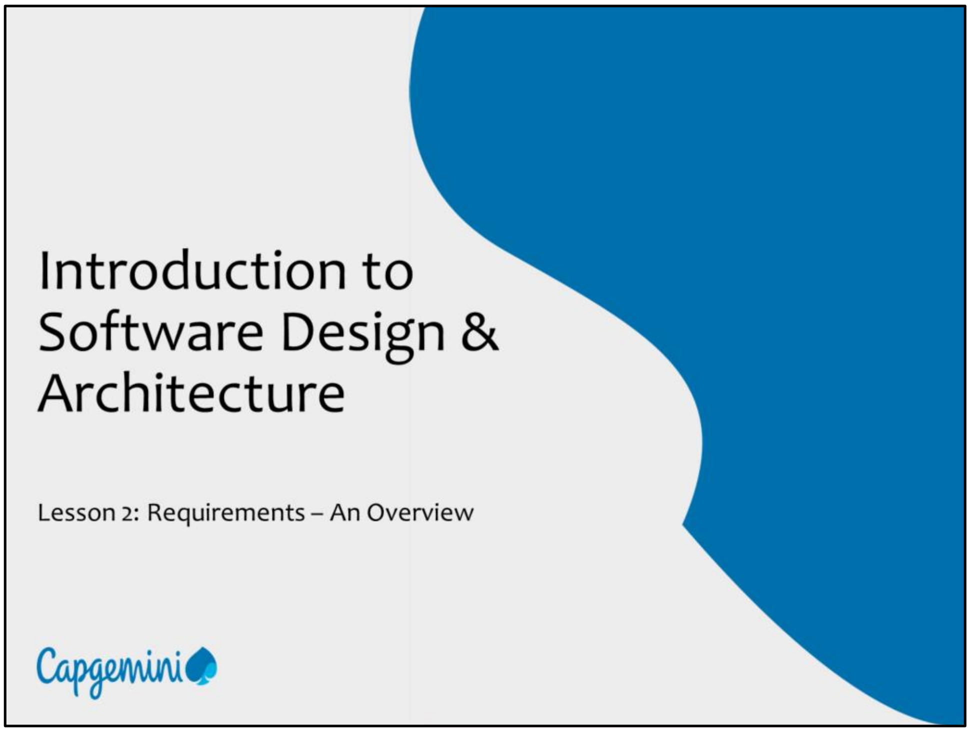
Instructor Notes:

Add instructor notes here.



Introduction to Software Design & Architecture

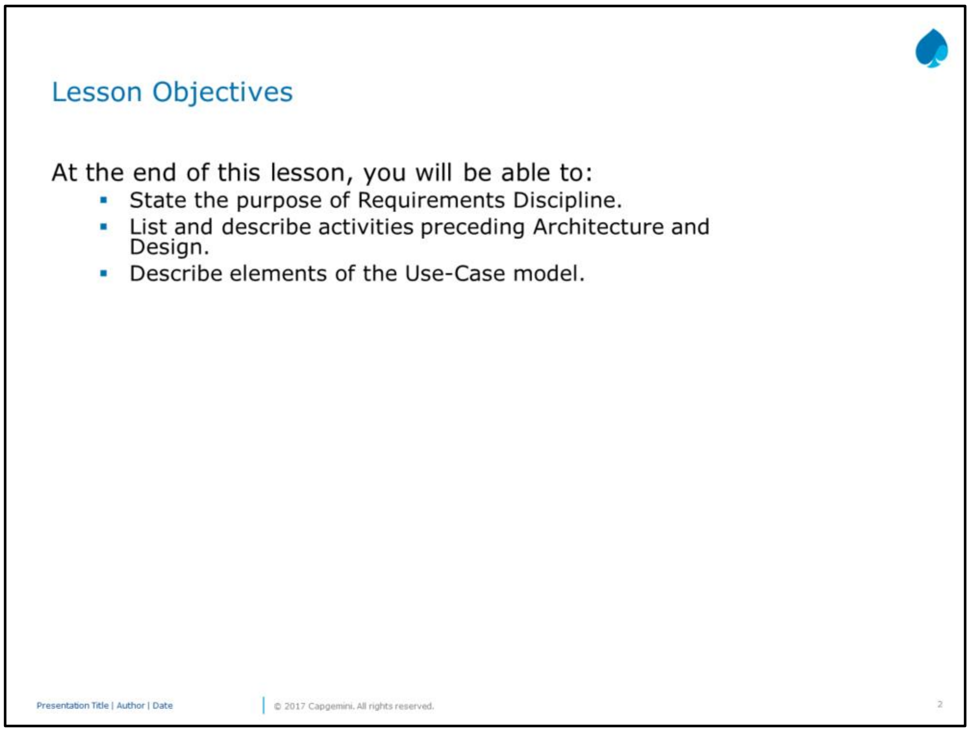
Lesson 2: Requirements – An Overview

Capgemini

Instructor Notes:

The steps of requirements discipline precede the steps of architecture and design. This lesson gives an overview of the Requirements Discipline, especially the Use Case Model which is a key input for analysis and design.

The focus is on understanding the outputs of Requirements Discipline, not creating them.



Lesson Objectives

At the end of this lesson, you will be able to:

- State the purpose of Requirements Discipline.
- List and describe activities preceding Architecture and Design.
- Describe elements of the Use-Case model.

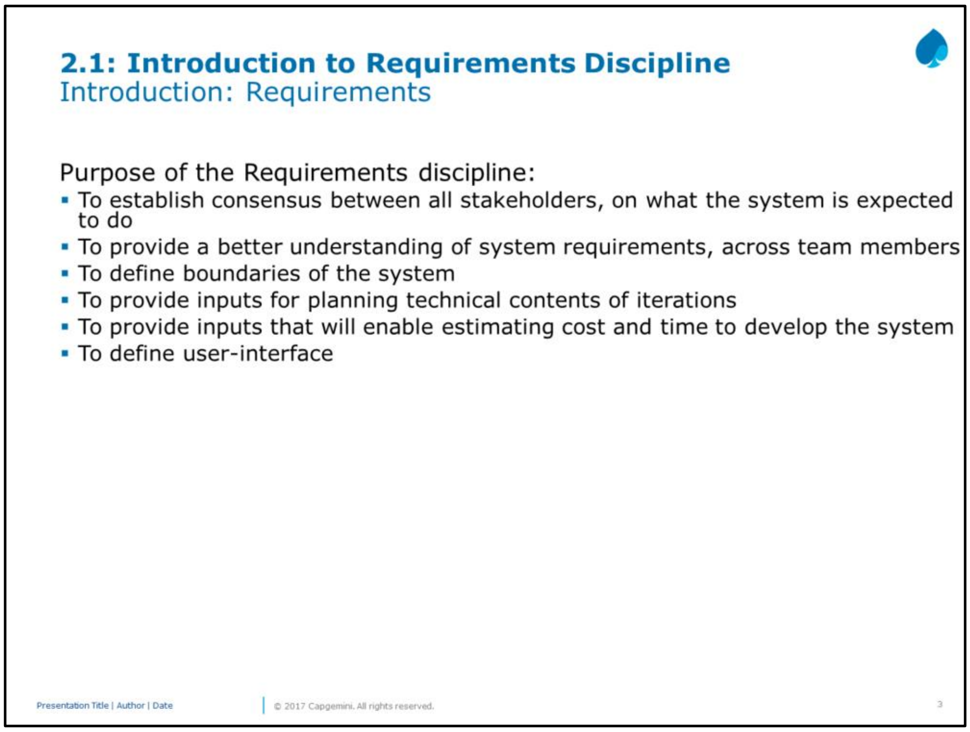
Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 2

<u>Lesson Objectives</u>:

- This lesson aims at providing an overview of the key artifacts of the **Requirements discipline** that are used in **Architecture and Design**.
- This module will also bring into perspective the close link between "**Requirements**" and "**Architecture and Design**" disciplines.

Instructor Notes:

Focus on system boundaries since it defines the scope of the system.



