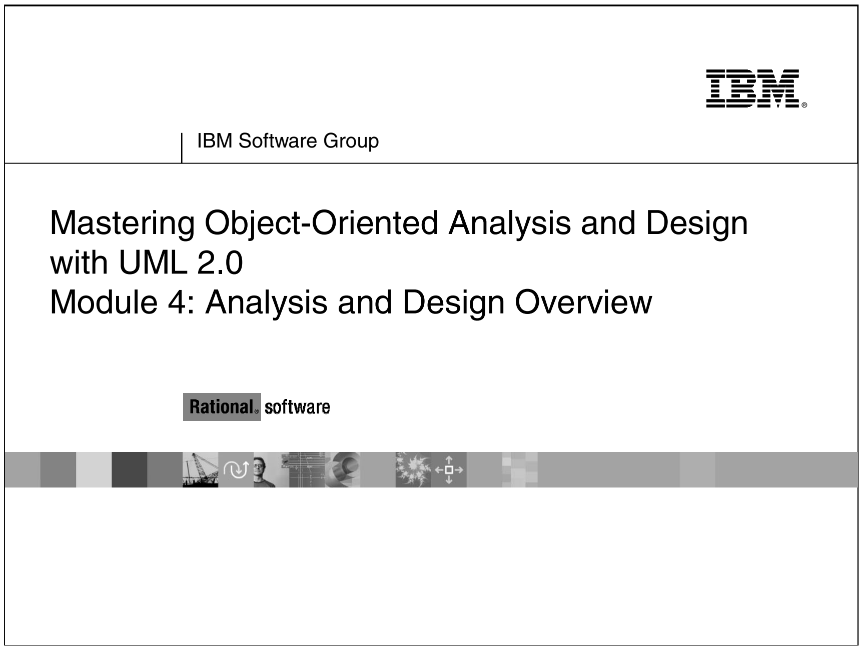
Instructor Notes:



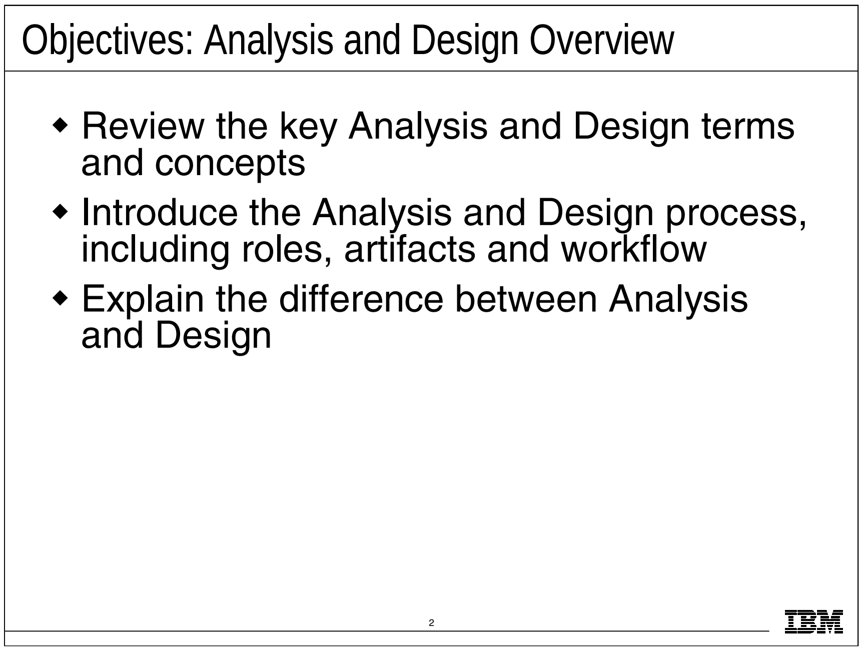
IBM Software Group

Mastering Object-Oriented Analysis and Design with UML 2.0
Module 4: Analysis and Design Overview

Rational. software

Instructor Notes:

Explore with the students the difference between analysis and design.



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.

Instructor Notes:

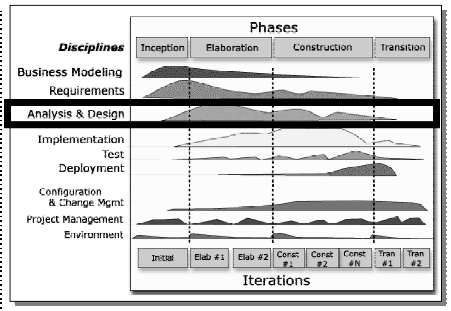
Review the Rational Unified Process Framework and the relationship of the Analysis and Design Discipline to the other disciplines in the Framework.

The Rational Unified Process Framework was introduced in the Requirements Module.

Highlight what part of the overall process we are concentrating on (analysis and design activities in an early elaboration iteration).

