


► ► ► **Module 4**


Analysis and Design Overview




IBM Software Group

Mastering Object-Oriented Analysis and Design
with UML 2.0

Module 4: Analysis and Design Overview





Topics

Analysis Versus Design.....	4-6
Analysis and Design Workflow.....	4-12
Review.....	4-19

Objectives: Analysis and Design Overview

Objectives: Analysis and Design Overview

- ♦ Review the key Analysis and Design terms and concepts
- ♦ Introduce the Analysis and Design process, including roles, artifacts and workflow
- ♦ Explain the difference between Analysis and Design

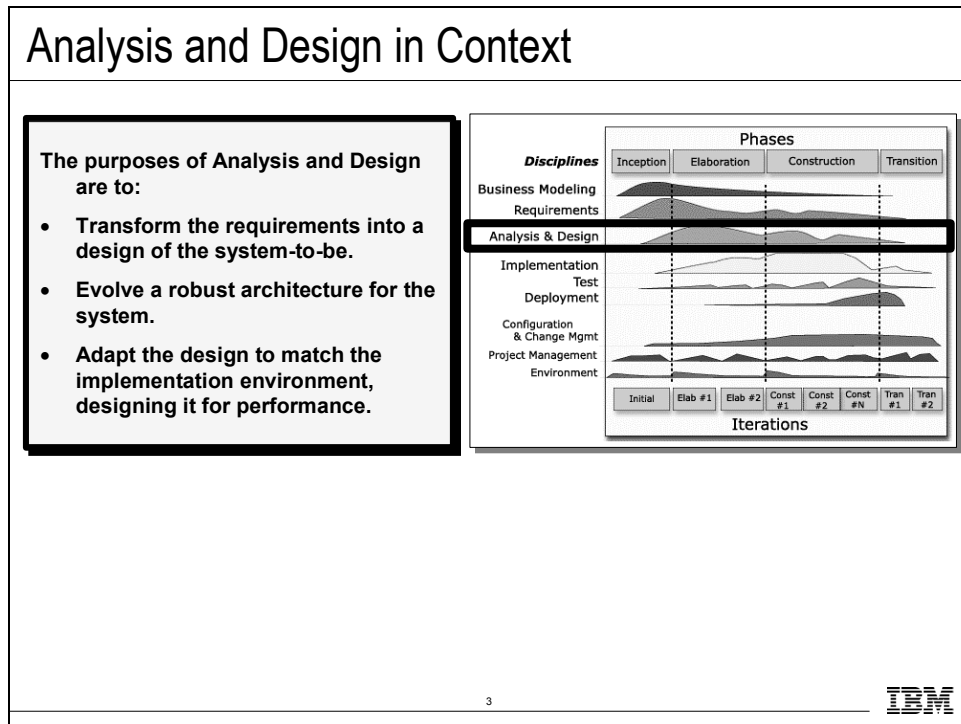
2



This Analysis and Design Overview module provides an overview of the Analysis and Design Discipline. It also defines the terms that span all Analysis and Design activities.

The details of each of the Analysis and Design activities will be described in subsequent modules. This module is meant to provide context for these detailed Analysis and Design activity modules.

Analysis and Design in Context



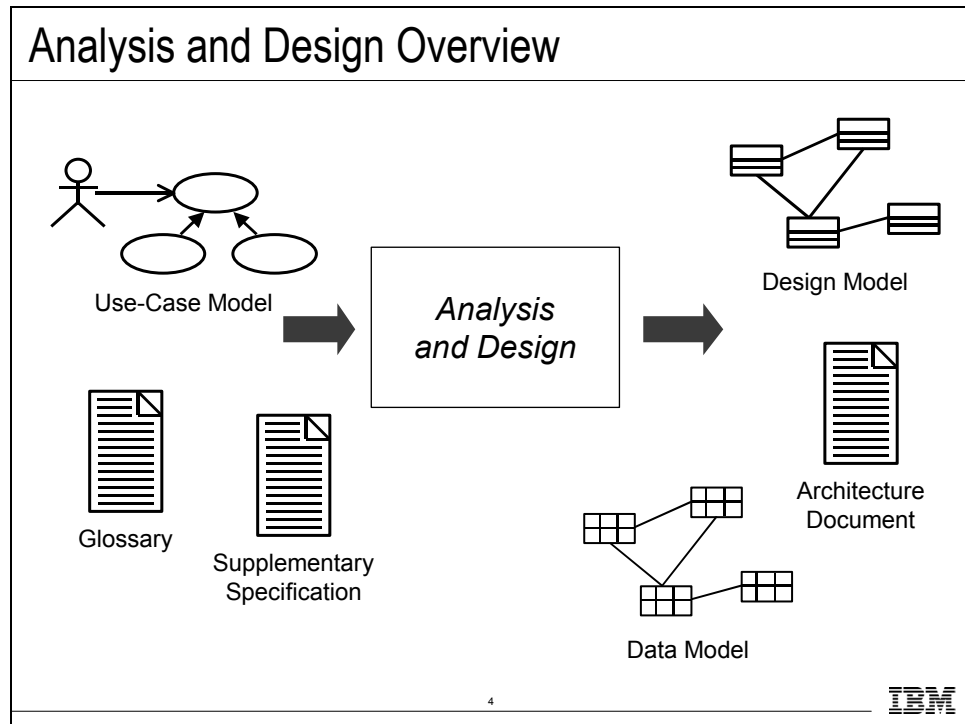
The purposes of Analysis and Design are to:

- Transform the requirements into a system design.
- Evolve a robust architecture for the system.
- Adapt the design to match the implementation environment, designing it for performance.

The Analysis and Design Discipline is related to other process disciplines.

- The Business Modeling Discipline provides an organizational context for the system.
- The Requirements Discipline provides the primary input for Analysis and Design.
- The Test Discipline tests the system designed during Analysis and Design.
- The Environment Discipline develops and maintains the supporting artifacts that are used during Analysis and Design.
- The Management Discipline plans the project and each iteration (described in an Iteration Plan).

Analysis and Design Overview



The input artifacts are the Use-Case Model, Glossary, and Supplementary Specification from the Requirements Discipline. The result of Analysis and Design is a Design Model that serves as an abstraction of the source code; that is, the Design Model acts as a blueprint of how the source code is structured and written. The Design Model consists of design classes structured into design packages; it also contains descriptions of how objects of these design classes collaborate to perform use cases (use-case realizations).

The design activities are centered around the notion of architecture. The production and validation of this architecture is the main focus of early design iterations. Architecture is represented by a number of architectural views that capture the major structural design decisions. In essence, architectural views are abstractions or simplifications of the entire design, in which important characteristics are made more visible by leaving details aside. The architecture is an important vehicle not only for developing a good Design Model, but also for increasing the quality of any model built during system development. The architecture is documented in the Architecture Document.

The development of the Architecture Document is out of the scope of this course, but we will discuss its contents and how to interpret them.

Analysis & Design Overview Topics

Analysis & Design Overview Topics

- ☆ ♦ Key Concepts
 - ♦ Analysis and Design Workflow

5



We will start out by defining some key terms and concepts needed to describe the Analysis and Design workflow. These terms will be explained in more detail, along with other important terms and concepts in later modules of the course.

Once we have a common vocabulary, then we will walk through the Analysis and Design workflow.

Analysis Versus Design

Analysis Versus Design	
Analysis	Design
<ul style="list-style-type: none"> ▪ Focus on understanding the problem ▪ Idealized design ▪ Behavior ▪ System structure ▪ Functional requirements ▪ A small model 	<ul style="list-style-type: none"> ▪ Focus on understanding the solution ▪ Operations and attributes ▪ Performance ▪ Close to real code ▪ Object lifecycles ▪ Nonfunctional requirements ▪ A large model

6



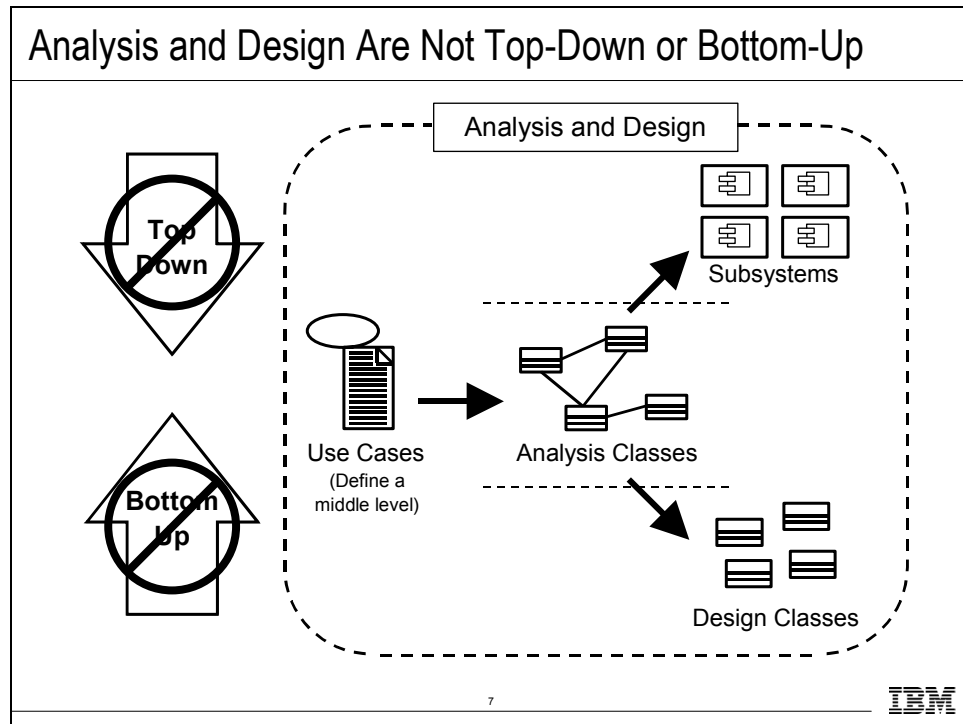
The differences between Analysis and Design are ones of focus and emphasis. The above slide lists the things that you focus on in Analysis versus Design.

