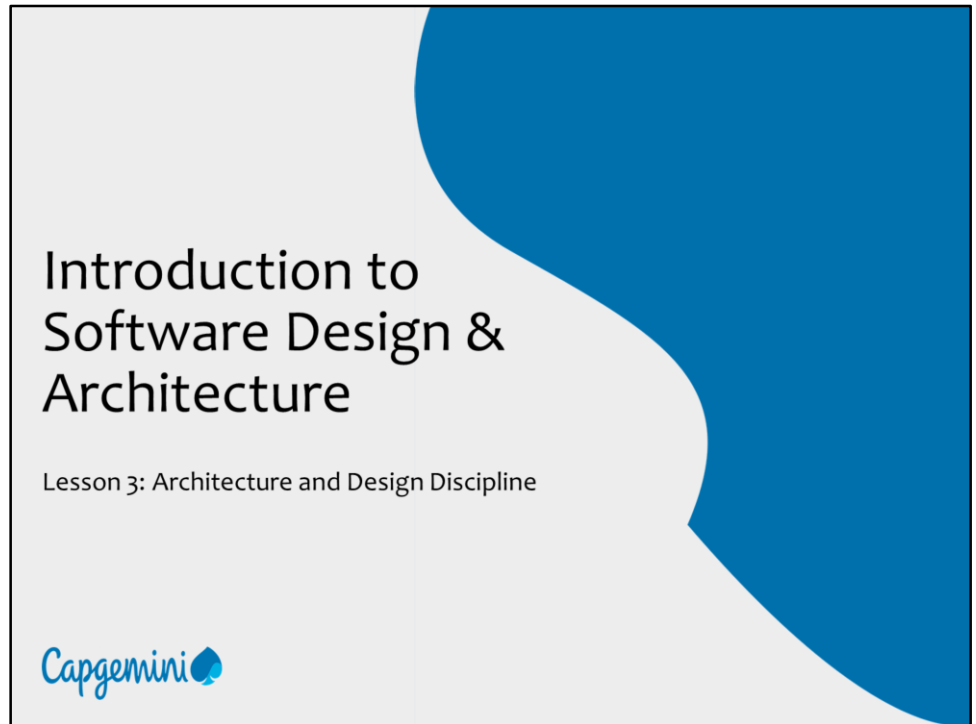


Instructor Notes:



## Instructor Notes:

This lesson is to provide a big picture for the Architecture and Design discipline. This sets the context for the subsequent lessons, where each of these activities will be looked at in detail.

### Lesson Objectives



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

- Understand Key Principles and Recommendations of the methodology for Architecture & Design
- Understand terms Architecture & Design and differences between them
- Understand Roles, Process Flow, Activities and Artifacts of Architecture & Design discipline

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

2

### Lesson Objectives:

- This lesson aims at providing an overview of the Architecture and Design discipline.
- Subsequent lessons will look into details of each activity of the Architecture and Design discipline.

## Instructor Notes:

Briefly explain these guiding principles. Each of these form the underlying guidelines for the activities of Architecture and Design.

### 3.1: Architecture and Design: Key Principles and Concepts

#### Key Principles

Principles used in the architecture and design process are as follows:

- Conduct Proof of Concept to validate architecture.
- Baseline Architecture by the end of elaboration phase.
- Visually Model using Unified Modeling Language (UML)

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

3

### Key Principles:

The guiding principles for Architecture and Design discipline are as follows:

1. Emphasis is given to creating one or more Proof of Concepts with an aim to validate the architectural solutions that are proposed.
2. The focus is on baselining the architecture by end of **Elaboration phase**. The methodology is based on IID, where initial iterations focus on minimizing the risks. Architecture is considered as one of the most risky items. Hence it is considered for initial iterations. **Architecture** has to be baselined by end of **Elaboration phase**.
3. The various decisions related to Architecture and Design are captured visually with the help of UML diagrams, to leverage the benefits that visual modeling provides in application development.

