Use Cases Testing

Lesson 7: Use case Testing

Capgemini

Lesson Objectives

To understand the following topics:

- Use case modeling
- Advantage of use cases
- Actor
- Goals and Requirements
- Goals and scenarios
- Naming Conventions
- Alternate Path
- Exceptions
- Errors
- Precondition & Post-condition
- Good practices
- Failure scenarios

Use Case Usage

Use cases help to:

- Describe a business process
- Capture functional requirements of a system
- Document design details of a system

Use Case

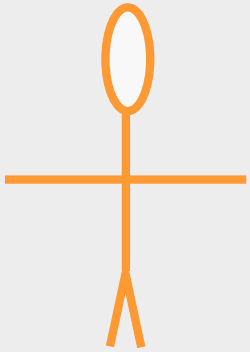
- An Use Case can be defined as a set of activities performed within a system by a User

- Each Use Case:
  - describes one logical interaction between the Actor and the system
  - defines what has changed by the interaction

Use Case Diagrams

Use Case Diagrams model the functionality of system by using Actors and Use Cases:

- Actor is a user of the system
- Use cases are services or functions provided by the system to its users



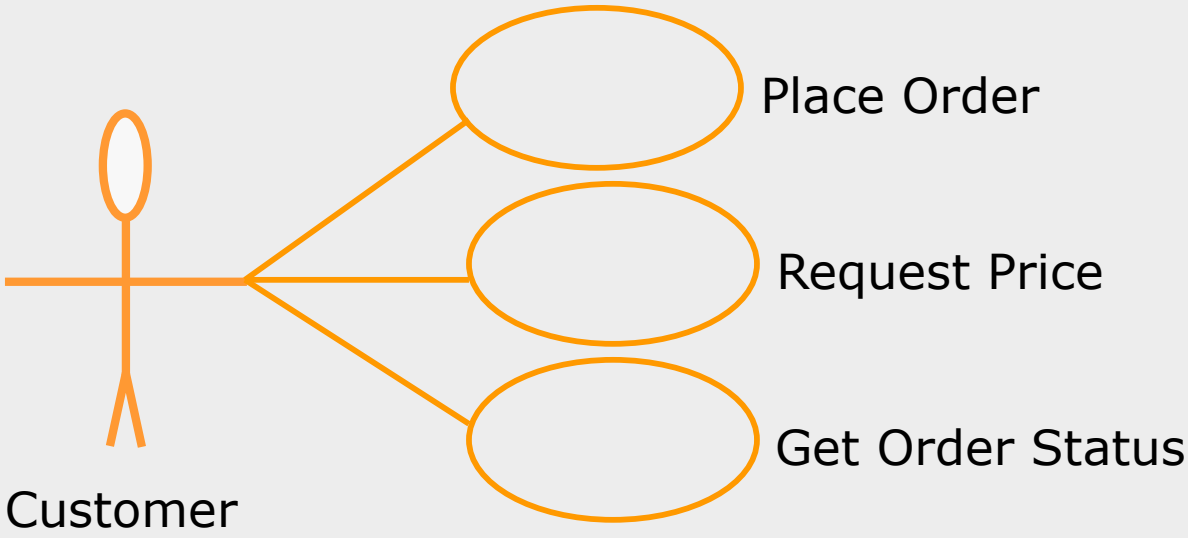
Actor

Use Case

Drawing The Use Case Diagram

A Use Case diagram has the following elements:

- Stick figure: representing an Actor
- Oval: representing a Use Case
- Association lines: representing communication between Actors and Use Cases



Advantages

- Easy to write
- Expressed in simple language that user understands
- Can be visually represented
- Can be used for validation (acceptance testing)
- Tools support
- Ideal for OO development
- Function oriented system (different types of users, complex behavioral patterns)

When Not to use case?

