title**
  - The client needs to reduce orders processing errors **by 50%** by the **end of this year** to **increase their revenue by 10%**
  - System should require a strong password containing a **minimum of 8 characters** with **at least one uppercase letter and 2 numbers**

## Techniques used for Capturing Requirements

- Use Cases
- User Stories
- Storyboards
- Wireframes
- Acceptance Tests
- Product Backlog
- Story Maps

## Use Cases

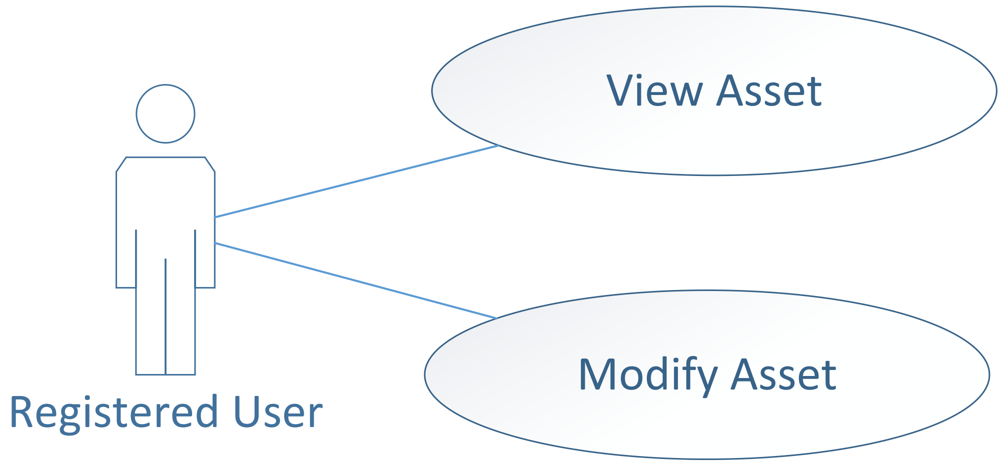
## Use Cases

- Methodology to identify, clarify and organize system requirements
- Consists of a set of possible sequences of interactions between systems and users
- A collection of possible scenarios related to a particular goal
- Should contain all system activities that have significance to the users

## Components of a Use Case

- Actor
  - A human or other external system that interacts with the system
- Use Case
  - Represents a goal or objective that the user of a system wants to achieve
- Relationship
  - Indicates access of system by an Actor. Connects Actor with a use case