## Instructor Notes:

Architecture is said to be a mile wide and an inch deep. What that means is it will cover the breadth of the system in terms of coming up with the skeletal backbone. So this will include the top level modules, interfaces between them, interfaces if any with external systems and so on.

Focus is on non-functional requirements as well as critical functional requirements.

### Concepts: Software Architecture

**Architecture** is high level "organizing structure" of the system.  
Guiding activities:

- Solution space for "**non-functional requirements**"
- Decision on "**technology stack**"
- "**Framework**" requirements definition and solution
- "**Critical decisions**" for some risky "functional" requirements

Primary Performer: Technical Architect

Secondary Performer: Designer



Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

4


### Concepts: Software Architecture:

- **Architecture** is the high level organization structure for the system. It establishes a skeletal framework for the system. It includes all critical decisions, such as:
  - Choice of technology stack for the application
  - Choice of framework
  - Choice of architectural style, and so on
- Architecture focuses on the non-functional requirements, such as:
  - Usability
  - Reliability
  - Performance
  - Security, and so on
- It also looks into the critical functional requirements.
- The **Technical Architect** is the role responsible for these tasks. **Designer** supports the Technical Architect in these activities.

## Instructor Notes:

A complex definition!  
Helps to break it into individual units and explain.

Note that there are multiple perceptions about what constitutes software architecture.



### Concepts: Software Architecture (Contd.)

*"The **architecture** is the fundamental organization of a **system** embodied in its **components**, their **relationships** to each other, and to the **environment**, and the principles guiding its design and evolution."* [IEEE 1471]

Presentation Title | Author | Date
© 2017 Capgemini. All rights reserved.
5

### **Concepts: Software Architecture:**

- **Software architecture** has been defined in many different ways. Some websites provide a list of these definitions. One of these definitions is shown on the above slide.
- The key terms have been highlighted. Their definitions are given below:
  - **System:** A system can be any of the following:
    - individual applications
    - computer systems
    - subsystems
    - system of systems

A system can be defined as a collection of components arranged in a manner to fulfill a specific activity (function) or set of activities.
  - **Environment:** It refers to the context or circumstances that influences the system. Environment includes the developmental, operational, and political situations playing on the system.
  - **Component:** A component is a modular part of a system. It defines its behavior in terms of provided and required interfaces. It is one of the principle architectural building blocks of a system. It can be also defined as an implementation unit of software.
  - **Relationship:** It defines how elements interact to accomplish the work of a system.

## Instructor Notes:

Just like there are styles in buildings (For eg., Victorian Architecture or Moghul Architecture), there are styles for architecture too. These are captured as Architectural patterns such as MVC or Layers.

### Concepts: Software Architecture (Contd.)

Architecture includes all significant decisions, relating to the software system, such as:

- Identifying key, structural elements and their interfaces
- Defining system behavior as specified in collaborations among those elements
- Composition of these structural and behavioral elements into larger sub-systems
- Defining architectural style

Architectural decisions have to be well thought out because changing them has significant effects.

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

6

### Concepts: Software Architecture:

- Architecture focuses on the structural aspects of building the system. That includes:
  - Identifying the structural building blocks
  - Defining how these modules will interact with each other
- Architectural decisions are important decisions because the subsequent **design** will be based on the underlying **architecture**. Further, the subsequent **coding** will be based on these **designs**.

## Instructor Notes:

If architecture gives the skeleton, design adds flesh to that skeleton in terms of being a solution space for the functional requirements.

Our focus will be on creation of application design using UML since this is a OOAD course. Design patterns will be discussed in a separate course as part of designer certification.

Data Models is not in scope since it is done in consultation with database designer. We have a separate course that looks at database designing.

### Concepts: Design

Design focuses on modeling the “functional” aspects of an application.