- System has few users
- System dominated by non-functional requirements

7.3: Actor

Definition

Actor:

- An Actor can be described as follows:
  - Actor is any entity that is external to the system and directly interacts with the system, thus deriving some benefit from the interaction
  - Actor can be a human being, a machine, or a software
  - Actor is a role that a particular user plays while interacting with the system
  - Examples of Actors are End-user (roles), External systems, and External passive objects (entities)

Type of Actors:

Primary Actor

- Initiates the Use Case
- Calls on the system to deliver a service
- Has a goal with respect to the system

Supporting Actor

- provides a service to the system under design

Use Case: Actors and goals

A GOAL is:

- What an ACTOR wants to get with the help of the system

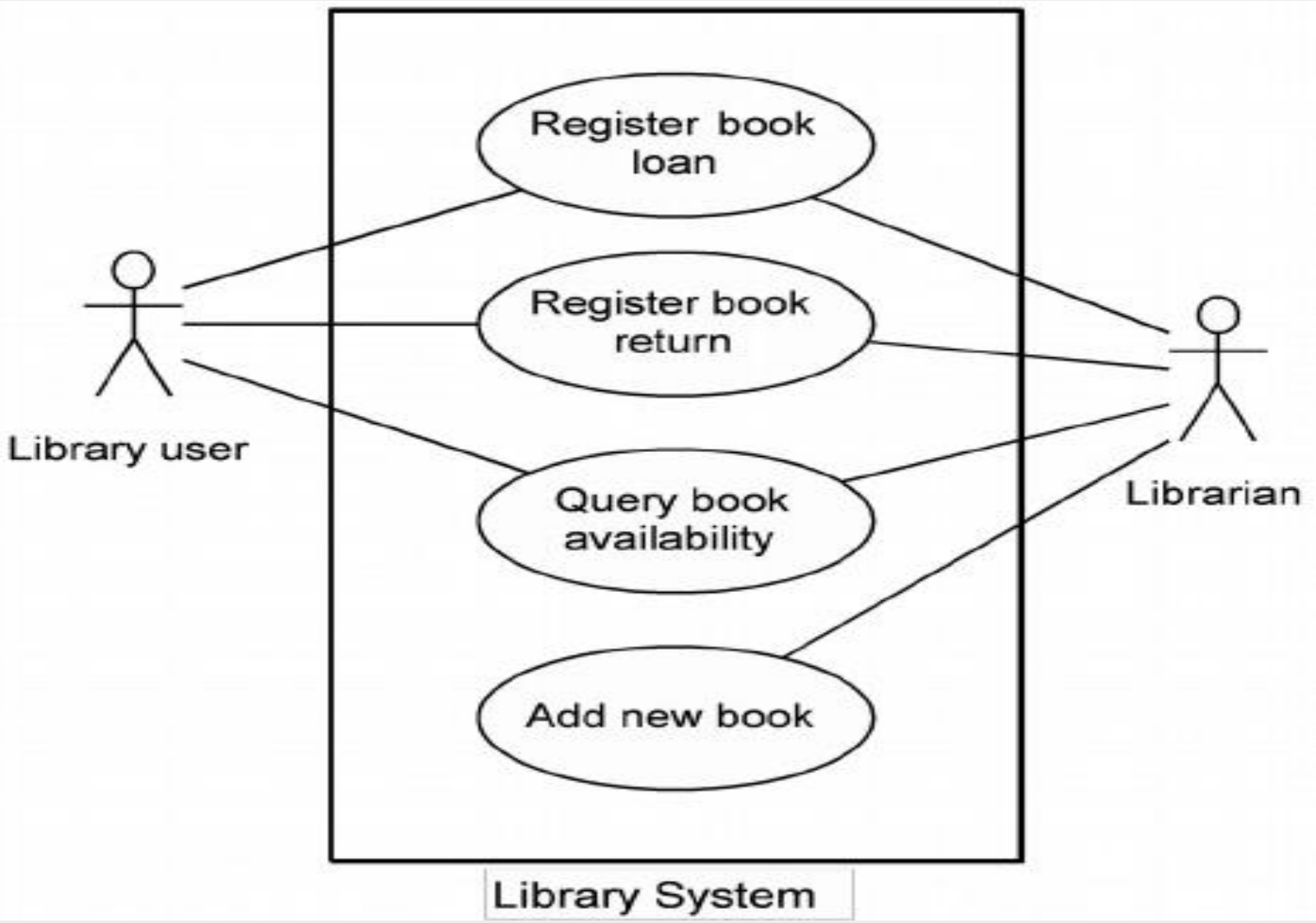
An ACTION is

- what the ACTOR must perform to reach the GOAL

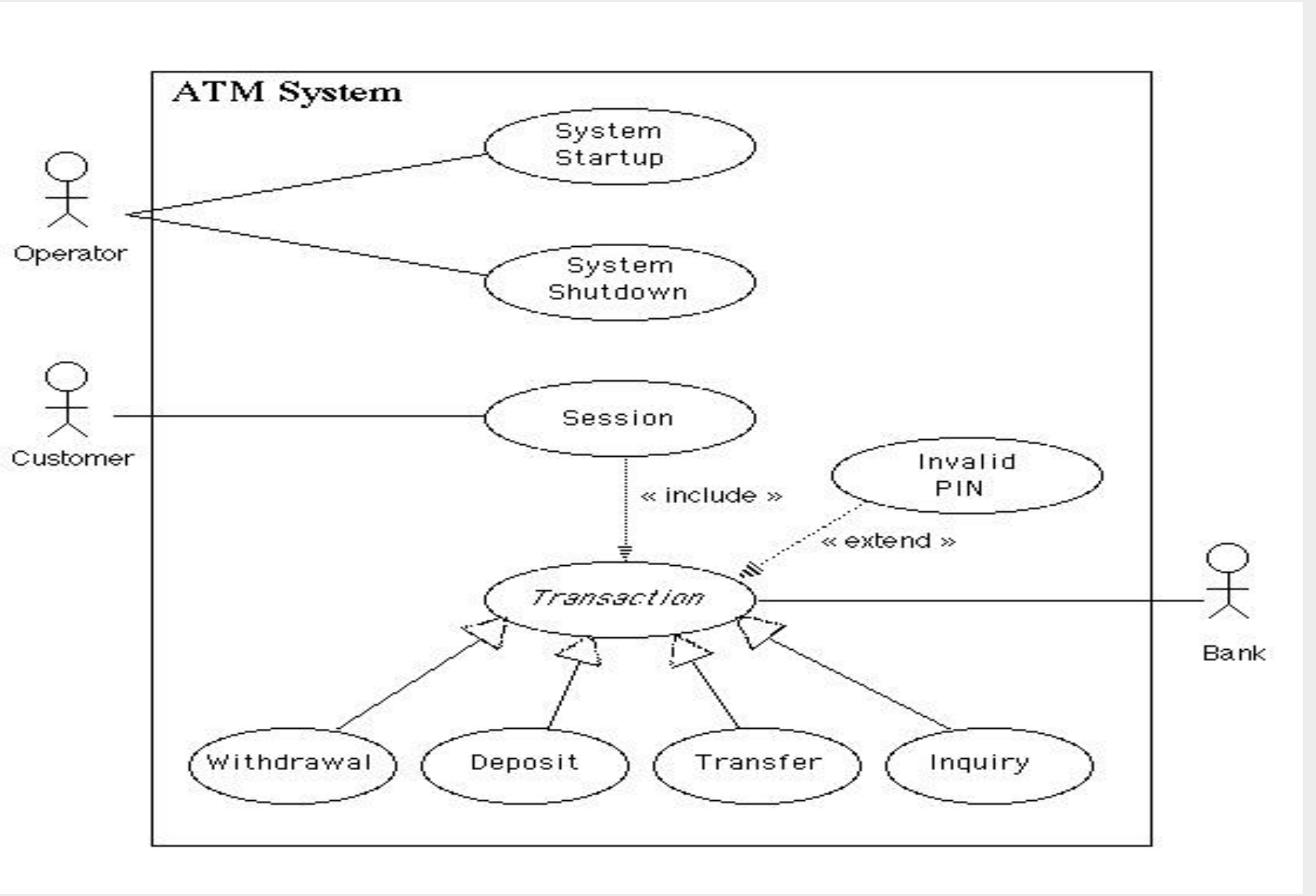