The goal in Analysis is to understand the problem and to begin to develop a visual model of what you are trying to build, independent of implementation and technology concerns. Analysis focuses on translating the functional requirements into software concepts. The idea is to get a rough cut at the objects that comprise our system, but focusing on behavior (and therefore encapsulation). We then move very quickly, nearly seamlessly, into "Design" and the other concerns.

A goal of Design is to refine the model with the intention of developing a Design Model that will allow a seamless transition to the coding phase. In design, we adapt to the implementation and the deployment environment. The implementation environment is the "developer" environment, which is a software superset and a hardware subset of the deployment environment.

In modeling, we start with a model that closely resembles the real world (Analysis), and then find more abstract (but more fundamental) solutions to a more generalized problem (Design). The real power of software design is that it can create more powerful metaphors for the real world that change the nature of the problem, making it easier to solve.

Analysis and Design Are Not Top-Down or Bottom-Up



The Analysis and Design Discipline is not top-down or bottom-up.

The use case comes in from the left and defines a middle level.

The analysis classes are not defined in a top-down pattern or a bottom-up pattern; they are in the middle. From this middle level one may move up or down.

Defining subsystems is moving up and defining design classes is moving down.

Analysis is both top-to-middle, middle-up, middle-down and bottom-to-middle. There is no way of saying that one path is more important than another — you have to travel on all paths to get the system right.

All of these four paths are equally important. That is why the bottom-up and top-down question cannot be solved.

What Is Architecture?

What Is Architecture?

- ♦ Software architecture encompasses a set of significant decisions about the organization of a software system.
 - Selection of the structural elements and their interfaces by which a system is composed
 - Behavior as specified in collaborations among those elements
 - Composition of these structural and behavioral elements into larger subsystems
 - Architectural style that guides this organization

*Grady Booch, Philippe Kruchten, Rich Reitman, Kurt Bittner; Rational
(derived from Mary Shaw)*

8



Based on extensive research, Rational has established a definition of architecture.

“Significant” in this context implies strategic, of major impact.

The architecture has a static and a dynamic perspective.

The architecture for similar systems should be similar (a particular style is used).

An equation we have used is:

$$\text{Architecture} = \text{Elements} + \text{Form} + \text{Rationale}.$$

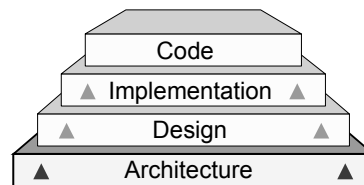
Rationale is essential for justifying a good architecture.

Patterns are the guidelines for assembling elements in some form. We will discuss patterns in the architecture modules.

Architecture Constrains Design and Implementation

Architecture Constrains Design and Implementation

- ♦ Architecture involves a set of strategic design decisions, rules or patterns that constrain design and construction.



Architecture decisions are the most fundamental decisions, and changing them will have significant effects.