- Also includes designing for UI and Non Functional Requirements in line with the architectural decisions

Guiding activities:

- Solution space for “functional requirements” based on defined architecture
- Choice of Design Patterns
- Creation of Application Design ( UML Models)
- Creation of Data Models (ER Models)

Primary Performer: Designer

Secondary Performers: Designers, Technical Architect



Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

7

### Concepts: Design:

- While Architecture focuses on non-functional aspects, **design** focuses on the **functional aspects** of the application. Design activities and decisions related to application design are taken in this phase. Design leads towards implementation. Hence focus is on creating detailed UML models and ER models to capture the designs.
- **Designer** is the primary owner of the design activities, with support from the **Technical Architect** for reviewing and the other **Designers in team** for detailing the designs.

## Instructor Notes:


Some of the outputs of Requirements Discipline which we discussed in earlier section become an input for the Architecture and Design discipline.

### 3.2: Introducing the Architecture and Design Discipline


#### Architecture and Design: Input Artifacts

Artifacts produced by the Requirements discipline, which serve as inputs to Architecture & Design:


- Use-Case Model (Use Case Diagram and Use Case Specifications)
- Supplementary Specifications for Non Functional Requirements



Use-Case Diagram



UC Specification



Supplementary Specification

Presentation Title | Author | Date
© 2017 Capgemini. All rights reserved.
8

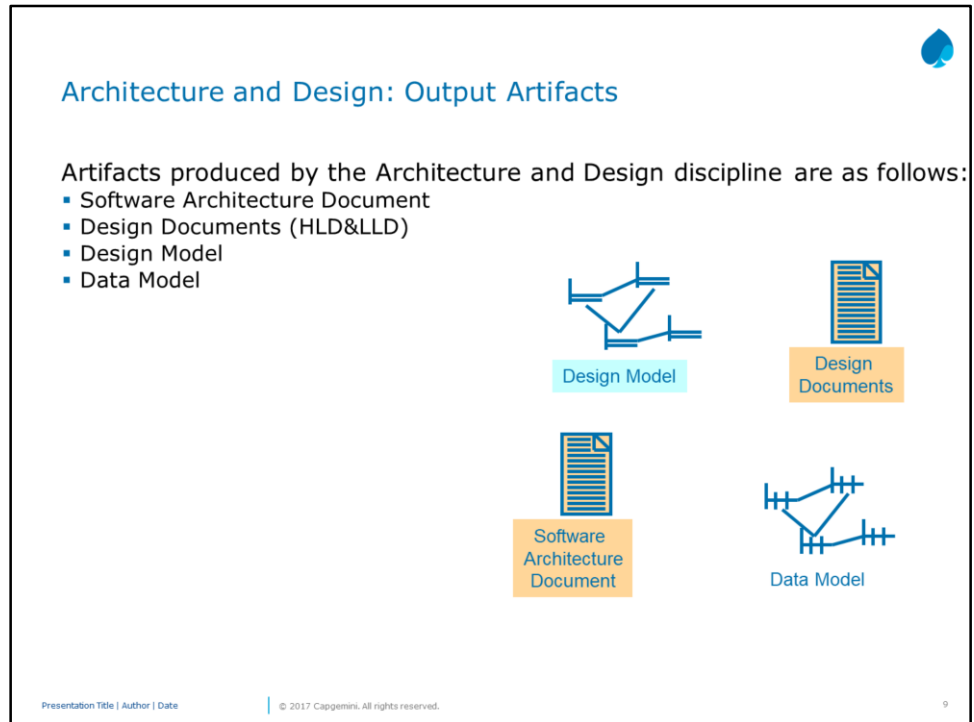
### Architecture and Design: Input Artifacts:

- **Requirements discipline** identifies and understands the requirements of all potential users. This information is gathered and collated to develop:
  - Use-Case models to capture the functional requirements
  - Supplementary specifications to capture the non-functional requirements
- These artifacts, produced by the **Requirements discipline** are the key inputs for the Architecture and Design discipline.



## Instructor Notes:

These artifacts would get created at different points in the Architecture and design process. There would be other intermediate artifacts too like the Analysis Model which evolves into the design model. There would also be certain other artifacts like test design strategy which we do not cover.