Analysis and Design in Context

The purposes of Analysis and Design are to:
- **Transform the requirements into a design of the system-to-be.**
- **Evolve a robust architecture for the system.**
- **Adapt the design to match the implementation environment, designing it for performance.**



3

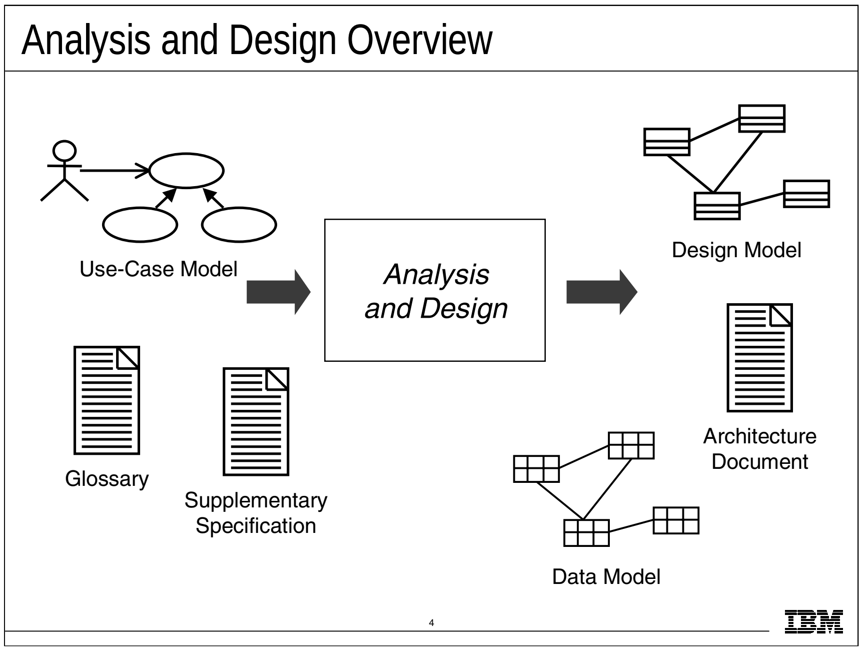
The purposes of Analysis and Design are to:
- Transform the requirements into a system design.
- Evolve a robust architecture for the system.
- Adapt the design to match the implementation environment, designing it for performance.

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).

Instructor Notes:

If you want a separate Analysis Model, then it would be listed as a separate artifact (in addition to the Design Model).

The data model is grayed out since it is not a necessary model in Analysis and Design.



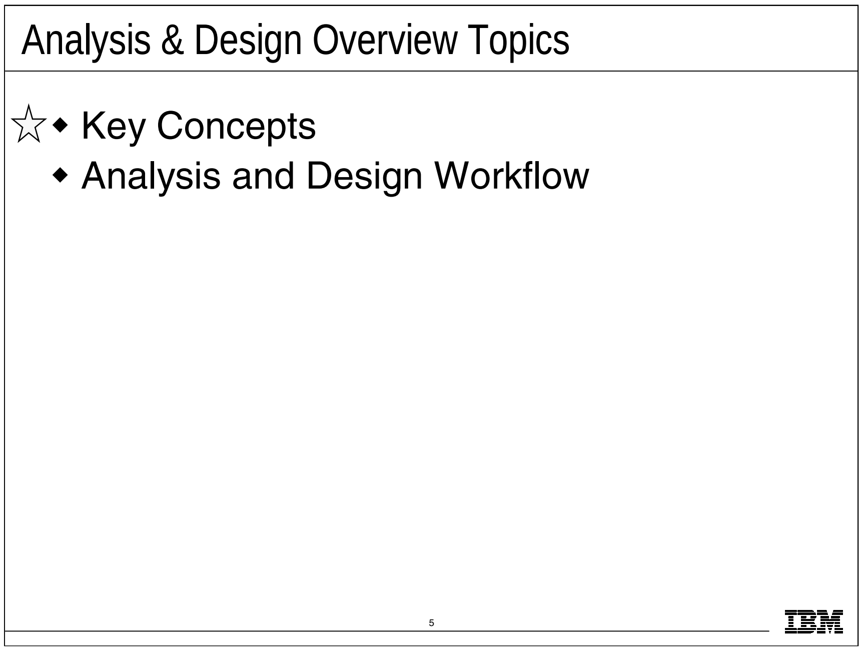
Analysis and Design Overview

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 it is contents and how to interpret them.