2.1: Introduction to Requirements Discipline
Introduction: Requirements

Purpose of the Requirements discipline:
- To establish consensus between all stakeholders, on what the system is expected to do
- To provide a better understanding of system requirements, across team members
- To define boundaries of the system
- To provide inputs for planning technical contents of iterations
- To provide inputs that will enable estimating cost and time to develop the system
- To define user-interface

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 3

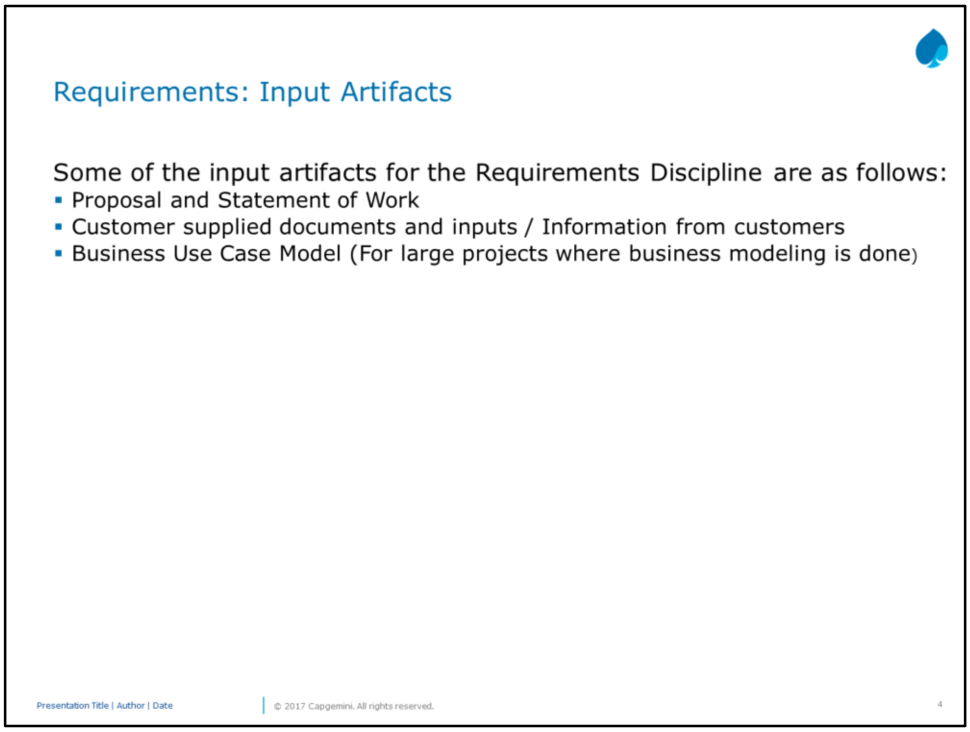
<u>Introduction to Requirements Discipline</u>:

- As part of the Requirements discipline activity, we look for the following:
  - ➢ Answering what the new system is expected to do? The answer has to be understood across team members.
  - ➢ Defining system requirements in terms that are easily understood across team members
  - ➢ Specifying the boundaries, that is, the scope of the system (what the system should do and what it should not do)
  - ➢ Providing inputs that will help in designing the technical contents of iterations
  - ➢ Providing inputs that will help in estimating expenditure in terms of cost and time
  - ➢ Defining an user-interface keeping in view user needs and goals
- Artifacts, which are produced by the Requirements discipline, serve as inputs for the Architecture and Design discipline. Errors made in Use-Case models are identified during Architecture and Design and a change request process is immediately initiated. This ensures early detection and rectification of errors.

Instructor Notes:

Business modeling is not done much in the Patni context.

A business use case model sets the context which helps understand the business. It specifies the business workflows which can then be used to derive requirements for the system.



Requirements: Input Artifacts

Some of the input artifacts for the Requirements Discipline are as follows:
- Proposal and Statement of Work
- Customer supplied documents and inputs / Information from customers
- Business Use Case Model (For large projects where business modeling is done)

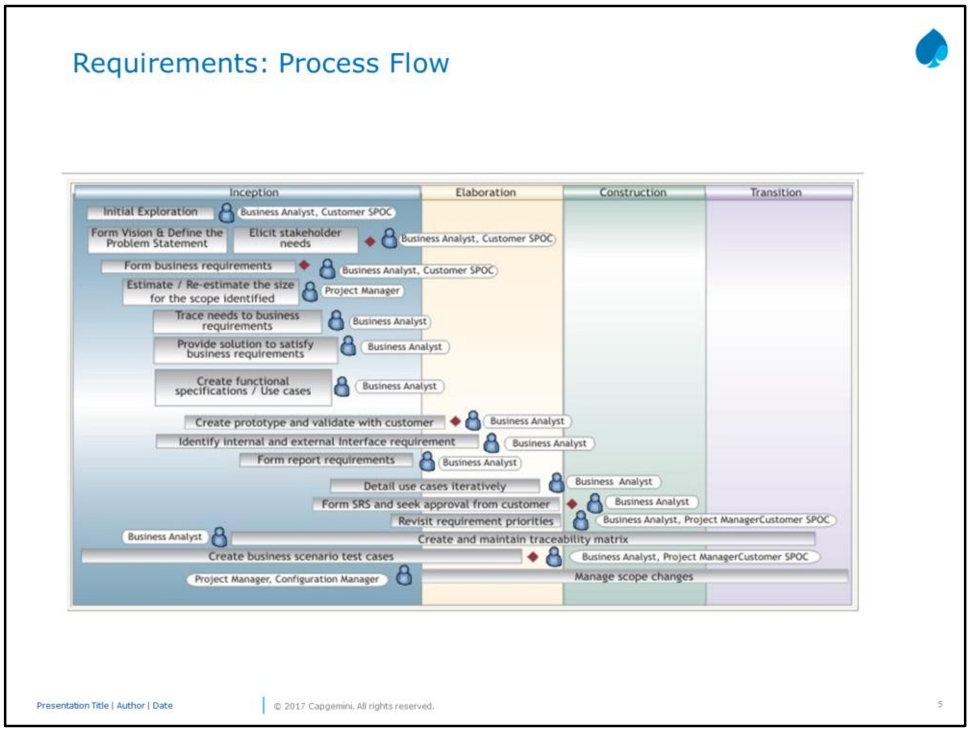
Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 4

Requirements: Input Artifacts:
- All artifacts such as proposals, SOWs, customer provided documents, and information will form an input to the Requirements Discipline.
- For large projects, business modeling activity precedes the Requirements Discipline. In that case, outputs of the Business Modeling discipline, such as the Business Use Case Model, form an input to the Requirements Discipline.

Instructor Notes:

This is to be only briefly touched upon.

Refer to Qzen pages on Requirements discipline for more explanations.


Inception
Elaboration
Construction
Transition
Initial Exploration
Business Analyst, Customer SPOC
Form Vision & Define the Problem Statement
Elicit stakeholder needs
Business Analyst, Customer SPOC
Form business requirements
Business Analyst, Customer SPOC
Estimate / Re-estimate the size for the scope identified
Project Manager
Trace needs to business requirements
Business Analyst
Provide solution to satisfy business requirements
Business Analyst
Create functional specifications / Use cases
Business Analyst
Create prototype and validate with customer
Business Analyst
Identify internal and external interface requirement
Business Analyst
Form report requirements
Business Analyst
Detail use cases iteratively
Business Analyst
Form SRS and seek approval from customer
Business Analyst
Revisit requirement priorities
Business Analyst, Project ManagerCustomer SPOC
Business Analyst
Create and maintain traceability matrix
Create business scenario test cases
Business Analyst, Project ManagerCustomer SPOC
Project Manager, Configuration Manager
Manage scope changes

Requirements: Process Flow

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

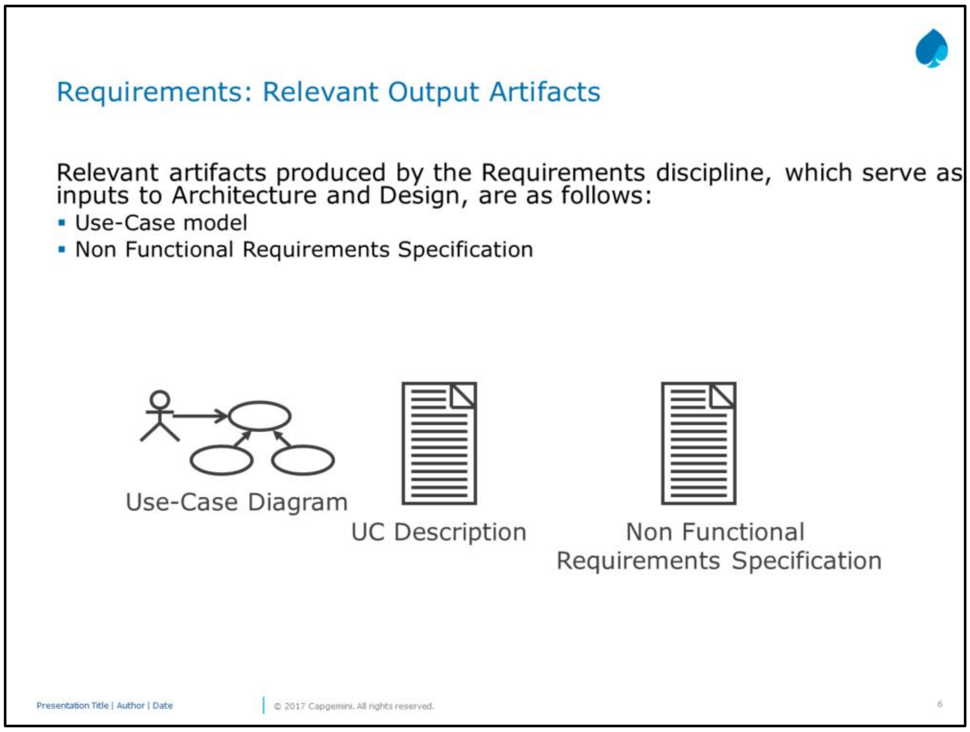
5

Requirements: Process Flow:

- The process flow defined for Requirements discipline in Qzen is illustrated in the above slide.
- It involves the following:
  - ➢ Understanding the customer requirements and expectations
  - ➢ Coming up with the system requirements, both in terms of functional and non-functional requirements
  - ➢ Maintaining traceability for the requirements
  - ➢ Critical reviews and sign offs from customers at key milestones (this being the most important aspect)

Instructor Notes:

Functional requirements in the form of use cases talk about features needed in the system; while non-functional requirements capture the aspects such as usability, reliability, performance and supportability. (Acronym FURPS+ for Functionality and other Non Functional Requirements listed above. "+" indicates there can be many such requirements such as regulatory requirements, security etc.)