9

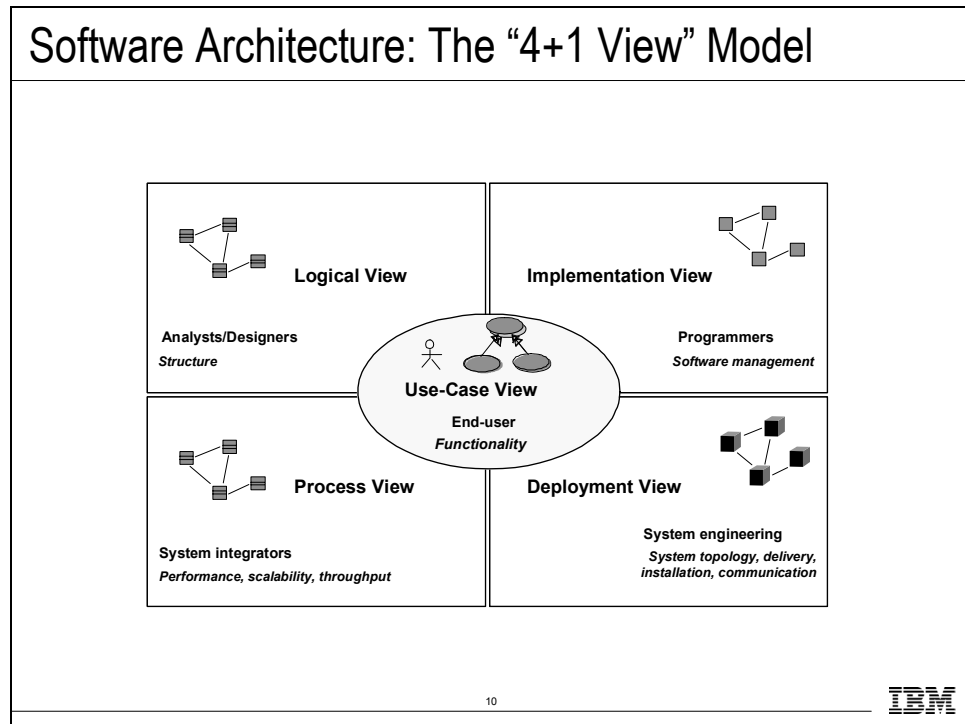


Architectures can be viewed as a set of key design decisions.

The architecture is the initial set of constraints placed on the system. Such constraints are the the most important ones. They constitute the fundamental decisions about the software design. Architecture puts a framework around the design. Architecture has been called strategic design.

An architect's job is to eliminate unnecessary creativity as the design has to fit into the architectural framework. As you move closer to code, creativity is focused. (The architecture frames the design which frames the implementation.) This is good because during Implementation, the creativity can be spent elsewhere (for example, for improving the quality, and performance) of the implementation (for example, code).

Software Architecture: The “4+1 View” Model



The above diagram shows the model Rational uses to describe the software architecture.

Architecture is many things to many different interested parties. On a particular project, there are usually multiple stakeholders, each with their own concerns and view of the system to be developed. The goal is to provide each of these stakeholders with a view of the system that addresses their concerns, and suppresses the other details.

To address these different needs, Rational has defined the “4+1 view” architecture model. An architectural view is a simplified description (an abstraction) of a system from a particular perspective or vantage point, covering particular concerns, and omitting entities that are not relevant to this perspective. Views are “slices” of models.

Not all systems require all views (for example, single processor: drop Deployment View; single process: drop Process View; small program: drop Implementation View, and so forth). A project may document all of these views or additional views. The number of views is dependent on the system you are building.

Each of these views, and the UML notation used to represent them, will be discussed in subsequent modules.

Analysis & Design Overview Topics

Analysis & Design Overview Topics

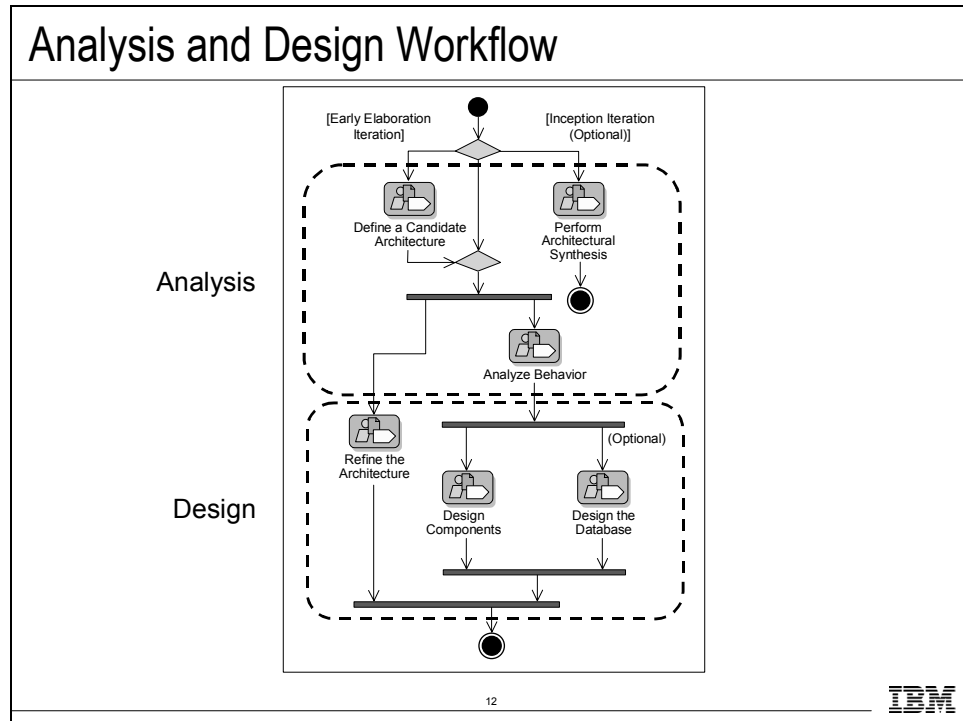
- ♦ Key Concepts
- ☆ ♦ Analysis and Design Workflow

11



Because we have a common vocabulary, we can now briefly discuss the activities of Analysis and Design and how they work together.