# Presenting Use Cases

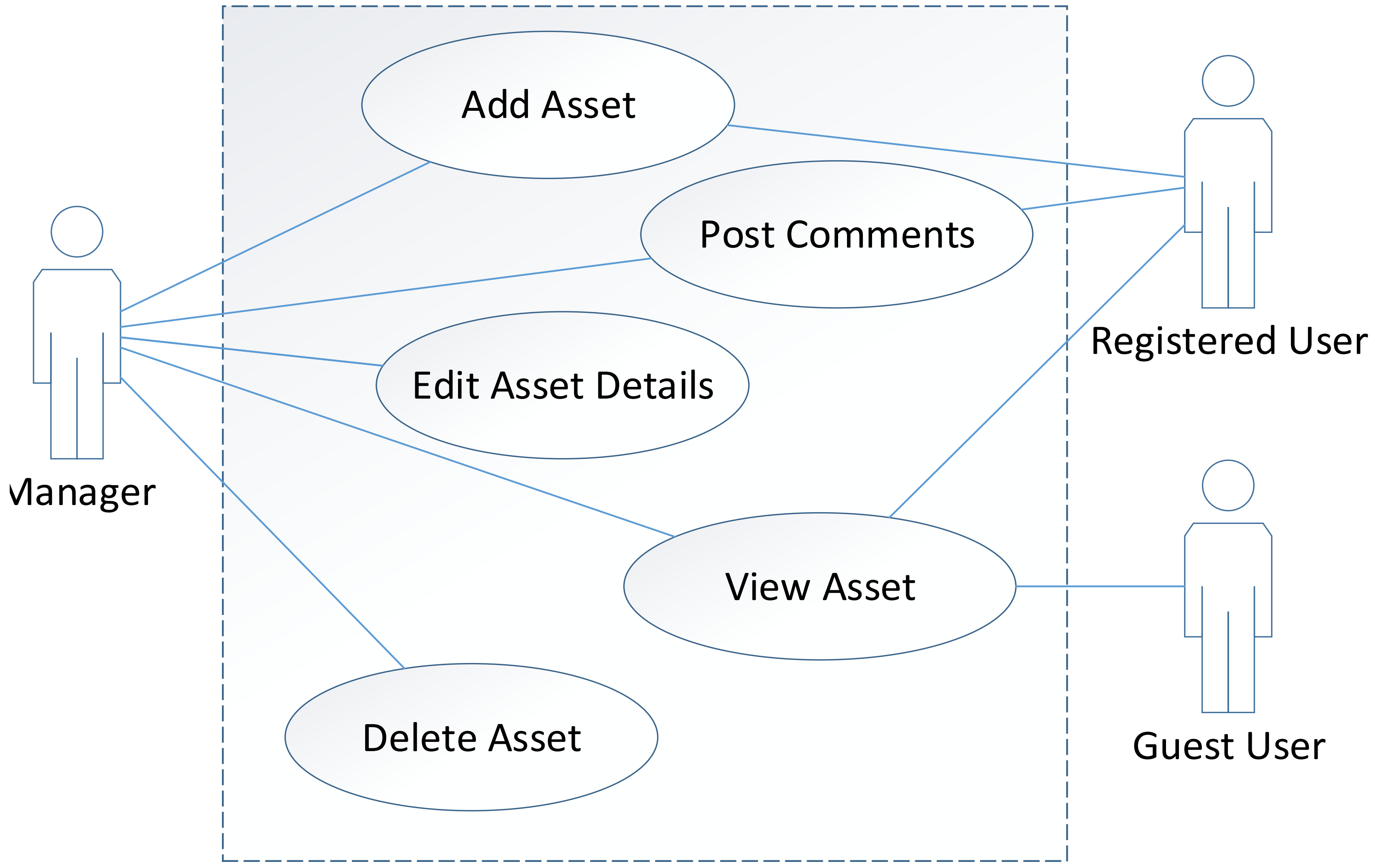


| **Name** | |
|---|---|
| Participating Actors | |
| Goals | |
| Triggers | |
| Pre-condition | |
| Post-condition | |
| Basic Flow | |
| Alternate Flows | |
| Exceptions | |
| Qualities | |

# Presenting Use Cases

| **Name** | **Modify Asset** |
|---|---|
| Participating Actors | Registered member |
| Goals | Member modifies details of an asset, previews changes and saves |
| Triggers | Member invokes a modify request |
| Pre-condition | Asset details with editing enabled is presented to member |
| Post-condition | Asset details are saved and an updated view is shown |
| Basic Flow | - System provides asset details in an editable area<br>- Member modifies asset's details and submits changes<br>- System saves the changes, completes post-processing<br>- System presents updated view to member |
| Alternate Flows | - Member cancels the modify request<br>- System discards any changes, shows the original view |
| Exceptions | No changes made after modify invoked |
| Qualities | - Updated asset details correctly presented to member<br>- Page takes less than 10 seconds to load |