Requirements: Relevant Output Artifacts

Relevant artifacts produced by the Requirements discipline, which serve as inputs to Architecture and Design, are as follows:
- Use-Case model
- Non Functional Requirements Specification

Use-Case Diagram
UC Description
Non Functional Requirements Specification

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 6

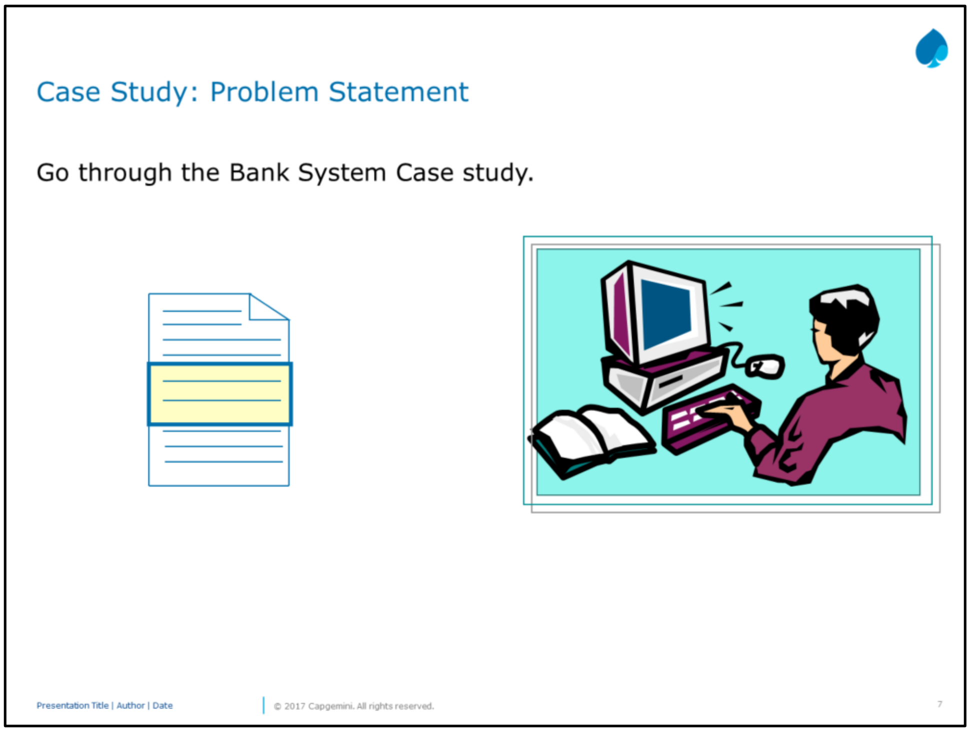
__Requirements: Relevant Output Artifacts__:

- Of the several artifacts that are produced by the **Requirements discipline**, the ones that are most relevant to the **Architecture and Design discipline** are as follows:
    - **Use-Case Model:**
        - ➢ The Use-Case model defines what the system will do.
        - ➢ It also serves as a method of communication across team members. The use-case model consists of actors and use cases. It gives a step-by-step description of how the system interacts with the actors and what the system does in the use-case.
        - ➢ The use-Case specification is also produced, which is a document that lists the use-case properties.
    - **Non Functional Requirements Specification:**
        - ➢ These contain the non-functional requirements for the system. A supplementary specification document can exist that includes these non-functional requirements as well as functional requirements, which cannot be defined as a use-case model but are complementary to the use-case model.
        - ➢ In many cases, a detailed UI specification can also get created and this too forms a useful input for the Architecture and Design discipline.

Instructor Notes:

Introduce and briefly explain the case study.

Through out the requirements and design steps, it is important to indicate to the participants that there are assumptions that we have made based on which we have arrived at certain set of UML diagrams. The focus is not on domain, so we may not have necessarily stuck to the "right" way a functionality may be getting implemented in a domain.



Case Study: Problem Statement

Go through the Bank System Case study.

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

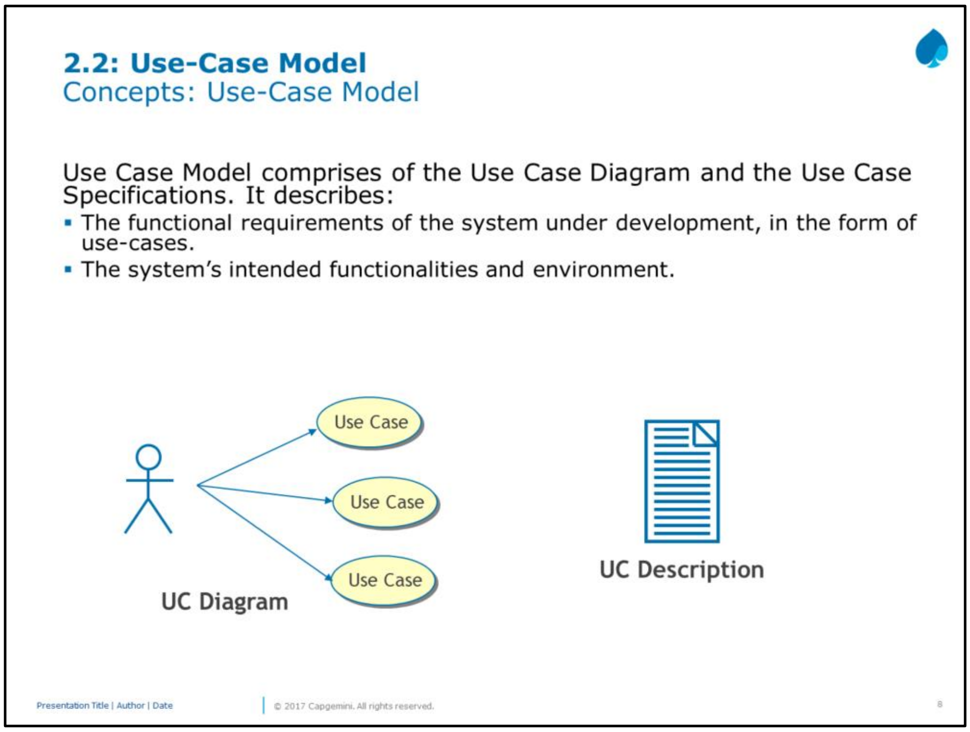
7

Case Study: Problem Statement:

- In order to make concepts and procedures easy and simple to understand, a specific case study has been taken up. Henceforth all exercises will be based on this problem domain.
- Refer to Appendix 1 for the problem statement of the case study.