Analysis and Design Workflow



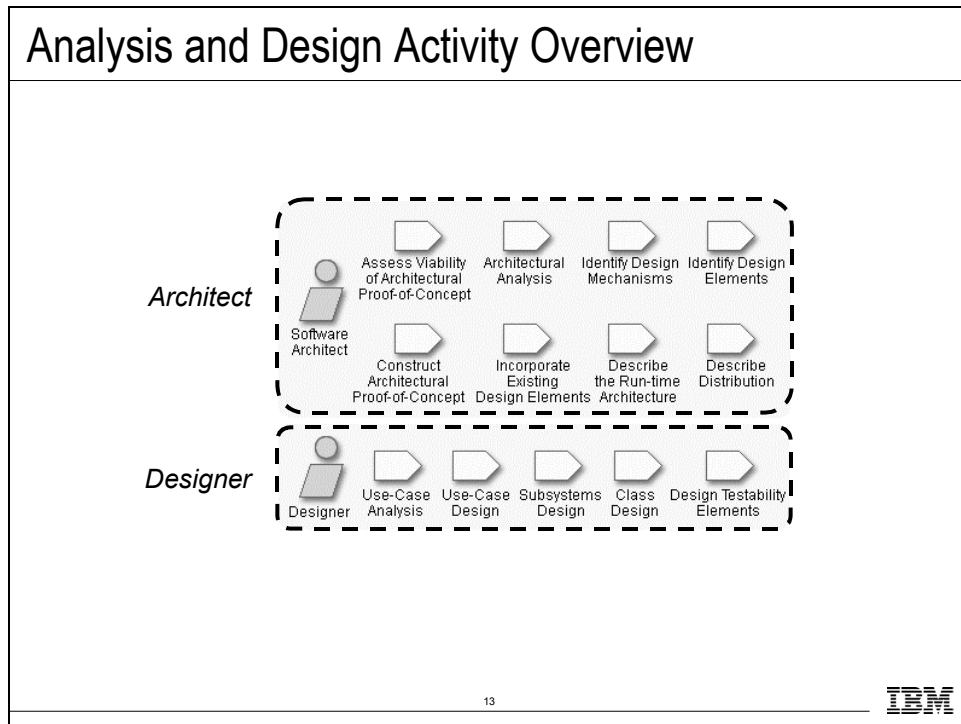
A mere enumeration of all workers, activities, and artifacts does not constitute a process. We need a way to describe the activities, some valuable result, and interactions between workers. A *workflow* is a sequence of activities that produces a result of observable value.

In UML terms, a workflow can be expressed as a sequence diagram, a collaboration diagram, or an activity diagram. We use a form of activity diagram in the Rational Unified Process. For each core workflow, an **activity** diagram is presented. This diagram shows the workflow, expressed in terms of workflow details.

This slide shows the Analysis and Design workflow. The early Elaboration Phase focuses on creating an initial architecture for the system (**Define a Candidate Architecture**) to provide a starting point for the main analysis work. If the architecture already exists (because it was produced in previous iterations, or projects, or is obtained from an application framework), the focus of the work changes to refining the architecture (**Refine the Architecture**) analyzing behavior, and creating an initial set of elements that provide the appropriate behavior (**Analyze Behavior**).

After the initial elements are identified, they are further refined. **Design Components** produce a set of components that provide the appropriate behavior to satisfy the requirements on the system. In parallel with these activities, persistence issues are handled in **Design the Database**. The result is an initial set of components that are further refined in the **Implementation Discipline**.