Instructor Notes:



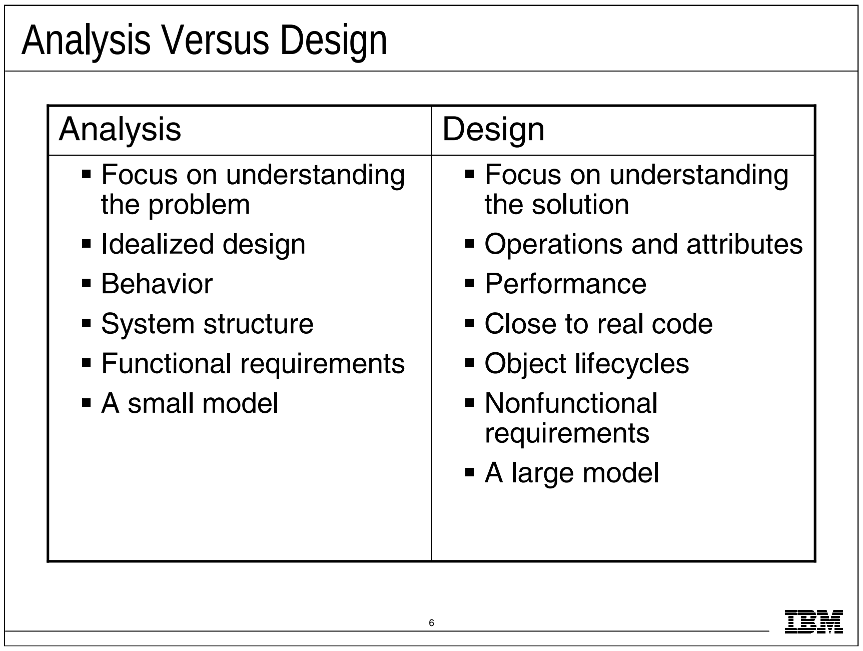
Analysis & Design Overview Topics

☆ ◆ 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	Design
▪ Focus on understanding the problem ▪ Idealized design ▪ Behavior ▪ System structure ▪ Functional requirements ▪ A small model	▪ 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.

Instructor Notes:

An instructor once said: "Analysis vs. Design is important because of the mind-set. If you tried to manage all of the Analysis and Design issues in one go, your brain would explode on all but the most trivial developments. It's just too much to take in, comprehend, and model in one go. So I mentally switch 'OK, Analysis — problem domain — don't care about memory, persistence, databases, language, etc'. Right now it's Design and I do care about all those things — but now I can stop trying to understand the business domain. It's about managing the 7+/-2 things I can think about in one go. It also separates the skills (a bit)."

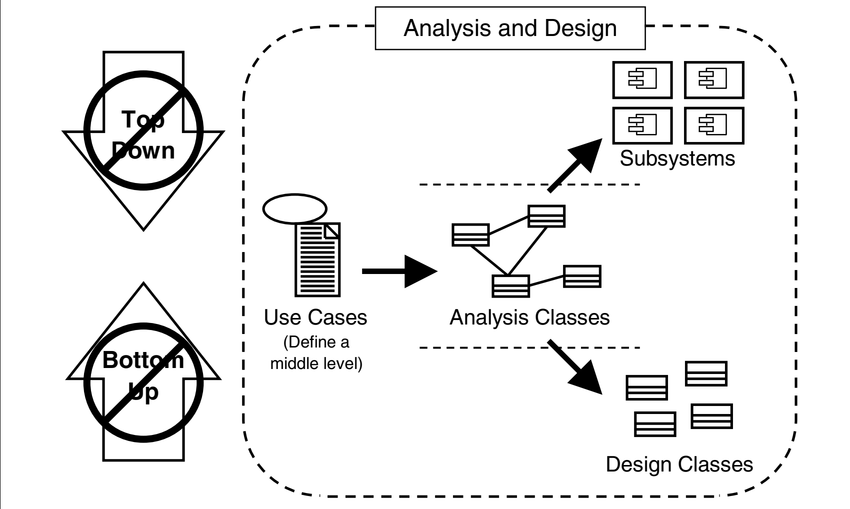
Another instructor once said: "Analysis is the study of and eventual comprehension of 'some thing' [the problem space]. But to understand it, we must invent some entities to hold onto the thing that we have just grasped — this inventive process is Design. That is, human cognitive thinking is actually interwoven Analysis/Design and any attempt to separate them is really only an idealization. Therefore, if I try to apply two separate 'steps' in the production of, say, a class diagram [and call those steps A and then D], I will introduce ambiguity into my process because cognitive process forces one to 'produce a solution' in order to 'understand a problem' - that D is necessary to have any results from A."

Think of Analysis as an idealized design step — where we ignore several complicating issues. This can turn into a religious debate.

Instructor Notes:

Mention to the students that normally with subsystems diagrams such as the one depicted here, you would include <<subsystem>> text.

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



7

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.