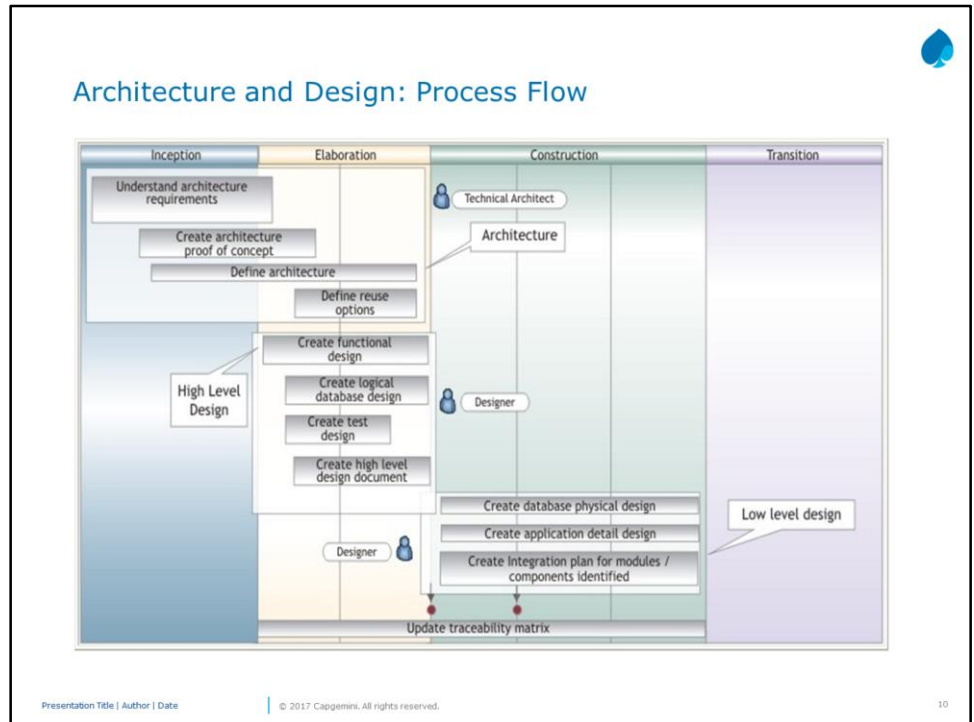


### Architecture and Design: Output Artifacts:

- Here are some of the artifacts that are produced by the Architecture and Design discipline:
  - **Software Architecture Document:** Early iterations in this discipline concentrate on the development and validation of the architecture. Software architecture is a collection of architectural views. Architectural views incorporate and reflect major design decisions. Each architectural view is an abstraction or simplification of the entire design, from a particular perspective, in which main features are highlighted, leaving out details. All **design decisions** and **architectural views** are collectively presented in the software architecture document.
  - **Design Document:** This presents a consolidated view of the design decisions, details of which will be available in the detailed design model.
  - **Design Model:** The design model consists of **design classes** that are grouped into **design packages**. Objects defined by these design classes collaborate to provide the required features in the system.
  - **Data Model:** The data model describes the logical and physical representations of persistent data used by the application. In cases where the application utilizes a relational database management system (RDBMS), the data model may also include model elements for stored procedures, triggers, constraints, etc., that define the interaction of the application components with the RDBMS.

## Instructor Notes:

Set the context for the Architecture and Design Discipline with this process flow.



### **Architecture and Design: Process Flow:**

- The above slide shows the Qzen Development Methodology process flow for the Architecture and Design discipline.
- It gives an overview of the key roles and activities that come into play for this discipline.

## Instructor Notes:

For convenience, the activities are put together into 3 logical groups. Focus on when these activities get done.