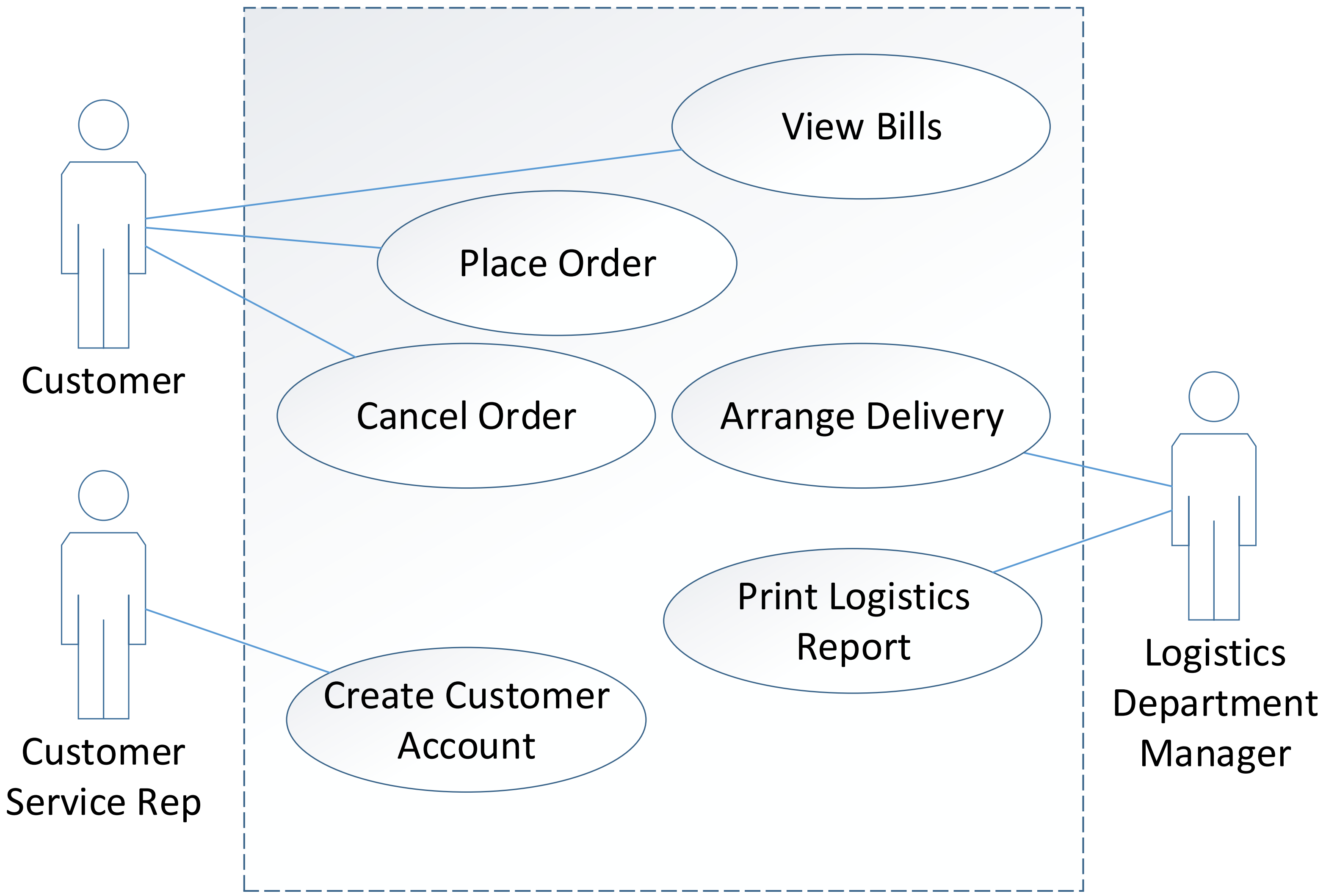
# Use Case Diagrams



## Client Requirements – Pizza Delivery

- Customer should be able to place an order for pizza. A customer service rep should be able to create a customer account if the customer doesn’t have one. The customer should be able to view her bills and cancel her order. The logistics department manager should be able to arrange for pizza delivery and print logistics report at the end of the day.

# Use Case – Pizza Delivery



## Use Case Exercise

- Read the following user requirements carefully, then sketch the use case diagram for the system, identifying the actors and their relationship with each use case
- Present any use case identified in the diagram, in detail, using the use case table shown earlier