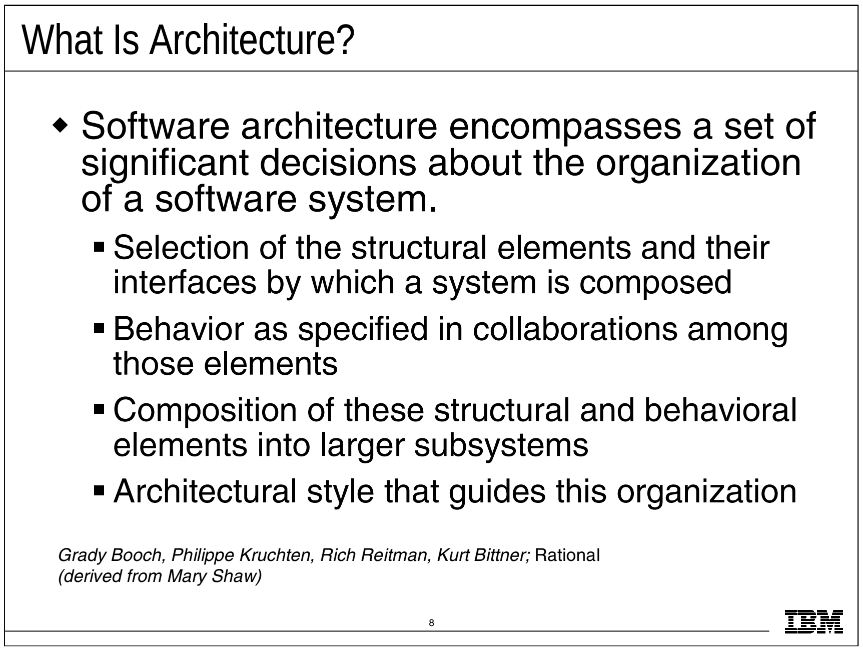
Defining subsystems is moving up and defining design classes is moving down.

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.

Instructor Notes:

Emphasize that the rationale for the architectural decisions is very important. Remember that the RUP is use-case driven AND architecture-centric. Therefore, the architecture is going to drive design activities.



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:

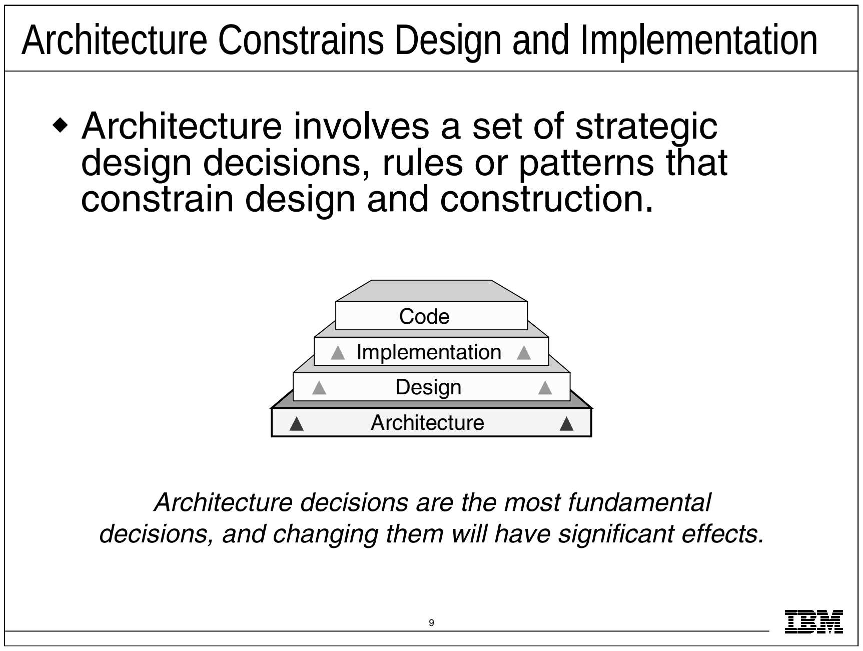
Architecture = Elements + Form + Rationale.

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.

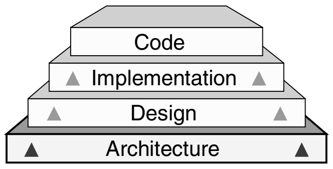
Instructor Notes:

If architecture is strategic design, then the rest of the design is the tactical design.



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).

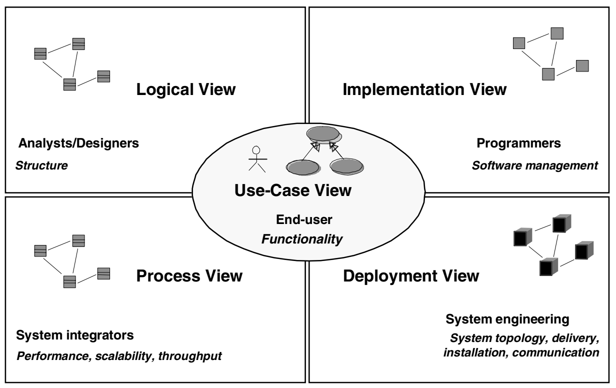
Instructor Notes:

Discuss the 4+1 views. These are covered in detail in the Rational Unified Process.