Instructor Notes:

Important to emphasize that Use Case Model is not just the diagram part but also the text in the form of use case descriptions, which complements the diagram



2.2: Use-Case Model
Concepts: Use-Case Model

Use Case Model comprises of the Use Case Diagram and the Use Case Specifications. It describes:

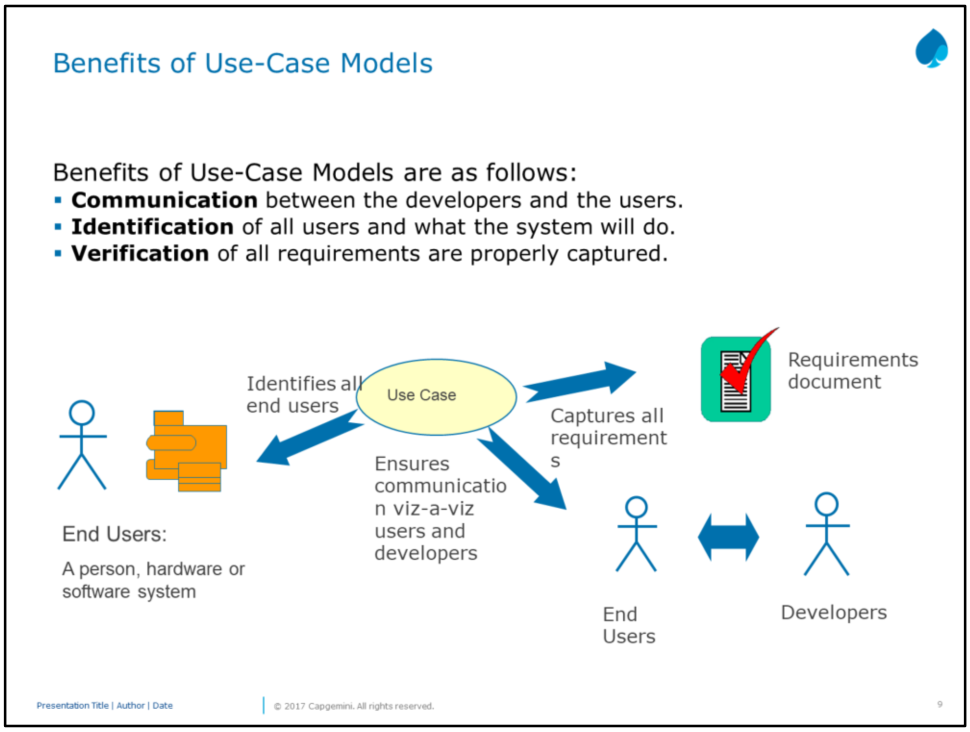
- The functional requirements of the system under development, in the form of use-cases.
- The system's intended functionalities and environment.

<u>Use-Case Model</u>:

- A **Use-Case Model** describes a system's functional requirements in terms of use cases. It is a model of the system's intended functionality and its environment.
- The **use-case model** serves as a **contract** between the customer and the developers. This model gets used in all phases of the development cycle.

Instructor Notes:

Use Case Model is central to the entire system development. It not only captures the requirements, but also drives the iterations based on prioritized use cases, drives design based on use cases and testing.



Benefits of Use-Case Models

Benefits of Use-Case Models are as follows:

- **Communication** between the developers and the users.
- **Identification** of all users and what the system will do.
- **Verification** of all requirements are properly captured.

Use Case
Identifies all end users
Captures all requirements
Ensures communication viz-a-viz users and developers
Requirements document
End Users:
A person, hardware or software system
End Users
Developers

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

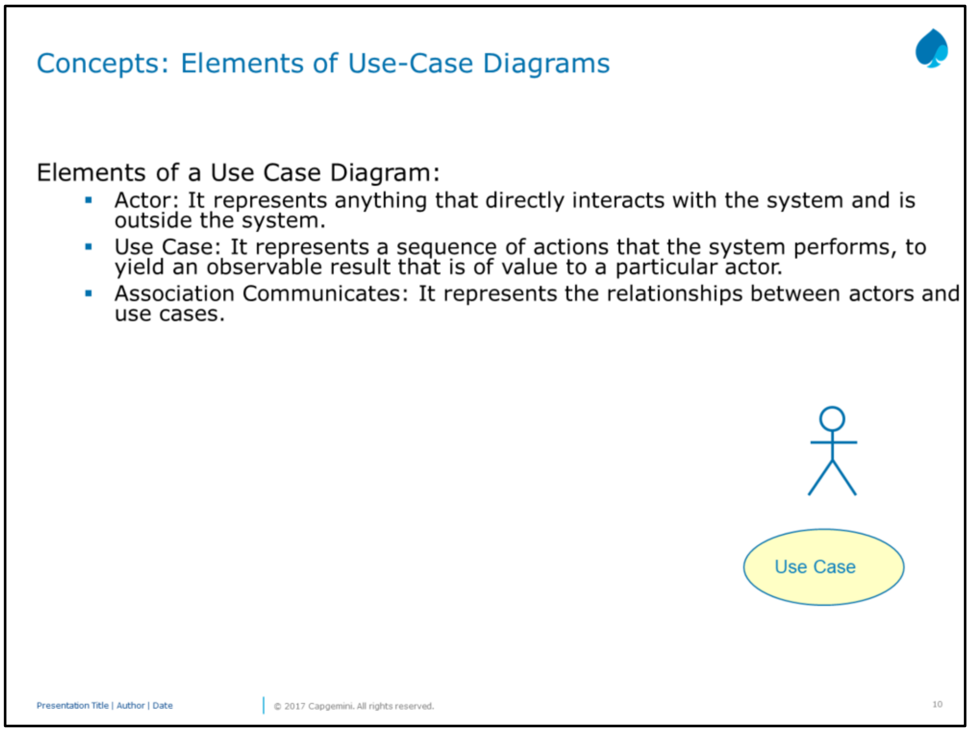
9

<u>Benefits of Use-Case Models</u>:

- The Use-Case model is a sort of blueprint on which the system is modeled. It is the device that provides the link between users, developers, and domain experts. Hence the use-case model must be simple, easy-to-understand, and encompassing all requirements.
- A use-case model must act as:
  - A **communicating agent** between the users and developers and domain experts.
  - An **identifying agent** by identifying all possible users of the new system
  - A **verifying agent**, by ensuring that all needs are identified and understood by the development team.
  - An **Identifying agent**, by identifying all actors that interact with the system. Actors help delimit the system and give a clearer picture of what it is supposed to do.

Instructor Notes:

Explain in the context of some examples from the case study. Encourage participants to identify actors and use cases for the case study.



Concepts: Elements of Use-Case Diagrams

Elements of a Use Case Diagram:

- Actor: It represents anything that directly interacts with the system and is outside the system.
- Use Case: It represents a sequence of actions that the system performs, to yield an observable result that is of value to a particular actor.
- Association Communicates: It represents the relationships between actors and use cases.

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

10

Concepts: Elements of Use-Case Diagrams:

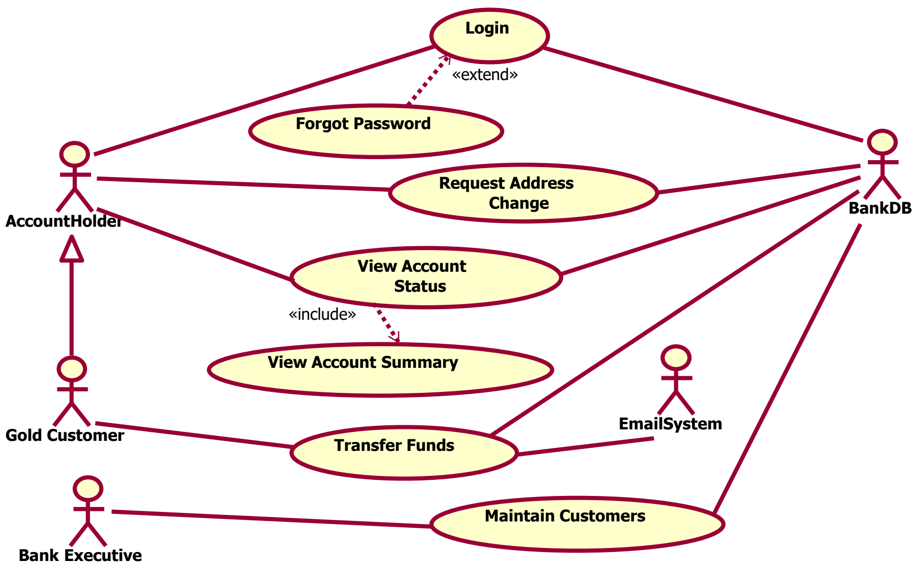
- **Actor**: It is each and every set of roles that users play when interacting with the system. An actor can be a person, a hardware device, or another software system. An actor should be directly interacting with the system, and gain something of value from the interaction. An actor is represented as a stick figure.
- **Use case**: It defines what a system does. However, it does not define how it does it. A use-case is a representation of a set of events that occur when an actor uses the system to complete a process. Usually a use-case is a relatively long process, and not just an individual step or transaction. The use case captures the interaction between the actor and the system for accomplishing a functionality.
- **Association**: This is a line between the actor and the use case that represent the relationship between the actor and the use case.

Instructor Notes:

Apart from relationship between actor and use case, there can also be relationship between actors indicated by generalization relationship and relationship between use cases as include and extend relationships.

Include indicates mandatory behaviour where included use case is always done as part of the other use case. Extend on the other hands indicates optional behaviour, where the extended use case is done optionally based on some condition.