An INTERACTION is

- a sequence of steps that must be followed to complete the ACTION

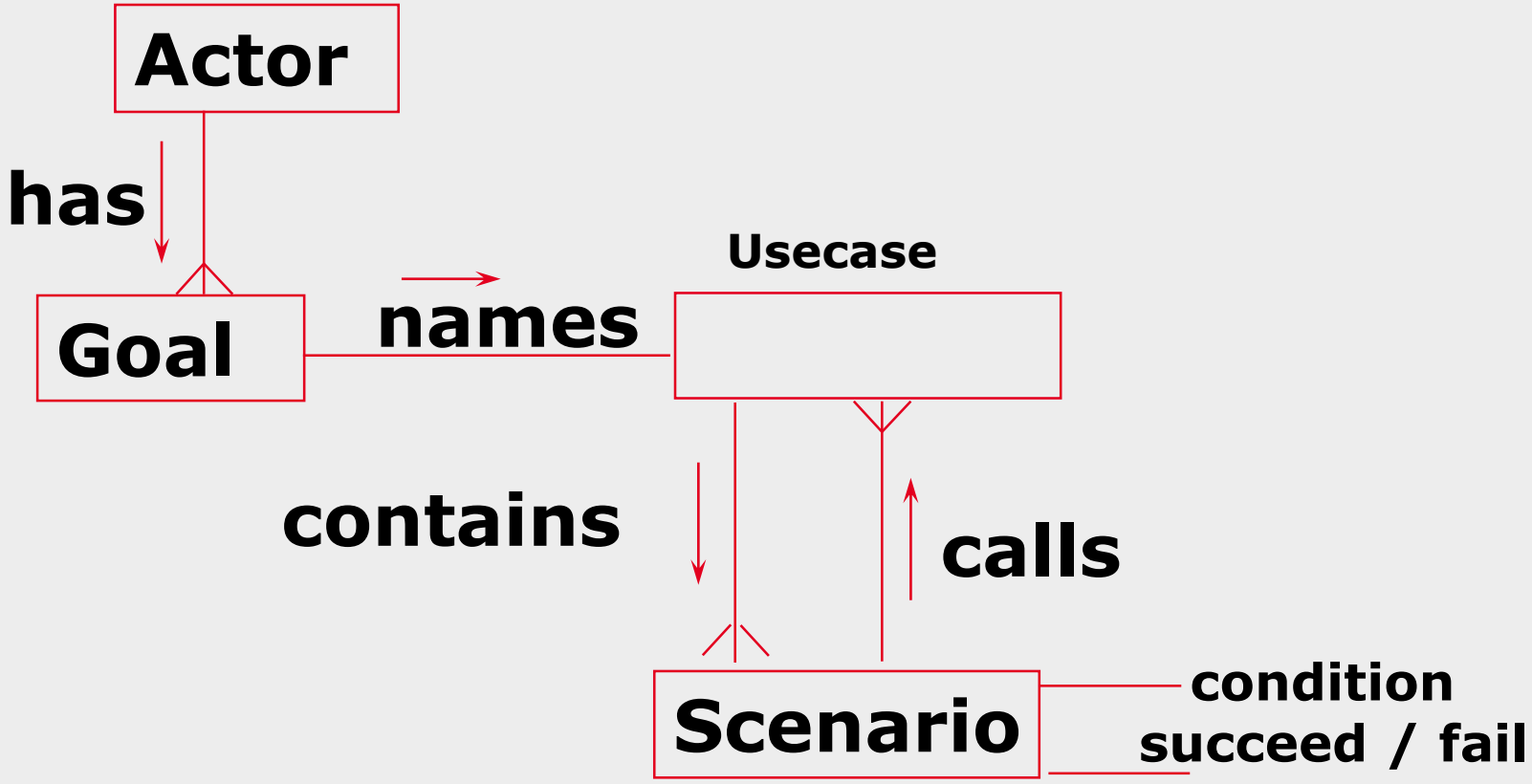
Use Case Diagram –Example 1



Use Case Diagram –Example 2



Use Cases: Actors And Goals



Use Cases: Goals And Requirements

Examining the Goals the system supports makes good functional requirements.

- “Place an order.”
- “Get money from my bank account.”
- “Get a quote.”
- “Find the most attractive alternative.”
- “Set up an advertising program.”