### Architecture and Design: Key Activities

#### Architecture:

- Aimed at defining the architecture for the system
- Performed in the Inception and Early Elaboration phases

#### High Level Design:

- Aimed at creating function design and logical database design
- Performed in the Elaboration phase

#### Low Level Design:

- Aimed at creating detailed design and physical database design
- Performed in each iteration of Construction phase

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

11

### Architecture and Design: Key Activities:

- The activities of Architecture and Design can be grouped into three top level groups:
  1. Activities relating to Architecture, where focus is on arriving at a solution for non-functional requirements to come up with a definition of Architecture of the system: These tasks get done in the **Inception** and early **Elaboration** phases. Architecture needs to get baselined by end of Elaboration phase.
  2. Activities relating to High Level Design, where focus is on creating a functional design for the application: This gets done in the **Elaboration** phase. This is done once for the entire system (using representative use cases).
  3. Activities relating to Low Level Design, where focus is coming up with detailed design of the application: There would be certain number of use cases taken up in each iteration and the Low Level Design will be done for those use cases only in the corresponding iteration of the **Construction** phase.

## Instructor Notes:

At this point, we can simply say these are the tasks. What happens in each task will get discussed later.

### Architecture and Design: Tasks

#### Architecture:

- AD01: Understand Architecture Requirements
- AD02: Create Architecture Proof Of Concept
- AD03: Define Architecture
- AD04: Define Reuse Options

#### High Level Design:

- AD05: Create Functional Design ( Analysis Model)
- AD06: Create Logical Database Design
- AD07: Create Test Design
- AD08: Create High Level Design Document

#### Low Level Design (per iteration):

- AD09: Create Database Physical Design
- AD10: Create Application Detail Design ( Design Model)
- AD11: Create Integration Plan for Modules/ Components Identified

Presentation Title | Author | Date


© 2017 Capgemini. All rights reserved.

12

**Note:** The above slide shows a list of tasks that get done. Each of these will get discussed in later sections.

## Instructor Notes:

During the course, we may not completely create each of these artifacts but will be looking at many of the inputs which are needed to put together these documents.



**Architecture and Design: Artifacts**

- Architecture
  - Software Architecture Document\*
- High Level Design
  - Data Model\*
  - Analysis Model
  - High Level Design Document\*
- Low Level Design
  - Design Model
  - Low Level Design Document\*

\* templates available

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

### Architecture and Design: Artifacts:

- The above slide shows various artifacts that are related to Architecture and Design discipline. Qzen provides templates for many of these artifacts.
- **Analysis Model** and **Design Model** are UML models. The subsequent slides provide more details about them.

## Instructor Notes:

This is to give a big picture of the UML models that will get created. Since the focus is on OOAD, we will be spending a lot of time discussing the UML Models.