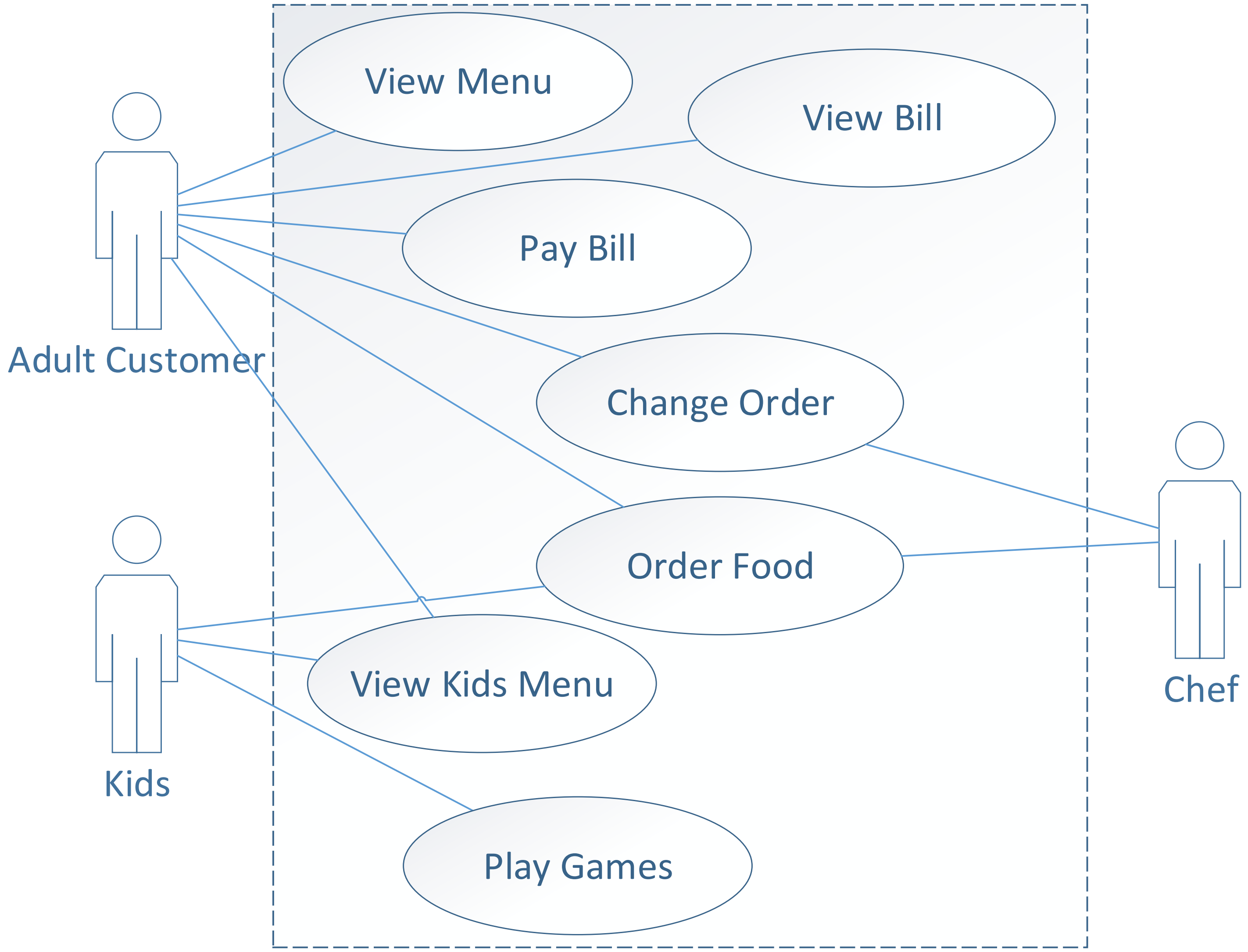
# Client Requirement – Restaurant Ordering System

Customer should be able to view the menu and place an order using a touch screen device. Kids should have a separate page with their own menu, maybe with their own games. But it should be easy enough for them to be able to make an order themselves.

Customers should be able to specify any changes for their meals before submitting their order. The kitchen should be able to view these orders as they come in. Customers should be able to view and pay their bill within the system

| **Name** | |
|---|---|
| Participating Actors | |
| Goals | |
| Triggers | |
| Pre-condition | |
| Post-condition | |
| Basic Flow | |
| Alternate Flows | |
| Exceptions | |
| Qualities | |

# Sample Solution



# Sample Solution

| **Name** | **Pay Bill** |
|---|---|
| Participating Actors | Adult Customer |
| Goals | Customer previews bill and makes payment |
| Triggers | Customer invokes bill payment |
| Pre-condition | Food has been ordered by the customer |
| Post-condition | Payment is successful and receipt is printed |
| Basic Flow | - System shows itemized bill for ordered food<br>- Customer previews bill<br>- Customer pays bill<br>- System confirms payment and prints receipt |
| Alternate Flows | - Customer cancels payment<br>- Customer modifies order after payment |
| Exceptions | Customer's electronic payment is declined |
| Qualities | - Process completed in less than 5 clicks (taps)<br>- Multiple payment options available |