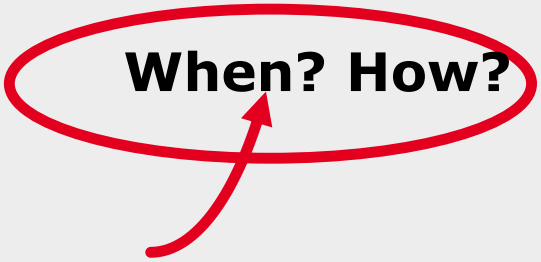
Goals summarize system function in understandable, verifiable terms of use.

Use Cases: Goals And Requirements

Structured narrative keeps the context visible

Just paragraphs:

- "ATM system has in interface with Bank database. It verifies the customer details and provides options. …"



With structured narrative:

- "Customer swipes the ATM card and enters the pin number. ATM system has an interface with Bank database to verify the customer pin. The system displays options to the customer

Goals And Scenarios

A use case pulls goals and scenarios together

In Use Case " Withdraw money from ATM"

- Scenario 1: Everything works well ...
- Scenario 2: Customer enters invalid pin …
- Scenario 3: Insufficient balance ...

Use case is goal statement plus the scenarios.

Note the grammar for Use Case: active verb first

Use Cases: Naming Actors

- Group individuals according to their common use of the system
- Identify the roles they take on when they use the system
- Each role is a potential actor
- Name each role and define its distinguishing characteristics
- Do not equate job title with role name. Roles cut across job titles
- Use the common name for an existing system; don’t invent a new name to match its role

Use Cases: Alternative Paths

For each significant action:

- Is there another significant way to accomplish it that could be taken at this point? (Variation)
- Is there something that could go wrong? (Exception)
- Is there something that could go really, really wrong (Error)

Use Cases: Exceptions

Exceptions are deviations from the typical case that happen during the normal course of events

- They should be handled, not ignored
- How to resolve them can be open to debate

What if a user mistypes her password?

What if an order can’t be fulfilled?

What if a connection to a web browser is dropped?

Use Cases: Errors

Errors are when things unexpectedly go wrong. They can result from malformed data, bad programs or logic errors, or broken hardware

- Little can be done easily to “fix things up and proceed”
- Recovery requires drastic measures

What if the disk is full?

What if equipment cannot be provisioned?

What if the OS crashes?

Use Cases: Precondition & Postcondition

Precondition

- is constraint not the event that starts the use case
- state what must always be true before beginning a scenario in use case are not tested within the use case; rather, they are conditions that are assumed to be true
- communicate noteworthy assumptions that the use case writer thinks readers should be alerted to condition to start an execution of test case

Use Cases: Precondition & Postcondition

Post conditions

- state what must be true on successful completion of the use case—either the main success scenario or some alternate path
- should be true regardless of which alternative flows were executed
- can be a powerful tool for describing use cases
- First define what the use case is supposed to achieve, the post condition

Use Cases: Good Practices

- Make the use cases clear, short and easy to read
- Use active voice, present tense, make sure actor and actor's intent are visible in each step
- Every Use case has two possible endings: Success and Failure / alternate courses. Gather both
- Create a list of primary actors and their goals (actor-goal list)
- Restrict use cases to capture system behavior….use cases are not suitable for other type of requirements

Failure Scenarios

Don’t overlook failure scenarios

- “What if their credit is too low?”
- “What if they run over the extended credit limit?”
- “What if we cannot deliver the quantity?”
- “What if data is corrupt in the database?”
- (These are commonly overlooked situations)

Summary

In this lesson, you have learnt:

- The use case model provides a complete, black-box, outside-in view of system functionality.
- A use case is a procedure by which an actor may use the system.
- A good use case is largely a sequence of interactions across the system boundary between the actor(s) and the system written in easy to understand natural language.
- An actor is always outside the system being defined
- The actor that starts the use case is the primary actor for that use case
- An actor is a role defined by the set of use cases with which it is associated

Review Question

- Question 1: _____ is very effective elicitation technique.

- Question 2: Actors live outside the boundary of the system.

  Option: True / False

- Question 3: Postconditions are basically conditions to start an execution of test case.

  Option: True / False

Review Question: Match the Following

1.Precondition	A. Non-recoverable extensions
2. Actor	B. Expectedly go wrong
3. Errors	C. These are not tested within the use case
4. Post condition	D. Unexpectedly go wrong
5. Exception	E. Must be true on successful completion of the use case
	F. Live outside the boundary of the system