Analysis and Design Activity Overview

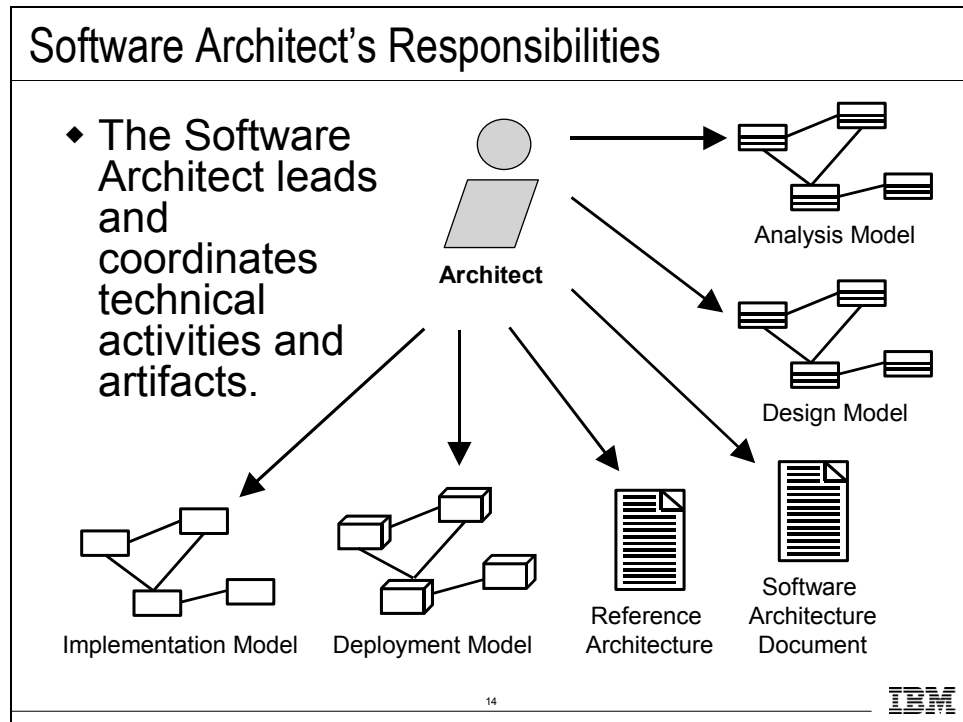


Remember, for Analysis and Design, we start out with the Use-Case Model and the supplementary specifications from the Requirements Discipline and end up with the Design Model that serves as an abstraction of the source code.

The design activities are centered around the notion of architecture. The production and validation of this architecture are the main focal points of early design iterations. The architecture is an important vehicle not only for developing a good Design Model, but also for increasing the quality of any model built during system development.

The focus of this course is on the activities of the designer. The architect activities are discussed, but many of the architectural decisions will be given. Many of the architect and designer activities will be addressed in individual course modules.

Software Architect's Responsibilities

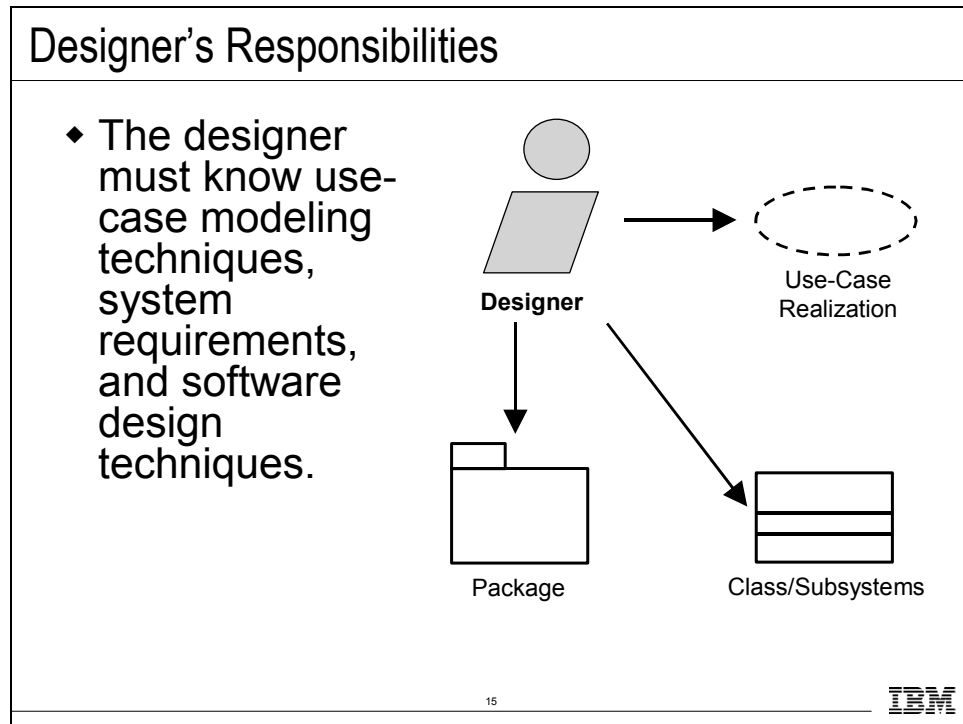


The software architect role leads and coordinates technical activities and artifacts throughout the project. The software architect establishes the overall structure for each architectural view: the decomposition of the view, the grouping of elements, and the interfaces between these major groupings. Therefore, in contrast to the other roles, the software architect's view is one of breadth as opposed to one of depth.