Case Study: Use-Case Diagram:

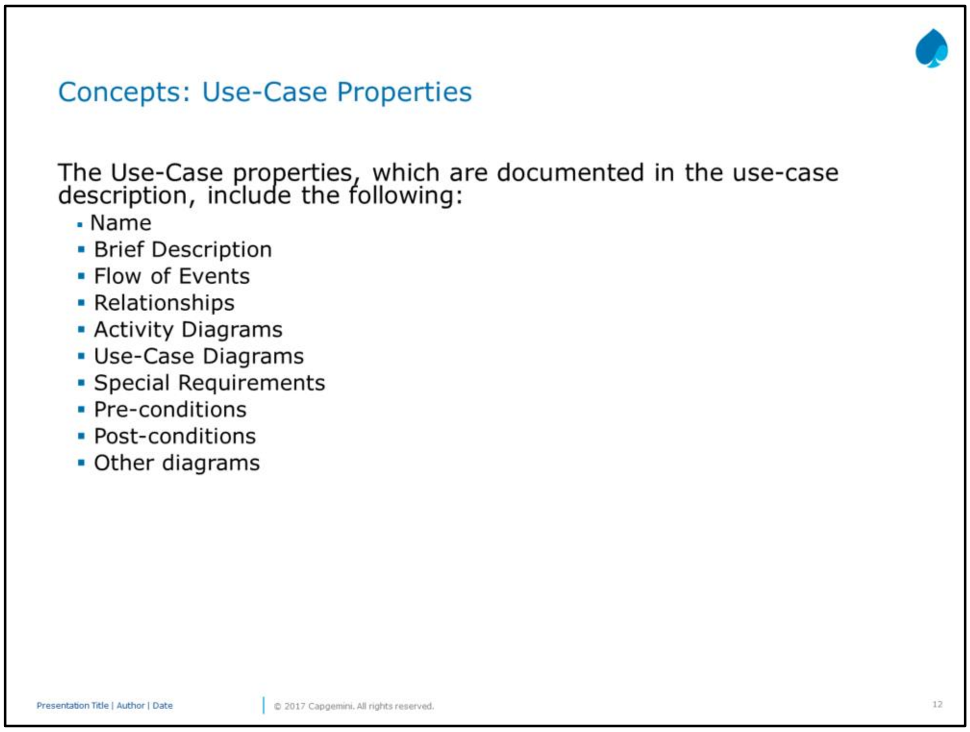


Example: Use-Case Diagram for Banking System:

- Here is an example of a use case diagram for a Banking System.
  - ➢ UML Notation for an Actor is a stick figure.
    - ▪ Account Holder, Gold Customer, Bank Executive, BankDB, and EmailSystem are identified actors of this system.
    - ▪ Gold Customer is an Account Holder with special privileges and is hence indicated with a generalization relationship.
  - ➢ UML Notation for a Use Case is an oval.
    - ▪ Login, Request Address Change, Transfer Funds, Maintain Customers, etc. are use cases.
    - ▪ The use case Forgot Password extends the Login use case and would be invoked when the Account Holder clicks the link for forgot password.
    - ▪ The use case View Account Summary is included as part of the use case View Account Status.

Instructor Notes:

Note that UML does not have a standard format for capturing use case descriptions. But most of the properties indicated here are captured in a use case description document.



Concepts: Use-Case Properties

The Use-Case properties, which are documented in the use-case description, include the following:

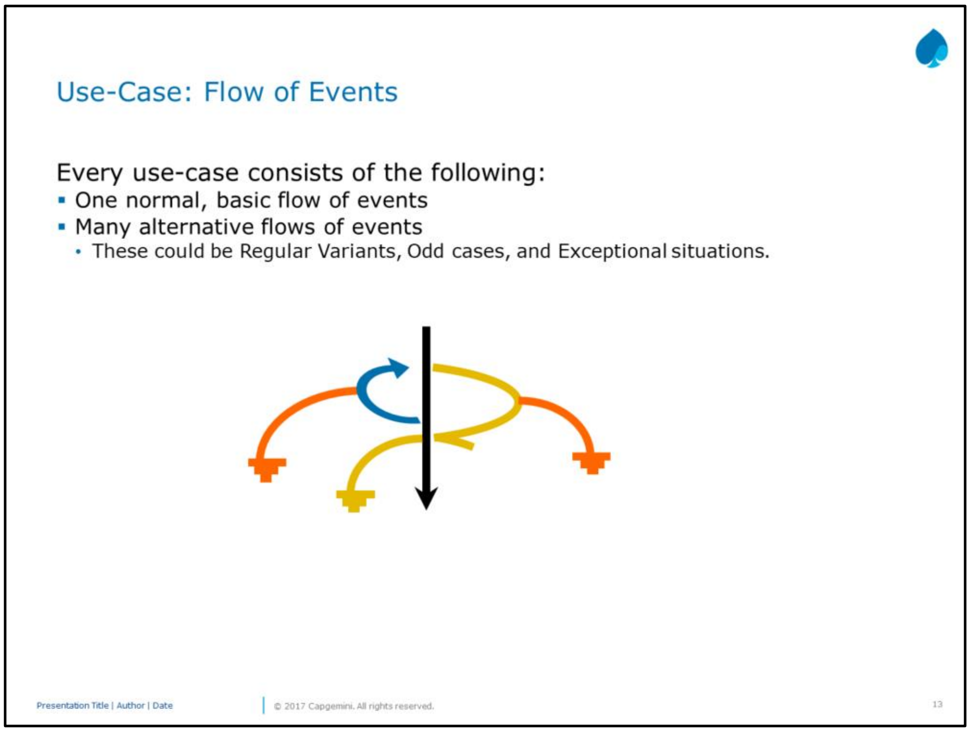
- Name
- Brief Description
- Flow of Events
- Relationships
- Activity Diagrams
- Use-Case Diagrams
- Special Requirements
- Pre-conditions
- Post-conditions
- Other diagrams

Concepts: Use-Case Properties:

- Use-case properties are as given below:
  - **Name:** It is the name of the use-case
  - **Brief Description:** It describes the role and its activity.
  - **Flow of events:** This gives a textual description of what the system does.
  - **Relationships:** It depicts associations, that is, Includes and extends relationships, that the use-case is involved in.
  - **Activity diagrams:** This provides a graphical depiction of the flow of events.
  - **Use-Case diagrams:** It graphically shows the relationships that the use case is involved in.
  - **Special Requirements:** It is a textual description of all non-functional requirements of a system. These are requirements that cannot be shown in the form of a use case.
  - **Pre-conditions:** They define conditions relating to starting of an use case.
  - **Post-conditions:** They define conditions relating to termination of an use case.
  - **Other diagrams:** They are hand-drawn diagrams or screen shots, from an user-interface prototype. They can be used to describe the use-case.

Instructor Notes:

Flow of events is the essential part of a use case description where the interaction between the actor and system is captured. “All is Well” in the basic flow. Alternate flows capture all other possibilities. Regular variant is a subflow (like add/modify/delete customer), odd case is something occurring once in a while (like an yearly interest calculation) and exceptional scenarios are the error conditions (invalid inputs or connection errors etc).



Use-Case: Flow of Events

Every use-case consists of the following:

- One normal, basic flow of events
- Many alternative flows of events
  - These could be Regular Variants, Odd cases, and Exceptional situations.

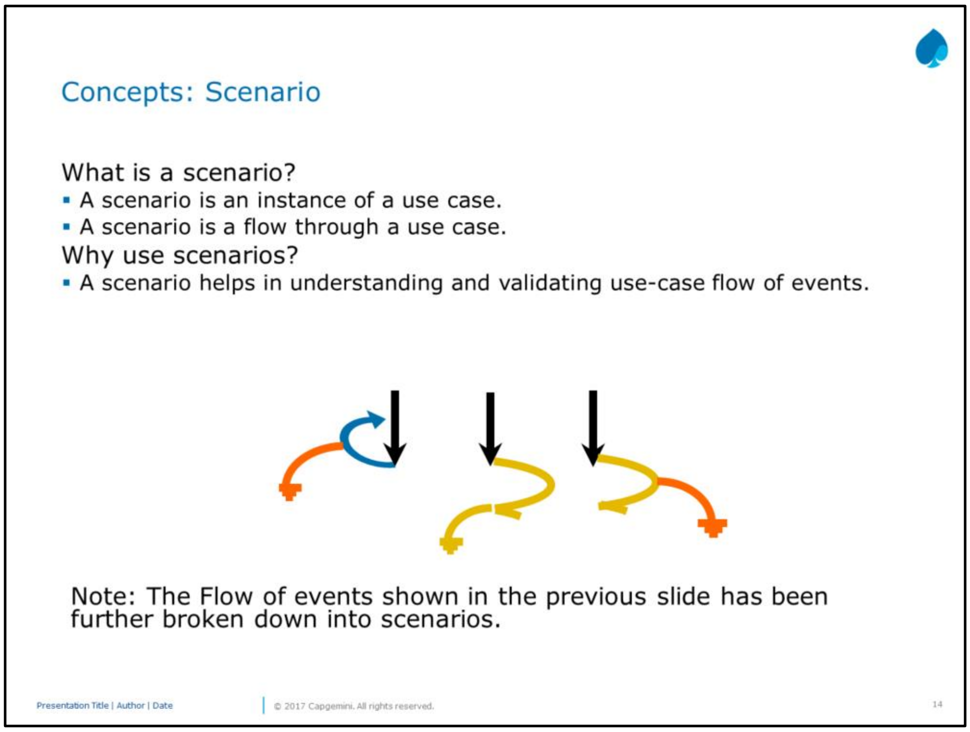
Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 13

<u>Use-Case: Flow of Events</u>:

- A use case flow of events describes what the system does.
- The main flow of events is the Basic Flow, or Happy Path. It is also called a Sunny Day scenario.
- A use case may also have several alternate paths to take care of the exceptions and other variants.