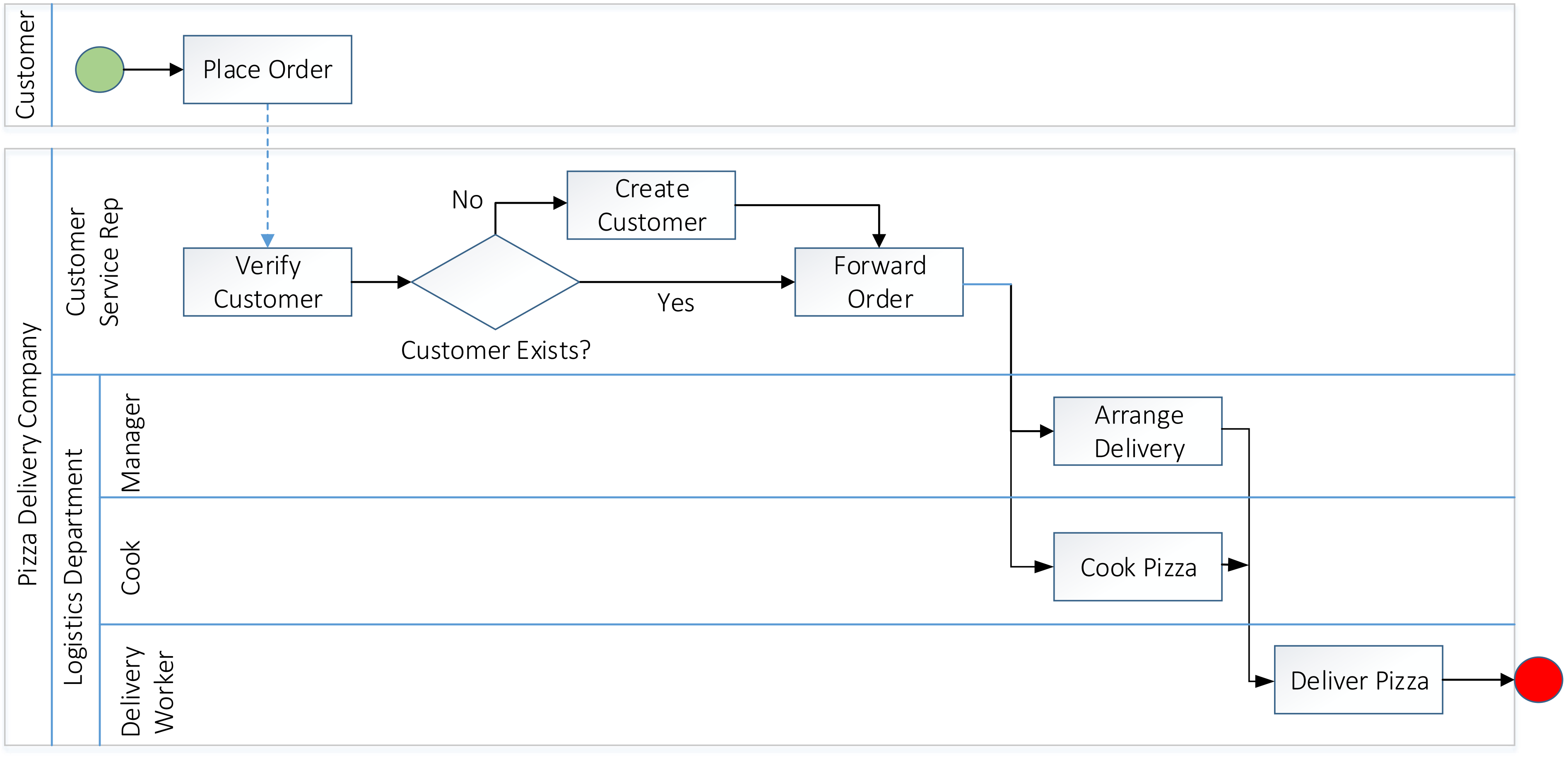
## Business Process Diagram (BPD)

- A business process is modelled using a BPD
- BPD is a flow-chart showing the process flow, participants involved and message exchanges between participants
- BPDs model how different participants collaborate together to accomplish a business objective

## BPD versus Use Case Diagram

- BPD models workflows. It shows how participants work together, while use case diagram shows the system functions that users want
- BPD can help in constructing a use case diagram

# BPD Example – Pizza Delivery


Customer
Place Order
Pizza Delivery Company
Customer Service Rep
Verify Customer
No
Create Customer
Yes
Forward Order
Customer Exists?
Logistics Department
Manager
Arrange Delivery
Cook
Cook Pizza
Delivery Worker
Deliver Pizza