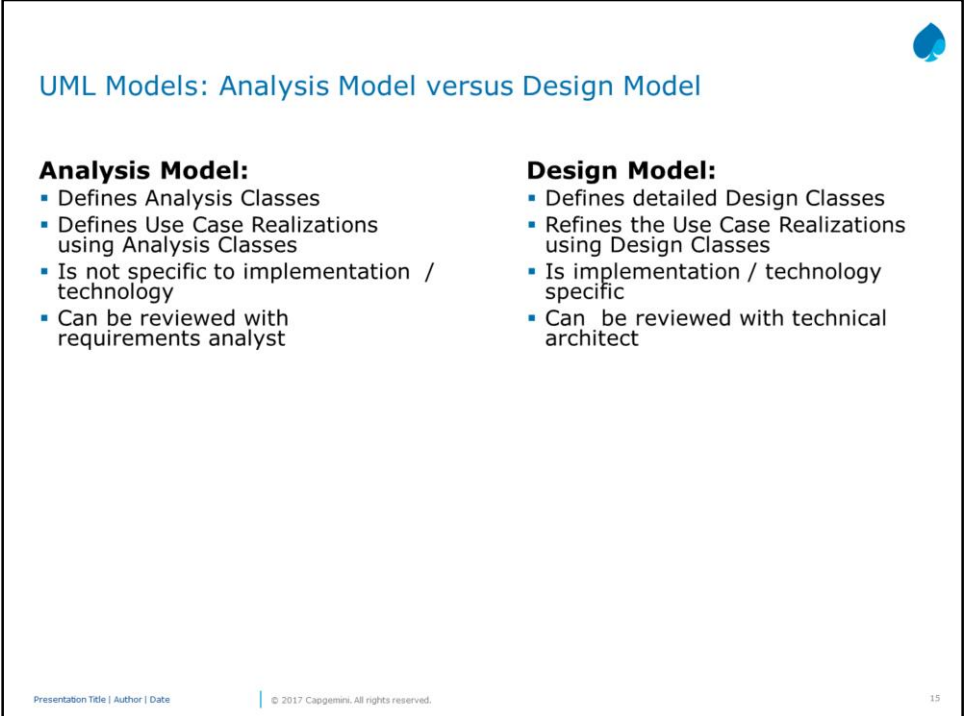
Architecture and Design: UML Models	
<b>Use Case Model</b>	<ul style="list-style-type: none"> <li>It is created as part of the Requirements Discipline. The Use Case Model consists of Use Case Diagram, complemented by the Use Case Descriptions.</li> <li>It captures the Functional Requirements of the System.</li> </ul>
<b>Analysis Model</b>	<ul style="list-style-type: none"> <li>It is created as part of High Level Design activities. The Analysis Model includes Analysis Classes and Use Case Realizations.</li> <li>It captures the conceptual, technology independent view of the system.</li> </ul>
<b>Design Model</b>	<ul style="list-style-type: none"> <li>It is created as part of Low Level Design activities. The Design Model is a refinement of the Analysis Model to include detailed application designs.</li> <li>It captures technology specific detailed design for the application.</li> </ul>

### Architecture and Design: UML Models:

- The above slide shows the different UML models which we shall encounter.
- We have already seen the **Use-Case Model** while discussing the Requirements Discipline. This is used to come up with a technology agnostic **Analysis Model** of the system during the High Level Design activities. The Analysis Model is then refined to obtain the detailed design of the application during the Low Level Design activities.

## Instructor Notes:

Analysis Model is just an intermediate step to arrive at the design model. A top level understanding of the differences will help prior to actual creation of these models.



The slide is titled "UML Models: Analysis Model versus Design Model" in blue text. It features a blue spade icon in the top right corner. The content is organized into two columns. The left column is headed "Analysis Model:" and lists four bullet points: "Defines Analysis Classes", "Defines Use Case Realizations using Analysis Classes", "Is not specific to implementation / technology", and "Can be reviewed with requirements analyst". The right column is headed "Design Model:" and lists four bullet points: "Defines detailed Design Classes", "Refines the Use Case Realizations using Design Classes", "Is implementation / technology specific", and "Can be reviewed with technical architect". At the bottom, there is a footer with "Presentation Title | Author | Date" on the left, "© 2017 Capgemini. All rights reserved." in the center, and the number "15" on the right.

<b>Analysis Model:</b>	<b>Design Model:</b>
▪ Defines Analysis Classes	▪ Defines detailed Design Classes
▪ Defines Use Case Realizations using Analysis Classes	▪ Refines the Use Case Realizations using Design Classes
▪ Is not specific to implementation / technology	▪ Is implementation / technology specific
▪ Can be reviewed with requirements analyst	▪ Can be reviewed with technical architect

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

### UML Models: Analysis Model versus Design Model:

- The above slide shows how the Analysis Model and Design Model compare with each other.
  - The Analysis Model is conceptual with no details such as datatypes included.
  - The Design Model details out the Analysis Model and includes technology specific details. The Design Model is at a detailed stage where it can be taken up for implementation.

## Instructor Notes:

Here are some key characteristics of a good design.