In summary, the software architect must be well-rounded and possess maturity, vision, and a depth of experience that allows for grasping issues quickly and making educated, critical judgment in the absence of complete information. More specifically, the software architect, or members of the architecture team, must combine these skills:

- Experience in both the problem domain, through a thorough understanding of the requirements, and the software engineering domain. If there is a team, these qualities can be spread among the team members, but at least one software architect must provide the global vision for the project.
- Leadership in order to drive the technical effort across the various teams, and to make critical decisions under pressure and make those decisions stick. To be effective, the software architect and the project manager must work closely together, with the software architect leading the technical issues and the project manager leading the administrative issues. The software architect must have the authority to make technical decisions.
- Communication to earn trust, to persuade, to motivate, and to mentor. The software architect cannot lead by decree — only by the consent of the rest of the project. In order to be effective, the software architect must earn the respect of the project team, the project manager, the customer, and the user community, as well as the management team.
- Goal orientation and being proactive with a relentless focus on results. The software architect is the technical driving force behind the project, not a visionary or dreamer. The career of a successful software architect is a long series of sub-optimal decisions made in uncertainty and under pressure. Only those who can focus on doing what needs to be done are successful in this environment of the project.

Designer's Responsibilities



The designer role defines the responsibilities, operations, attributes, and relationships of one or several classes, and determines how they are adjusted to the implementation environment. In addition, the designer role may have responsibility for one or more classes, including analysis, design, subsystems, or testability.

The designer must have a solid working knowledge of:

- Use-case modeling techniques
- System requirements
- Software design techniques, including object-oriented Analysis and Design techniques, and the Unified Modeling Language
- Technologies with which the system will be implemented

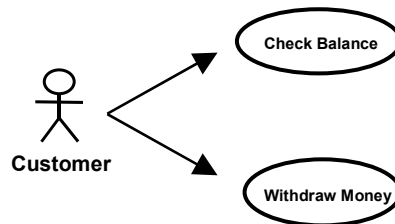
In addition, the designer must:

- Understand the architecture of the system, as represented in the Software Architecture Document.

Review: Analysis and Design Is Use-Case Driven

Review: Analysis and Design Is Use-Case Driven

- ♦ Use cases defined for a system are the basis for the entire development process.
- ♦ Benefits of use cases:
 - Concise, simple, and understandable by a wide range of stakeholders.
 - Help synchronize the content of different models.



16

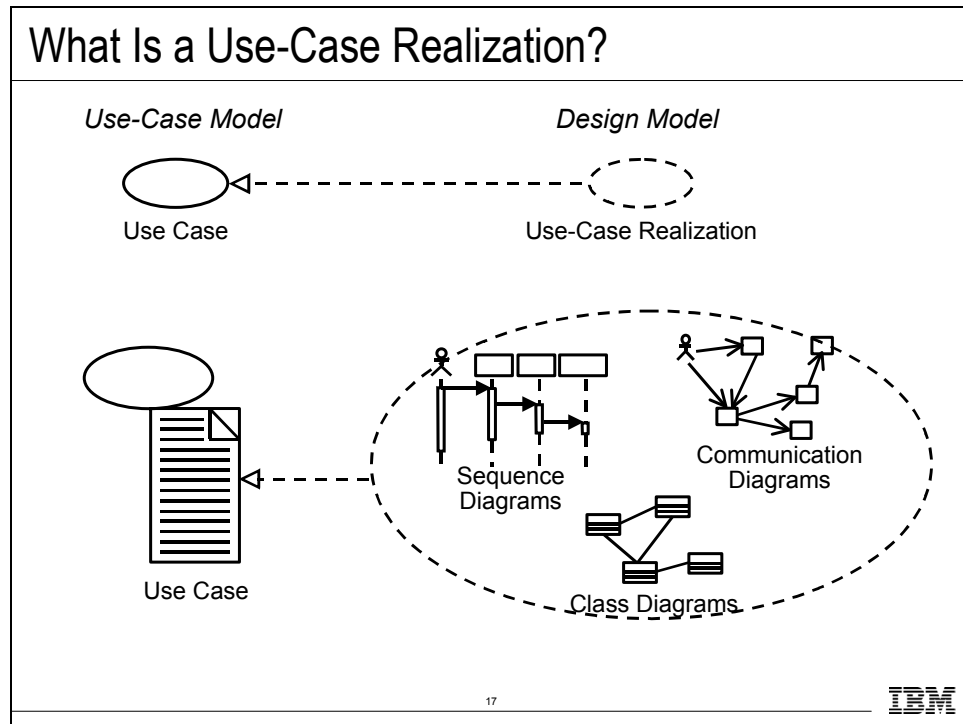
IBM

Use cases are one recommended method for organizing your requirements. Instead of a bulleted list of requirements, you organize them in a way that tells how someone may use the system. By doing so, you make a requirement more complete and consistent. You can also better understand the importance of a requirement from a user's perspective.