The 4+1 model is introduced here because it is important for the students to see how the views fit together up front, in order to set context. The individual views are addressed in the specific architecture modules:

•The Logical View will be discussed in the Architectural Analysis and Identify Design Elements module.
•The Process View will be discussed in the Describe Concurrency module.
•The Deployment View will be discussed in the Describe Distribution module.
•The Implementation View will be discussed briefly in the Identify Design Elements module; however, the Implementation View is developed during Implementation and is thus considered out of scope for this Analysis and Design course.

Software Architecture: The “4+1 View” Model



10

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

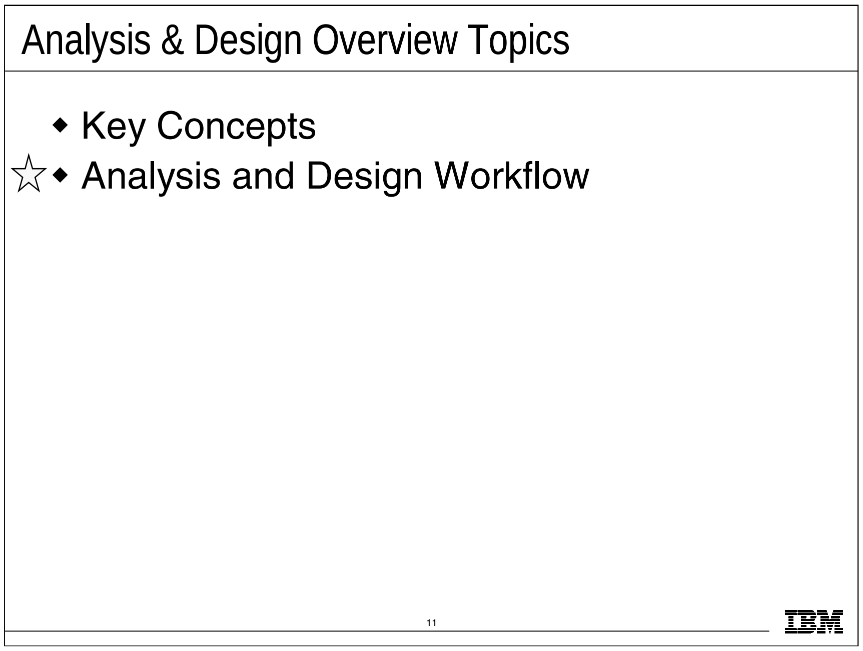
Architecture is many things to many different interested parties. On a particular project, there are usually multiple stakeholders, each with their own concerns and view of the system to be developed. The goal is to provide each of these stakeholders with a view of the system that addresses their concerns,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

Instructor Notes:



Analysis & Design Overview Topics

- Key Concepts
- ☆ Analysis and Design Workflow

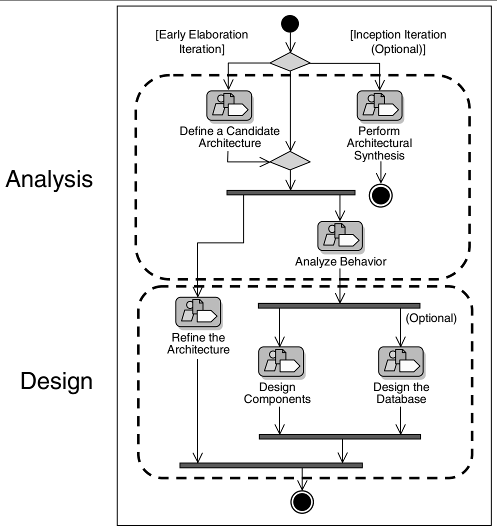
11

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

Instructor Notes:

Each workflow contains RUP activities that can take place. RUP activities can exist in many different workflow activities. For example, Use-Case Analysis can be found in both Define a Candidate Architecture and Analyze Behavior.

Analysis and Design Workflow


[Early Elaboration Iteration]
[Inception Iteration (Optional)]
Define a Candidate Architecture
Perform Architectural Synthesis
Analysis
Analyze Behavior
Refine the Architecture
(Optional)
Design
Design Components
Design the Database

12

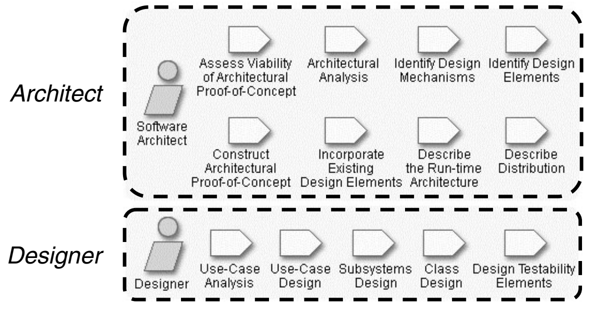
A mere enumeration of all workers, activities, and artifacts does not constitute a process. We need a way to describe the activities, some valuable result, and interactions between workers. A *workflow* is a sequence of activities that produces a result of observable value.