### Characteristics of a Good Design



A good software design:

- Is dynamic and resilient.
- Is capable of adapting to frequent change requirements during:
  - Development phase
  - Maintenance phase
- Changes minimally to accommodate extension of requirements.
- Changes minimally to accommodate radical changes in the input and output methods of the program.
- Has no redundancy.

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

16

### Characteristics of a Good Design:

- Today, we live in a world that is highly dynamic and diverse in nature. As a result, our requirements too are constantly changing. Therefore it is not surprising that “dynamic” and “resilient” software systems are the need of the day.
- The challenge is in developing a software system capable of adapting to the ever changing requirements, complexities of the problem domain, and difficulty of managing software processes.
- A **good design** and **code** adapts easily to the frequent changes that are done during development and maintenance. Very often extensive changes are involved, when a new functionality is added to an application. The design should be able to incorporate the added functionality with minimum change to the existing codes.
- Existing codes are already tested units, and changing them may result in unwanted changes in related functionalities. A software designer should make his or her design **foolproof** against such eventualities.



## Instructor Notes:

If the design satisfies the requirements but exhibits characteristics shown here, it could mean that design has scope for improvement! Note that many of them relate to high interdependence between classes/modules.

These have been stated by Robert Martin (now known as Uncle Bob), one of the key figures in the field of Object Orientation.

### Symptoms of a Rotting Design

Here are some obvious symptoms of a rotting design:

- **Rigidity:**
  - Code does not adapt to changes.
  - Managers refuse to allow changes in the software.
- **Fragility:**
  - The smallest of changes results in a cascading effect.
  - Code breaks in many places, with every change.
- **Immobility:**
  - The design poses inability to reuse software from other projects.
- **Viscosity:**
  - It is easier to hack than retain the original design.
  - The development environment is slow and inefficient.

Presentation Title | Author | Date

© 2017 Capgemini. All rights reserved.

17

### Symptoms of a Rotting Design:

It does not take much for a software to rot during its journey from design to implementation. A rotting design is a mass of codes that the designers find increasingly difficult to handle. Finally, software engineers and designers call for a redesign project. However, you need to remember that such redesigns are rarely successful. Therefore you need to look out for the symptoms, and prevent them from growing out of control.

There are four primary symptoms of a rotting design:

**Rigidity:** It is the tendency of a software to change even in the smallest of ways. Every change results in a cascading effect, bringing about subsequent changes in related modules. When software exhibits such characteristics, the managers avoid fixing even the simplest of problems.

**Fragility:** It is the tendency of the software to break in many places every time it changes. Often the break occurs at a point remotely connected with the point of change. In fact, the two points may not be related at all. Due to the break, the fragility increases, and soon the situation goes beyond control.

**Immobility:** It is the inability to reuse software from other projects or from other parts of the same project. This situation occurs when a module has too much related software that it depends on.

**Viscosity:** Viscosity is of two types: viscosity of design and viscosity of environment. When design preserving methods are more difficult to implement than the hacks, then we can say that the viscosity of design is very high. When the design environment is slow and inefficient we can say that the viscosity of environment is high.

## Instructor Notes:

### Lesson Summary



In this lesson, you have learnt:

- The Architecture and Design discipline uses the Requirements Artifacts, Use Case Model, and Supplementary Specifications, to produce the following:
  - Software Architecture Document
  - Design Documents (High Level & Low Level)
  - Design Model
  - Data Model
- Key roles involved are that of Technical Architect and Designer

## Instructor Notes:

Answers for Review Questions:

Answer 1: Elaboration

Answer 2: Architecture

Answer 3: Design

Answer 4: Analysis Model and Design Model

Answer 5: Technical Architect, Designer

### Review – Questions

**Question 1:** Architecture needs to be baselined by end of \_\_\_\_\_ phase.

**Question 2:** Decision on Technology Stack is part of activities related to \_\_\_\_\_.

**Question 3:** \_\_\_\_\_ focuses on the functional aspects of the application design.

**Question 4:** Name the key UML models created during the Architecture and Design activities.

**Question 5:** Name the key roles for A&D discipline.