Instructor Notes:

Scenarios could be many since it is any combination of basic and alternate flows; hence it is not possible to capture all possible scenarios. However key scenarios can be captured.



Concepts: Scenario

What is a scenario?
- A scenario is an instance of a use case.
- A scenario is a flow through a use case.

Why use scenarios?
- A scenario helps in understanding and validating use-case flow of events.

Note: The Flow of events shown in the previous slide has been further broken down into scenarios.

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 14

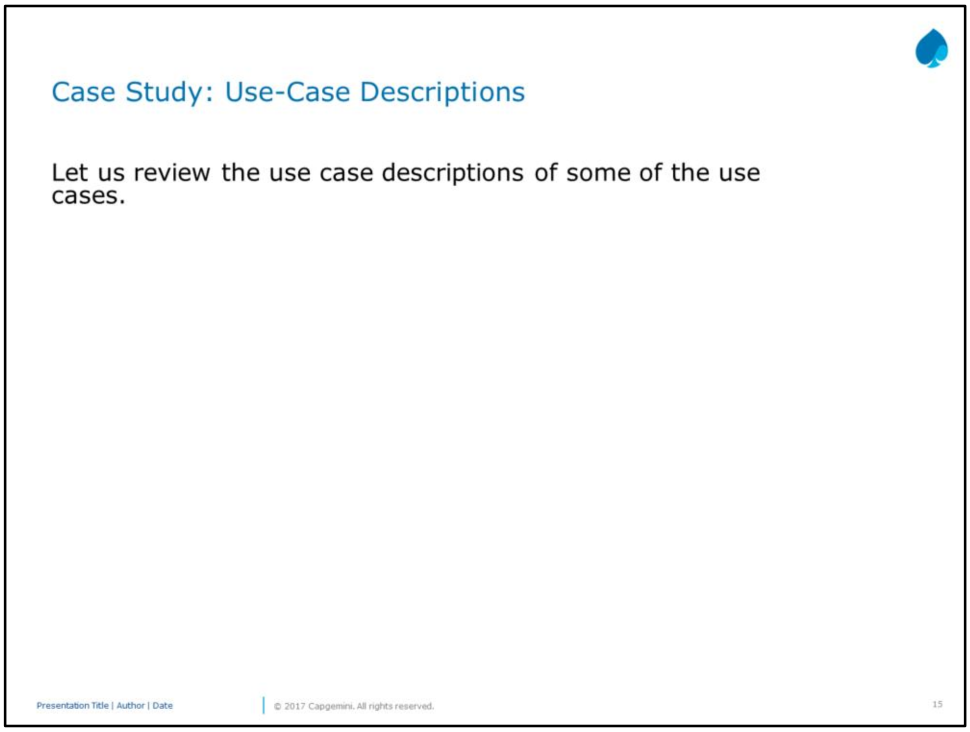
<u>Concepts: Scenario</u>:

- A scenario is a particular flow of events. Every use-case consists of many scenarios. Scenario is like a “situational description” that describes a system’s interaction with some quality attributes.
- Scenarios usually focus on three main problem areas:
  - **Use-Case scenarios:** These anticipate the various ways a system is likely to be used.
  - **Growth scenarios:** These anticipate the likely changes that the system will undergo.
  - **Exploratory scenarios:** These anticipate the stresses and changes the system is likely to bear.
- Scenarios usually consist of one basic flow and many alternative flows of events, in any number of combinations.
- Number of scenarios per use-case varies in keeping with the following principles:
  - There should be as many scenarios as will be required to understand the system under development.
  - Scenarios describing high-risk use-cases must be elaborate.
- Scenarios make excellent test cases.

Instructor Notes:

Walk through some use case descriptions focusing on different sections in the use case specification document.

Ideally the Requirements Team has to provide the Use Case Model to the Designer Team. It may not always happen in practice! So it would help that the designers spend some time thinking through the steps of the flow of events and document the use case description so that subsequent design steps are easier to handle.



Case Study: Use-Case Descriptions

Let us review the use case descriptions of some of the use cases.

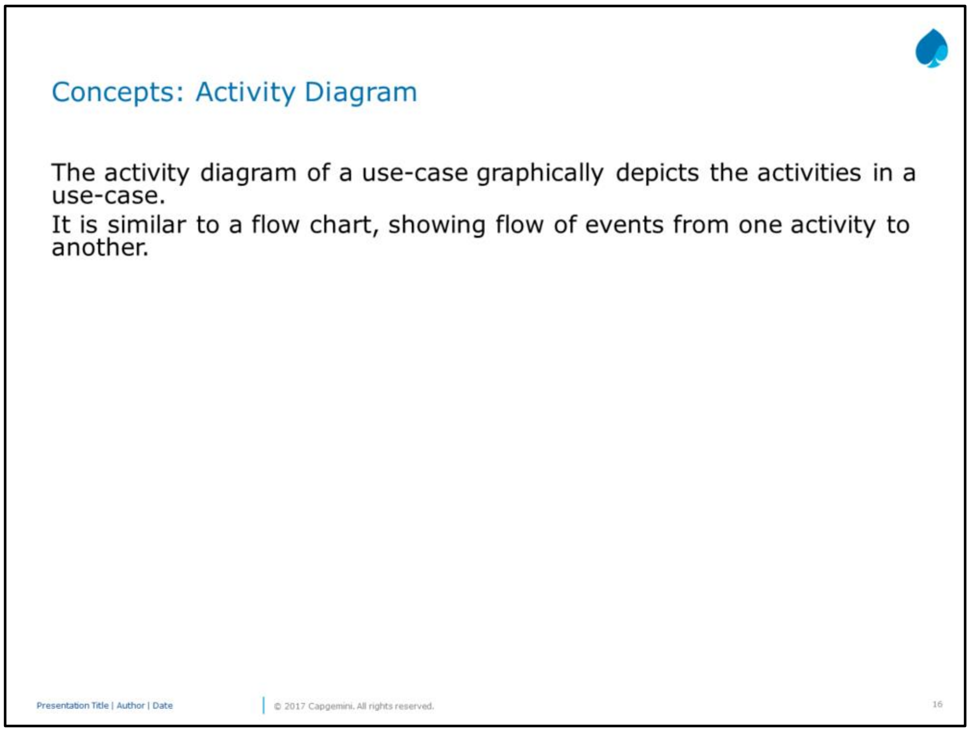
Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 15

<u>Case Study: Use-Case Descriptions</u>:

- Let us review the descriptions of some of the use cases.
- Refer Appendix 2 for the use case descriptions.

Instructor Notes:

The UML Activity Diagram is a very generic diagram that can be used at the level of the entire system; or to capture the flow of a use case or even to illustrate the logic within an operation. For the Requirements part, it can be used to capture the use case flow of events.



Concepts: Activity Diagram

The activity diagram of a use-case graphically depicts the activities in a use-case.

It is similar to a flow chart, showing flow of events from one activity to another.

Concepts: Activity Diagram:
Just as a **flowchart** can be used to graphically depict an **algorithm**, an **activity diagram** can be used to graphically represent the **flow of events** specified in the use case descriptions.