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


Architect
Software Architect
Assess Viability of Architectural Proof-of-Concept
Architectural Analysis
Identify Design Mechanisms
Identify Design Elements
Construct Architectural Proof-of-Concept
Incorporate Existing Design Elements
Describe the Run-time Architecture
Describe Distribution
Designer
Designer
Use-Case Analysis
Use-Case Design
Subsystems Design
Class Design
Design Testability Elements

13

Instructor Notes:

Walk the students through the activities reviewing the meaning of the representational icons (for example, workers and activities). Activities can be considered operations on the workers.

The order shown is not the order in which the activities can be executed. The Analysis and Design workflow helps to dictate order.

The process as described in this course develops a Design Model, not a separate Analysis Model. To maintain a separate Analysis Model, some modifications to the process are necessary, the discussion of which is beyond the scope of this course.

Emphasize the Use-Case Analysis activity and how it describes a process for finding classes and objects from use cases. Some people have called it: "closing the traceability gap." Use cases and scenarios help you get from requirements to objects.

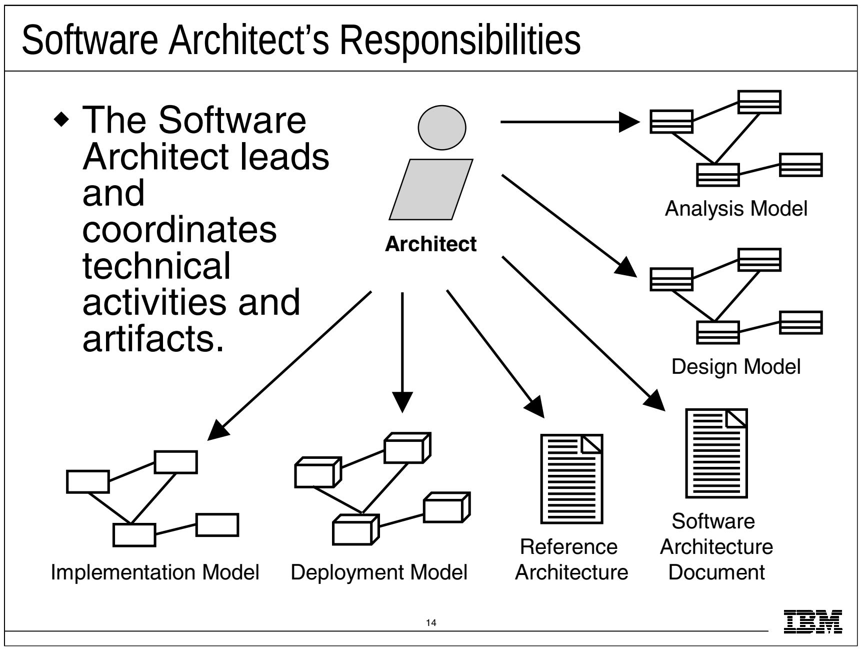
Remember, for Analysis and Design, we start out with the Use-Case Model and the supplementary specifications from the Requirements Discipline and end up with the Design Model that serves as an abstraction of the source code.

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.

Instructor Notes:

Here are some activities in which the Software Architect takes the lead.



Software Architect's Responsibilities

- The Software Architect leads and coordinates technical activities and artifacts.

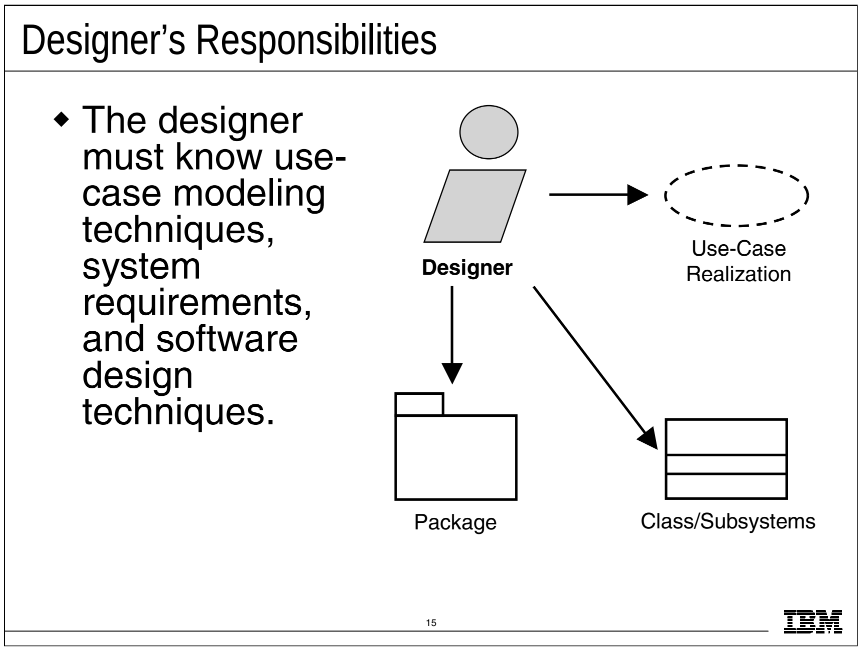
The software architect role leads and coordinates technical activities and artifacts throughout the project. The software architect establishes the overall structure for each architectural view: the decomposition of the view, the grouping of elements, and the interfaces between these major groupings. 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.