It is often difficult to tell how a system does what it is supposed to do from a traditional object-oriented system model. This stems from the lack of a common thread through the system when it performs certain tasks. Use cases are that thread, because they define the behavior performed by a system.

Use cases are not part of "traditional" object orientation, but their importance has become more and more apparent, further emphasizing the fact that use cases are part of the UML.

What Is a Use-Case Realization?



A use-case realization describes how a particular use case is realized within the Design Model, in terms of collaborating objects. A use-case realization ties together the use cases from the Use-Case Model with the classes and relationships of the Design Model. A use-case realization specifies what classes must be built to implement each use case.

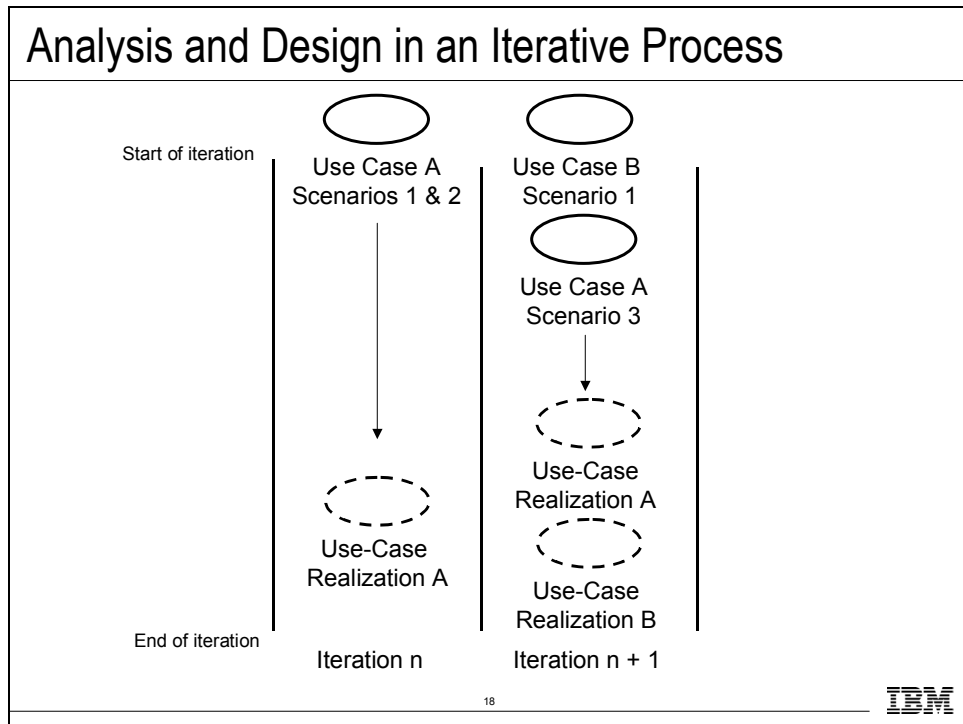
In the UML, use-case realizations are stereotyped collaborations. The symbol for a collaboration is an ellipsis containing the name of the collaboration. The symbol for a use-case realization is a dotted line version of the collaboration symbol.

A use-case realization in the Design Model can be traced to a use case in the Use-Case Model. A realization relationship is drawn from the use-case realization to the use case it realizes.

Within the UML, a use-case realization can be represented using a set of diagrams that model the context of the collaboration (the classes/objects that implement the use case and their relationships — class diagrams), and the interactions of the collaborations (how these classes/objects interact to perform the use cases — communication and sequence diagrams).

The number and types of the diagrams that are used depend on what is needed to provide a complete picture of the collaboration and the guidelines developed for the project under development.

Analysis and Design in an Iterative Process



This course assumes the developer is using an iterative process. Remember, each pass through the sequence of process workflows is called an **iteration**. Thus, from a development perspective, the software lifecycle is a succession of iterations, through which the software develops incrementally.

During the Analysis and Design workflow in an iteration a use case will serve as the primary input artifact. By going through the a series of activities defined in the Analysis and Design workflow, the development team will create an associated use-case realization that describes how a particular use case will be realized.

Review

Review: Analysis and Design Overview

- ♦ What is the purpose of the Analysis and Design Discipline?
- ♦ What are the input and output artifacts?
- ♦ Name and briefly describe the 4+1 Views of Architecture.
- ♦ What is the difference between Analysis and Design?
- ♦ What is architecture?