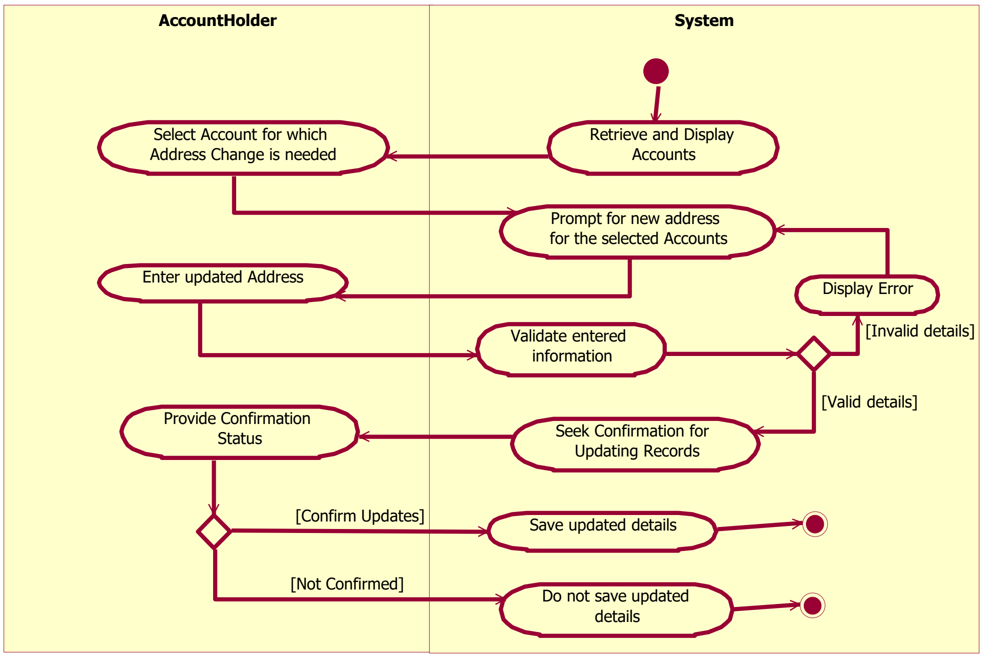
Instructor Notes:

Some other notations one might encounter on an activity diagram are:

1. Synchronization Bar, Fork: Indicates concurrent threads of activities after Fork.

2. Synchronization Bar, Join: Indicates that after all concurrent activities are completed, next activity can begin.

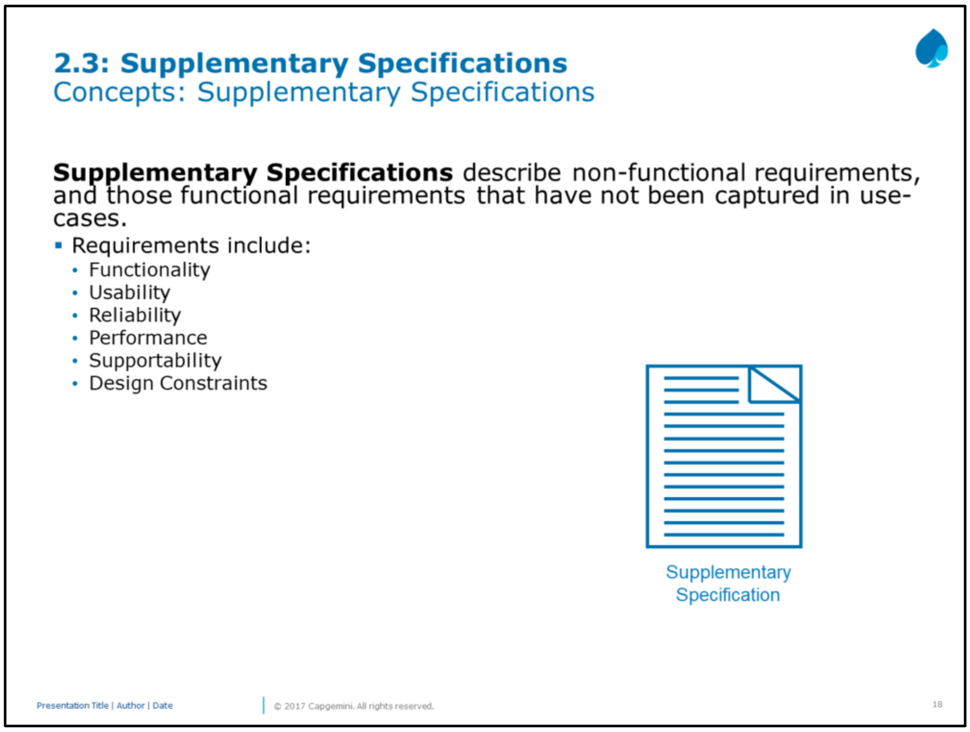
Case Study: Example of an Activity Diagram:


AccountHolder
System
Select Account for which Address Change is needed
Retrieve and Display Accounts
Prompt for new address for the selected Accounts
Enter updated Address
Display Error
Validate entered information
[Invalid details]
[Valid details]
Provide Confirmation Status
Seek Confirmation for Updating Records
[Confirm Updates]
Save updated details
[Not Confirmed]
Do not save updated details

- Here is an example of an activity diagram for the Request Address Change Use Case.
  - Activity Diagrams include the following:
    - One start state (represented as filled circles, this is where the flow begins)
    - One or more end states (represented with a concentric circle over filled circle, this is where the flow ends)
    - Activities
    - Decision boxes, etc.
  - Guard conditions are the conditions written on transition lines between activities.
  - Swimlanes are partitions that help understand who is responsible for each of the activities. Indicating swimlanes is optional.

Instructor Notes:

Acronym FURPS+ for Functionality and other Non Functional Requirements listed above. “+” indicates there can be many such requirements such as regulatory requirements, security etc.



2.3: Supplementary Specifications
Concepts: Supplementary Specifications

Supplementary Specifications describe non-functional requirements, and those functional requirements that have not been captured in use-cases.

- Requirements include:
  - Functionality
  - Usability
  - Reliability
  - Performance
  - Supportability
  - Design Constraints

Supplementary Specification

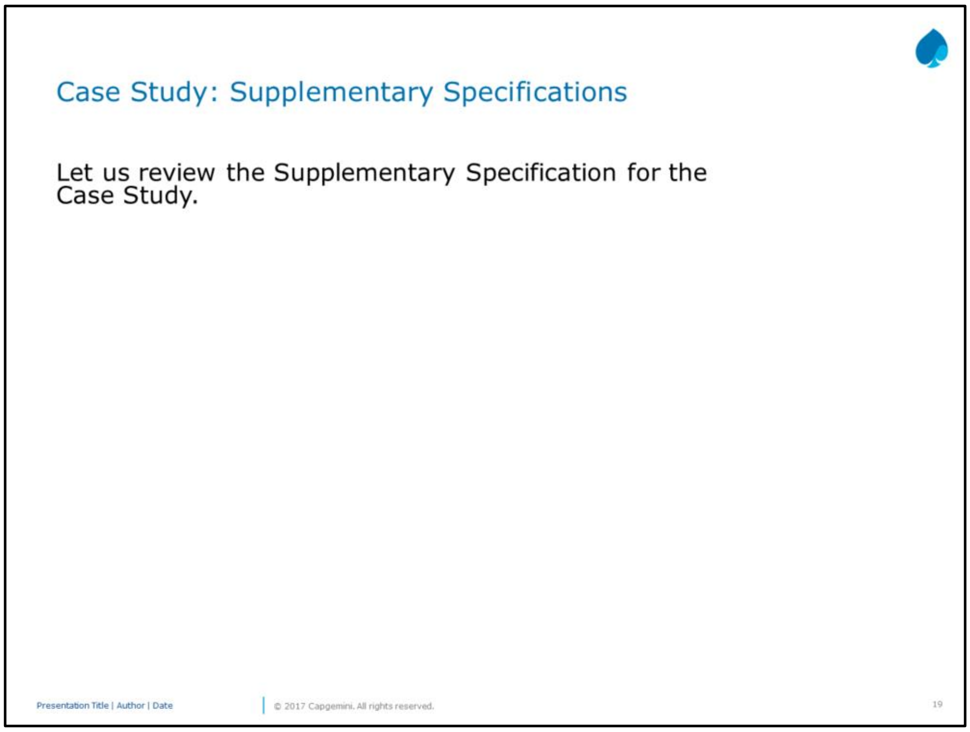
Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 18

<u>Concepts: Supplementary Specifications</u>:

- System requirements that are not described in use-cases are defined in the supplementary specifications. They complement and enhance information provided in the use-cases.
- Types of requirements captured in supplementary specifications include:
  - ➢ **Functionality:** They are functional requirements common to many use-cases. They are also those that cannot be captured in the Use Case Format.
  - ➢ **Usability:** They are requirements that relate to, or affect, the usability of the system. Examples include ease-of-use requirements or training requirements that specify how readily the system can be used by its actors.
  - ➢ **Reliability:** They are any requirements concerning the reliability of the system. Quantitative measures such as mean time between failure or defects per thousand lines of code should be stated.
  - ➢ **Performance:** They include the performance characteristics of the system. They include specific response times. Reference related use cases by name.
  - ➢ **Supportability:** They include any requirements that will enhance the supportability or maintainability of the system being built.
  - ➢ **Design Constraints:** They include any design constraints on the system being built.
- Supplementary Specifications go hand-in-hand with the Use-Case Model, implying that:
  - ➢ They are initially considered in the Inception phase as a complement to defining the scope of the system.
  - ➢ They are refined in an incremental fashion during the Elaboration and Construction phases.

Instructor Notes:

Walk through the non-functional requirements and the other functional requirements.



Case Study: Supplementary Specifications

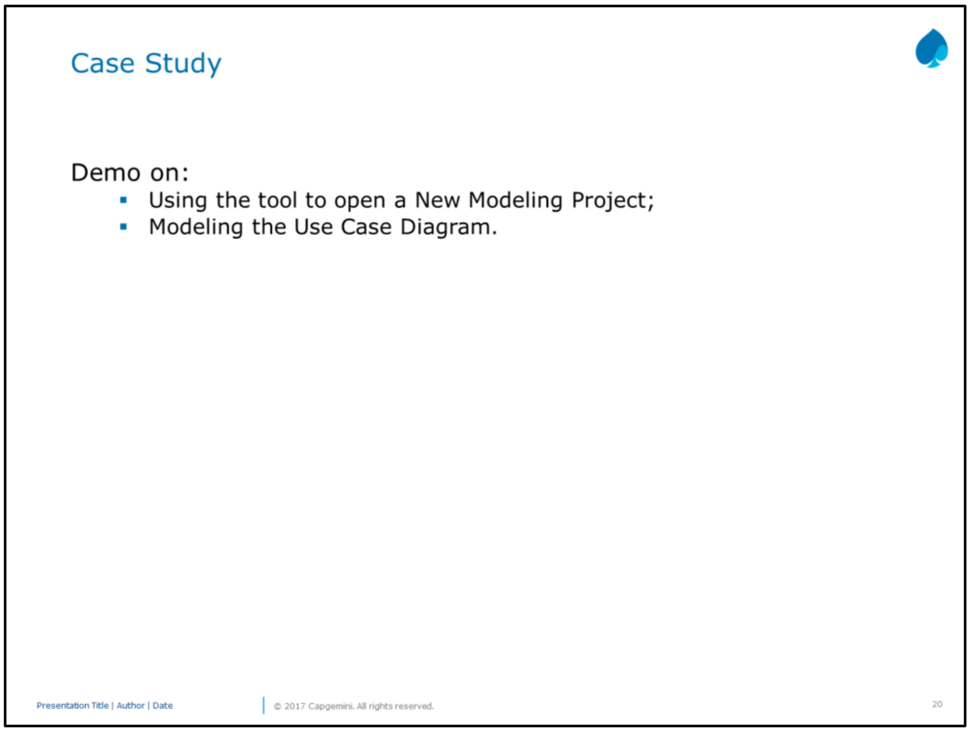
Let us review the Supplementary Specification for the Case Study.

Case Study: Supplementary Specifications:
- Let us now review the Supplementary Specification for the Case Study.
- Refer Appendix 3 for Supplementary Specification.

Instructor Notes:

We currently use Rational Software Architect for modeling different models and various design decisions.

Participants can follow the instructor and replicate the demo steps on their workstations.



Case Study

Demo on:

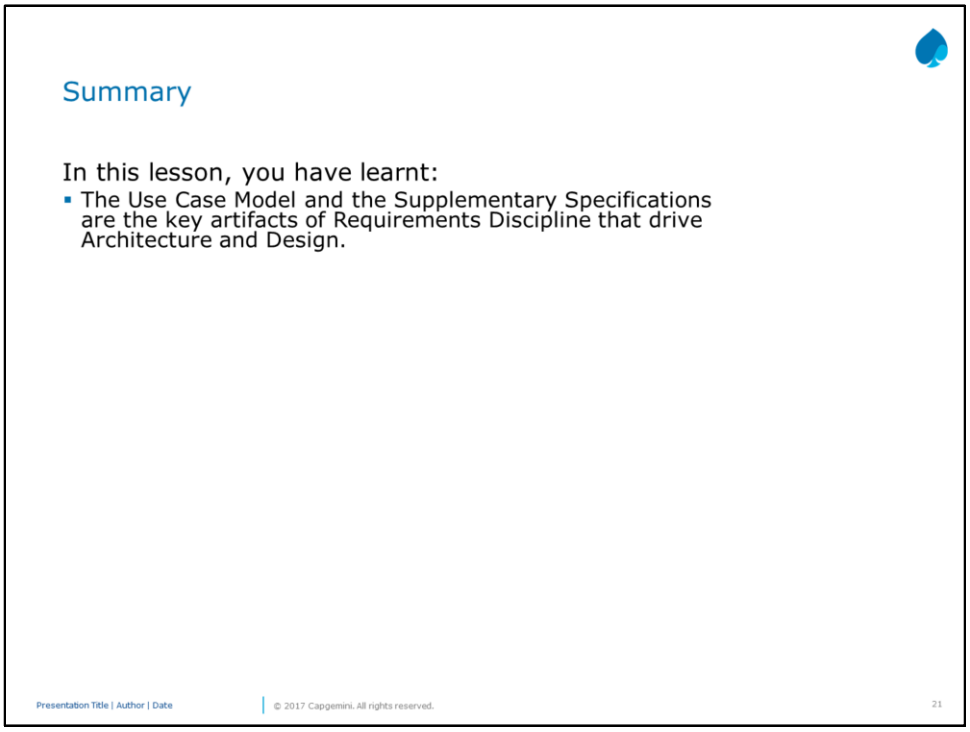
- Using the tool to open a New Modeling Project;
- Modeling the Use Case Diagram.

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

20

Instructor Notes:



Summary

In this lesson, you have learnt:

- The Use Case Model and the Supplementary Specifications are the key artifacts of Requirements Discipline that drive Architecture and Design.

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 21

Note:
With this overview of Requirements, we are all set to get into the activities of Design.

Instructor Notes:

Answers for Review Questions:

Answer 1: Use Case Model, Supplementary Specification

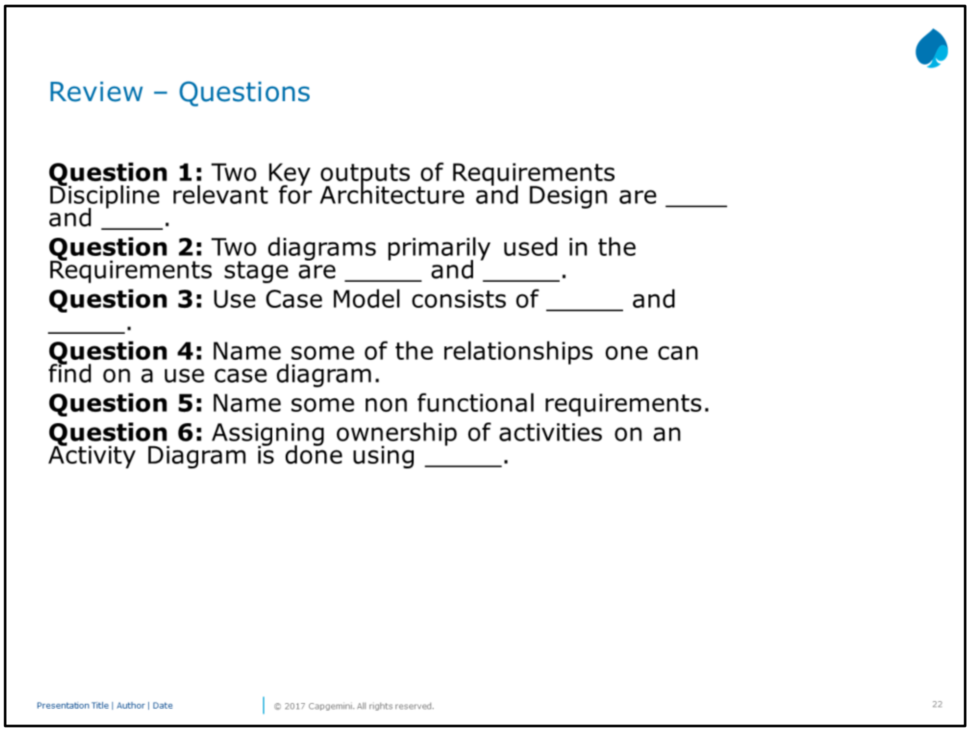
Answer 2: Use Case Diagram and Activity Diagram

Answer 3: Use Case Diagram and Use Case Description

Answer 4: Association, Generalization, include and extend

Answer 5: Usability, Reliability, Performance, Supportability, Security

Answer 6: Swim Lanes



Review – Questions

Question 1: Two Key outputs of Requirements Discipline relevant for Architecture and Design are _____ and _____.

Question 2: Two diagrams primarily used in the Requirements stage are ______ and ______.

Question 3: Use Case Model consists of ______ and ______.

Question 4: Name some of the relationships one can find on a use case diagram.

Question 5: Name some non functional requirements.

Question 6: Assigning ownership of activities on an Activity Diagram is done using ______.

Presentation Title | Author | Date | © 2017 Capgemini. All rights reserved. 22