Instructor Notes:

Many of the students in this class are probably designers and will be interested in learning about the designer's responsibilities. Emphasize the designer's role and importance.



Designer's Responsibilities

- The designer must know use-case modeling techniques, system requirements, and software design techniques.

Designer
Use-Case Realization
Package
Class/Subsystems
15

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:

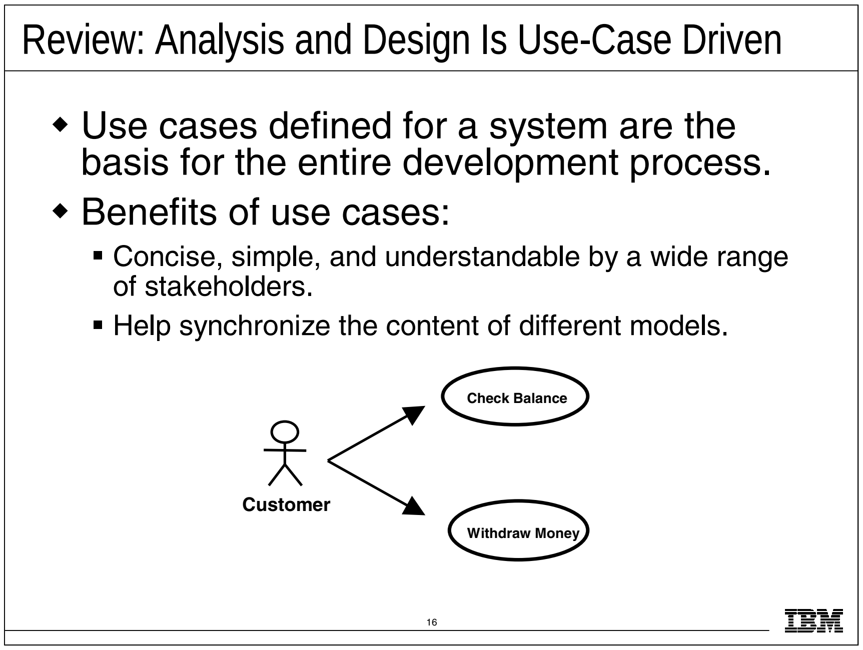
- Use-case modeling techniques
- System requirements
- Software design techniques, including object-oriented Analysis and Design techniques, and the Unified Modeling Language
- Technologies with which the system will be implemented

In addition, the designer must:

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

Instructor Notes:

This slide first appeared in the Essentials of Visual Modeling with UML course. The use-case realizations (definition on the next slide) that the students will be developing are an example of how the 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.

Customer
Check Balance
Withdraw Money
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.

Instructor Notes:

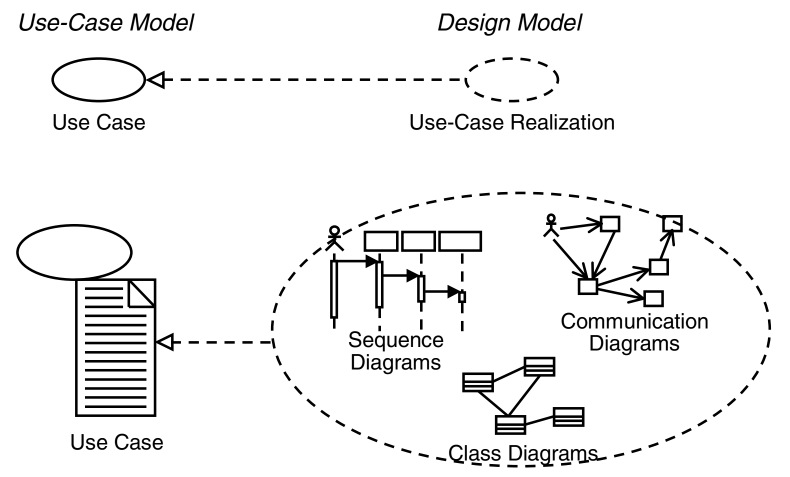
If you consider use cases and scenarios to be black box descriptions of the system, then the use-case realizations are the associated white box descriptions.

Use-case realizations are introduced here because they may be mentioned when providing the overview of the Analysis and Design workflow (the developing of use-case realizations is the main goal of Use-Case Analysis and the refinement of these use-case realizations is the main goal of Use-Case Design). The use case realizations are identified in Architectural Analysis, initially developed in Use-Case Analysis and then refined in Use-Case Design.

The Rational Unified Process includes templates for the the use-case realization, as well as the use-case realization icon.

In Rose, you cannot draw realizations between use cases, so a stereotyped association must be used instead. In Rose, use-case realizations are modeled as stereotyped use cases. This is discussed in more detail in the Use-Case Analysis module.

What Is a Use-Case Realization?



17

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.

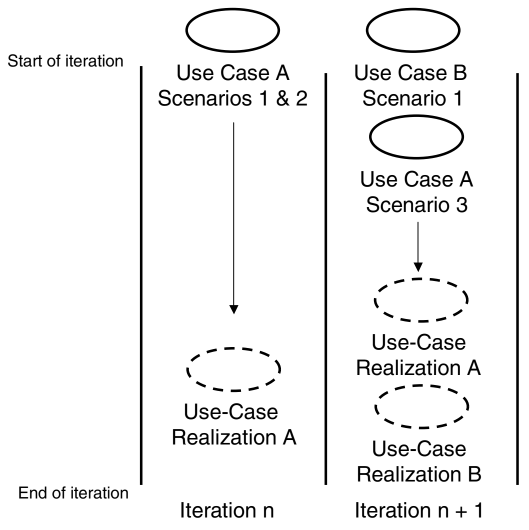
Within the UML, a use-case realization can be represented using a set of diagrams that model the context of the collaboration (the classes/objects that implement the use case and their relationships — class diagrams), and the interactions of the collaborations (how these classes/objects interact to perform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

Instructor Notes:

The purpose of this slide is to demonstrate that a team does not realize every use case in one giant iteration. Rather, the project will have a series of iterations in which one or more use cases will be realized.

Analysis and Design in an Iterative Process



18

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.

Instructor Notes:

1. The purpose of the Analysis and Design disciplines is 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.

2. The input artifacts are: the Use-Case Model, the glossary and the supplementary specification. The outputs are the design model, the data model and the architecture document.

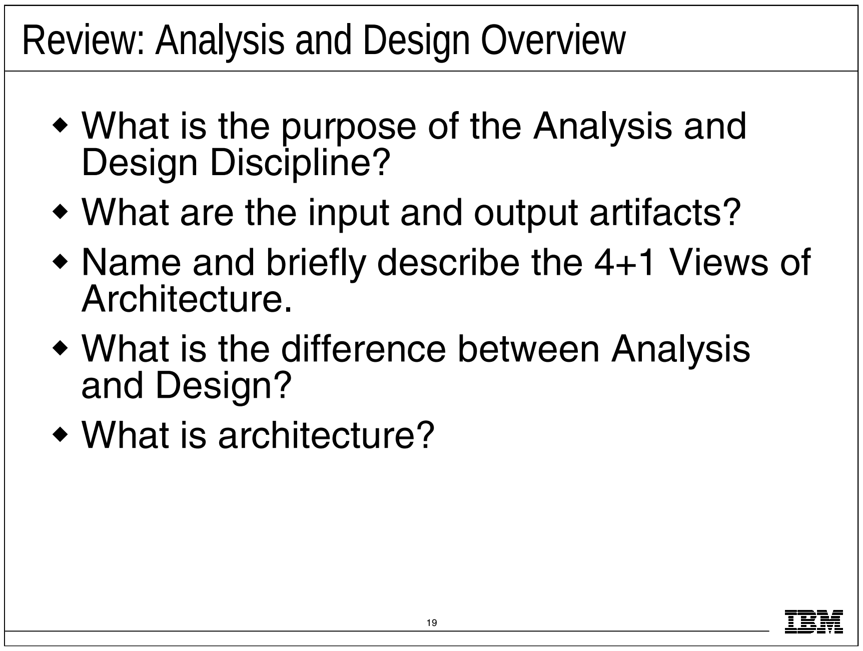
3. The 4+1 view of architecture includes the following views and their roles:

Use Case (functionality); **Logical** (structure); **Implementation** (software management); **Process** (performance, scalability, throughput); **Deployment** (system topology, delivery, installation communication).

4. The differences between Analysis and Design are ones of focus and emphasis. Slide 6 lists the things that you focus on in Analysis versus Design.

5. Software architecture encompasses a set of significant decisions about the organization of a software system. Architectures can be viewed as a set of key design decisions.

The architecture is the initial set of constraints placed on the system.



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?

19

Instructor Notes: