Introducing Modeling and the Software Development Process

Objectives

After completing this lesson, you should be able to do the following:

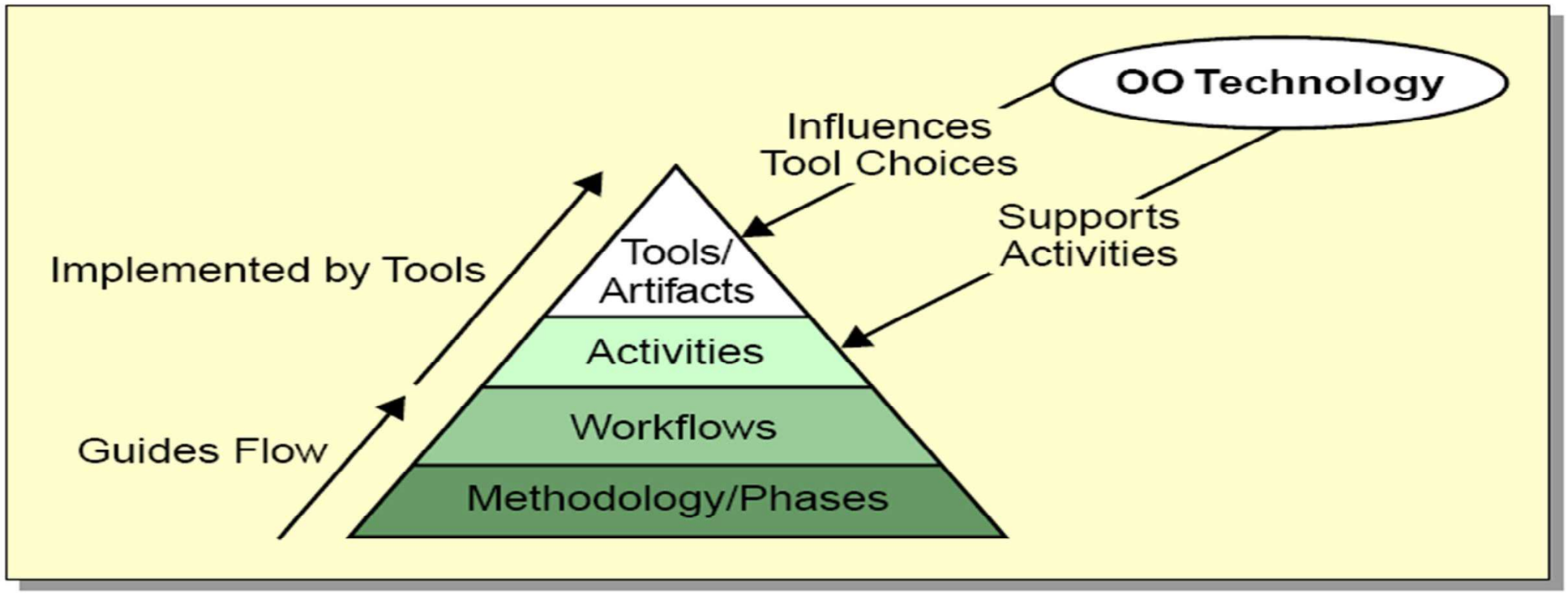
- Describe the Object-Oriented Software Development (OOSD) process
- Describe how modeling supports the OOSD process
- Describe the benefits of modeling software
- Explain the purpose, activities, and artifacts of the following OOSD workflows: Requirements Gathering, Requirements Analysis, Architecture, Design, Implementation, Testing, and Deployment

Describing Software Methodology

A methodology is “a body of methods, rules, and postulates employed by a discipline”

- In OOSD, methodology refers to the highest-level organization of a software project.
- This organization can be decomposed into medium-level phases. Phases are decomposed into workflows (disciplines). Workflows are decomposed into activities.
- Activities transform the artifacts from one workflow to another. The output of one workflow becomes the input into the next.
- The final artifact is a working software system that satisfies the initial artifacts: the system requirements.

The OOSD Hierarchy

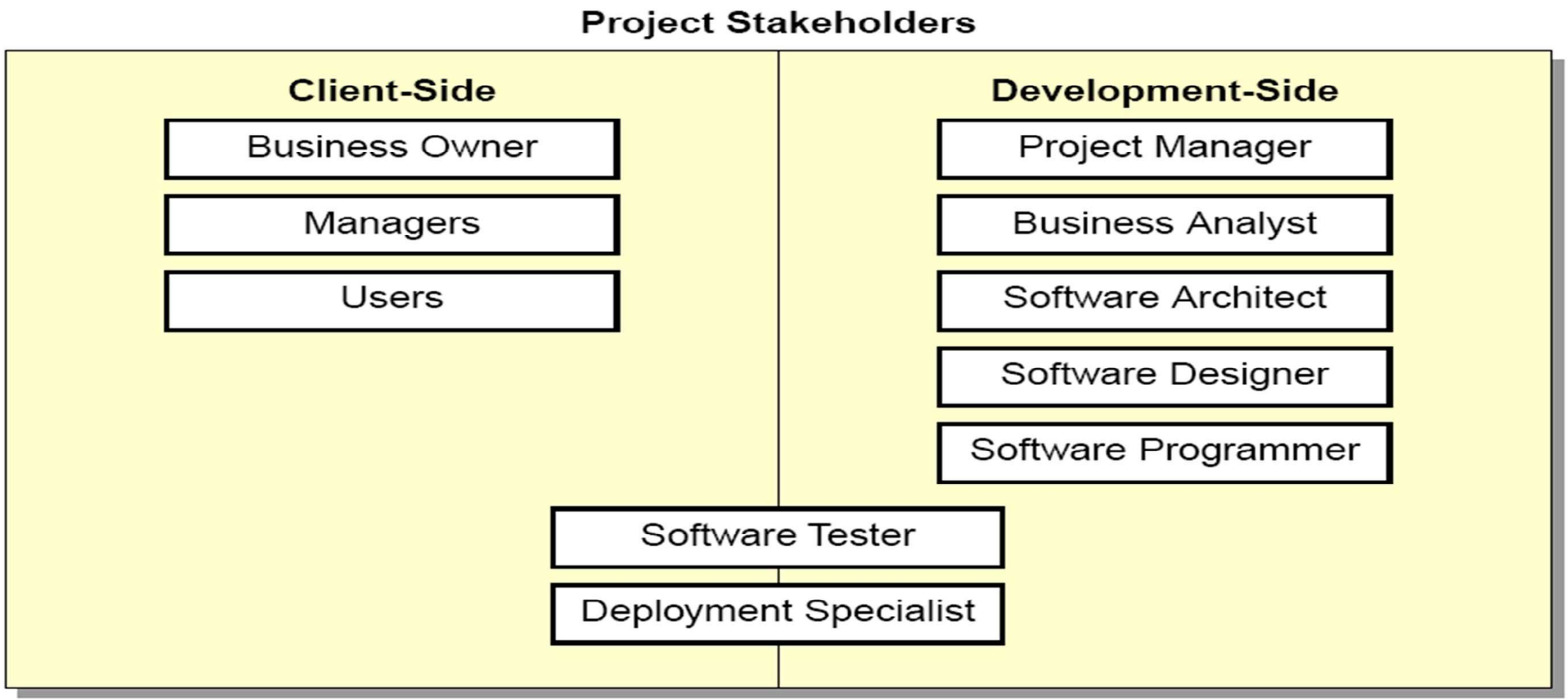


Listing the Workflows of the OOSD Process

Software development has traditionally encompassed the following workflows:

- Requirements Gathering
- Requirements Analysis
- Architecture
- Design
- Implementation
- Testing
- Deployment

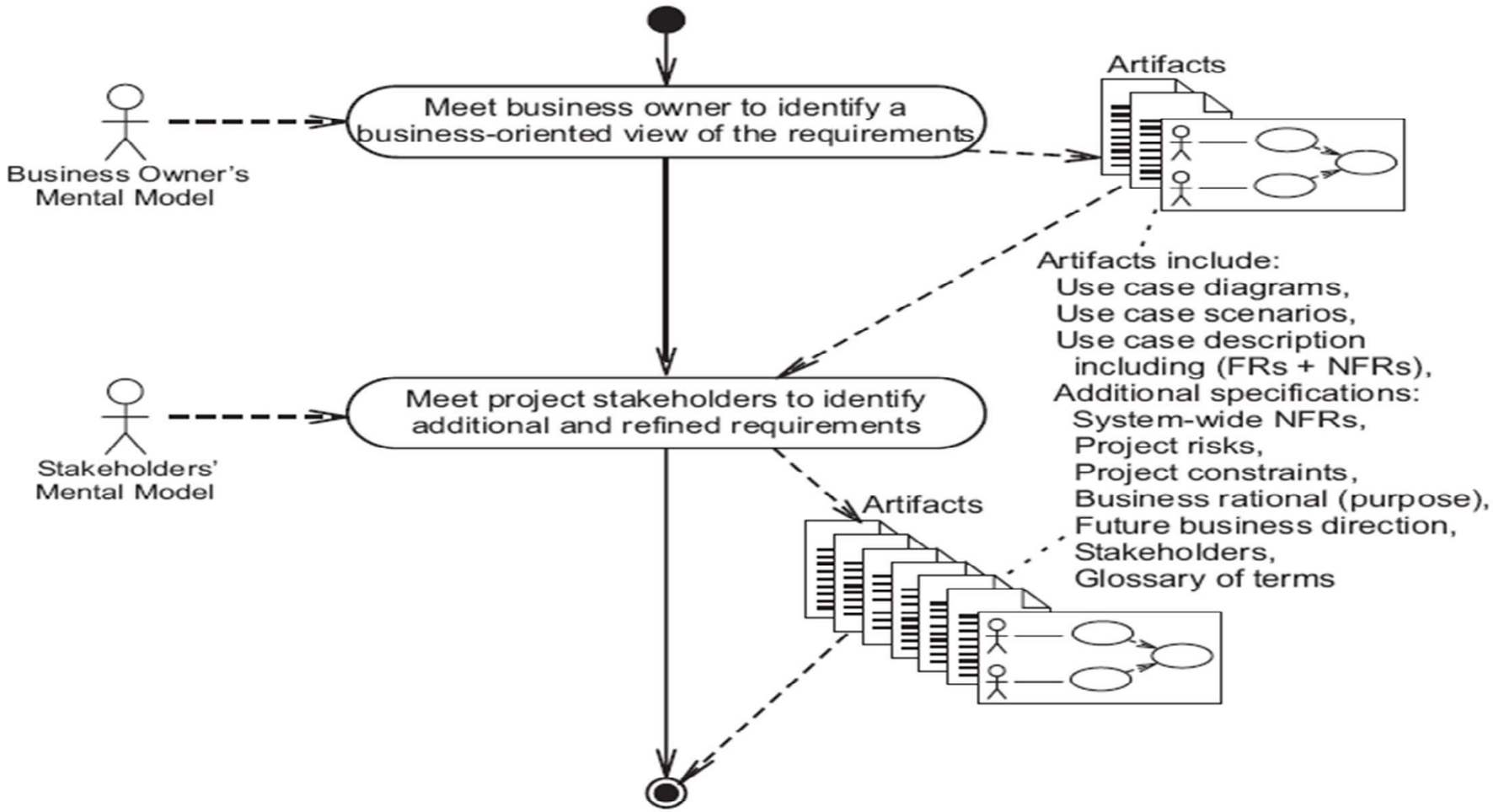
Describing the Software Team Job Roles



Exploring the Requirements Gathering

Workflow	Purpose	Description
Requirements Gathering	Determine *what* the system must do	Determine: • With whom the system interacts (actor) • What behaviors (called use cases) that the system must support • Detailed behavior of each use case, which includes the low-level functional requirements (FRs) • Non-functional requirements (NFRs)

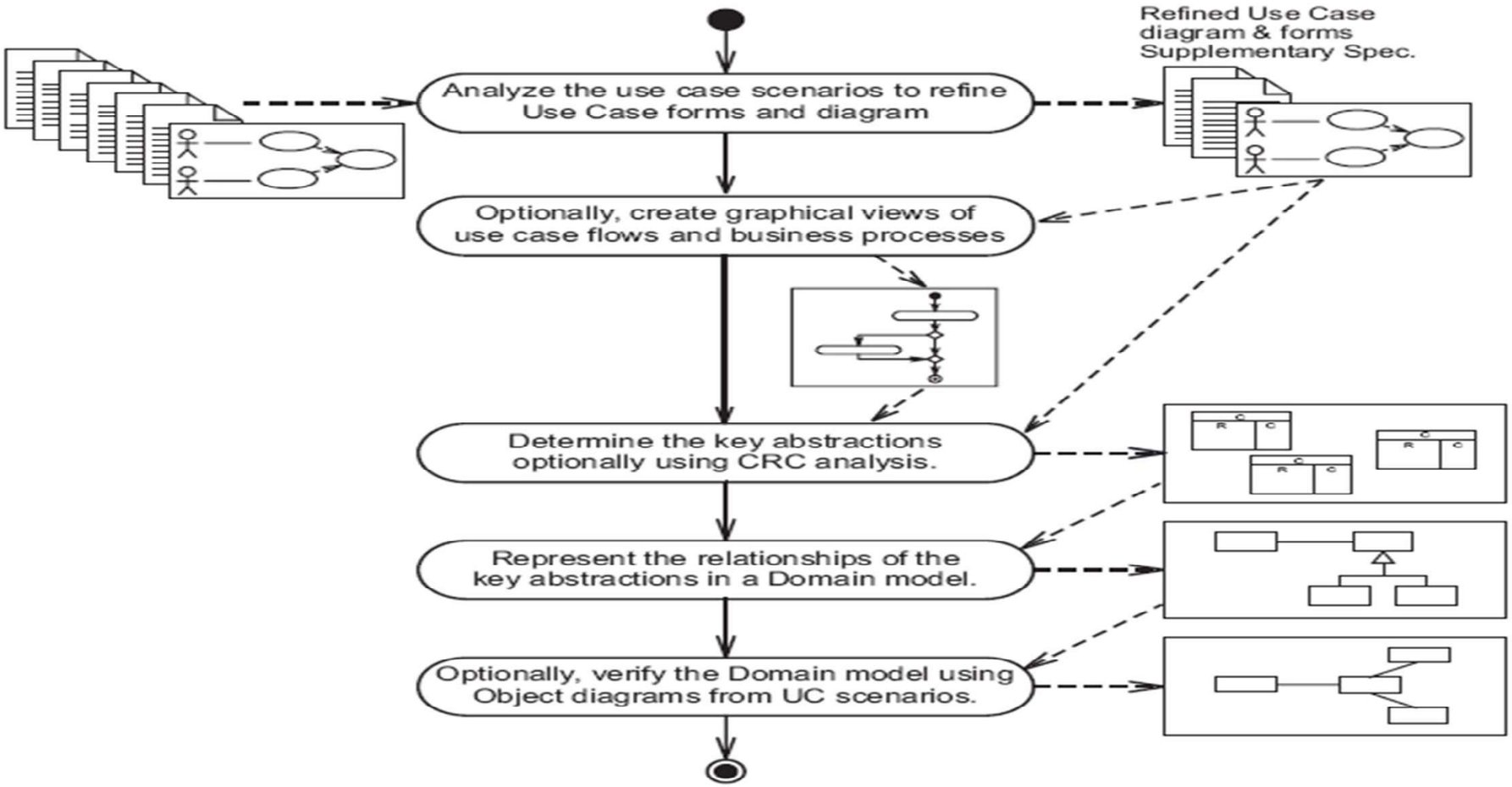
Activities and Artifacts of the Requirements



Exploring the Requirements Analysis

Workflow	Purpose	Description
Requirements Gathering	Determine *what* the system must do	
Requirements Analysis	Model the existing business processes	Determine: • The detailed behavior of each use case • Supplementary use cases • The key abstractions that exist in the current increment of the problem domain • A business domain class diagram

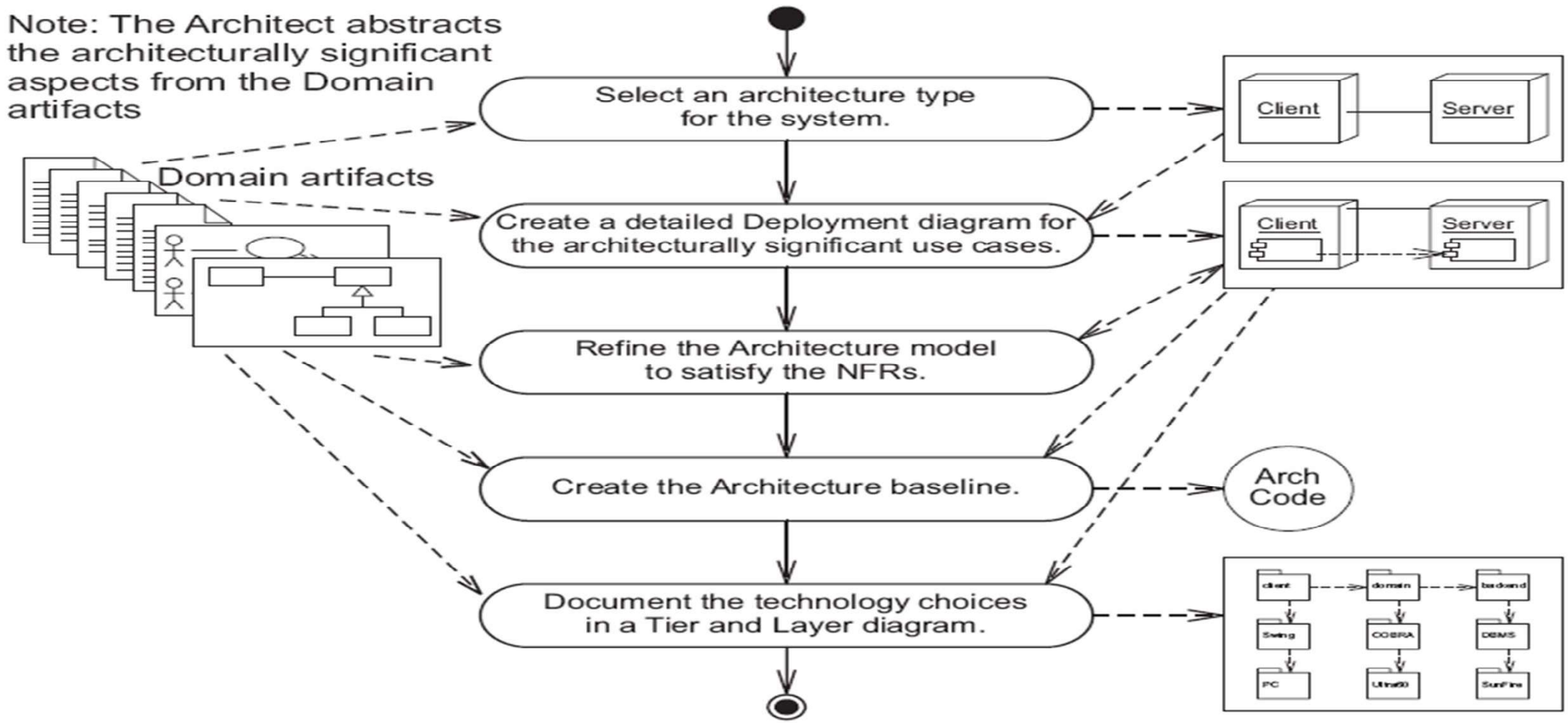
Activities and Artifacts of the Requirements



Exploring the ArchitectureWorkflow

Workflow	Purpose	Description
Requirements Gathering	Determine *what* the system must do	
Requirements Analysis	Model the existing business processes	
Architecture	Model the high-level system structure to satisfy the NFRs	• Develop the highest-level structure of the software solution • Identify the technologies that will support the Architecture model • Elaborate the Architecture model with Architectural patterns to satisfy NFRs

Activities and Artifacts of the Architecture



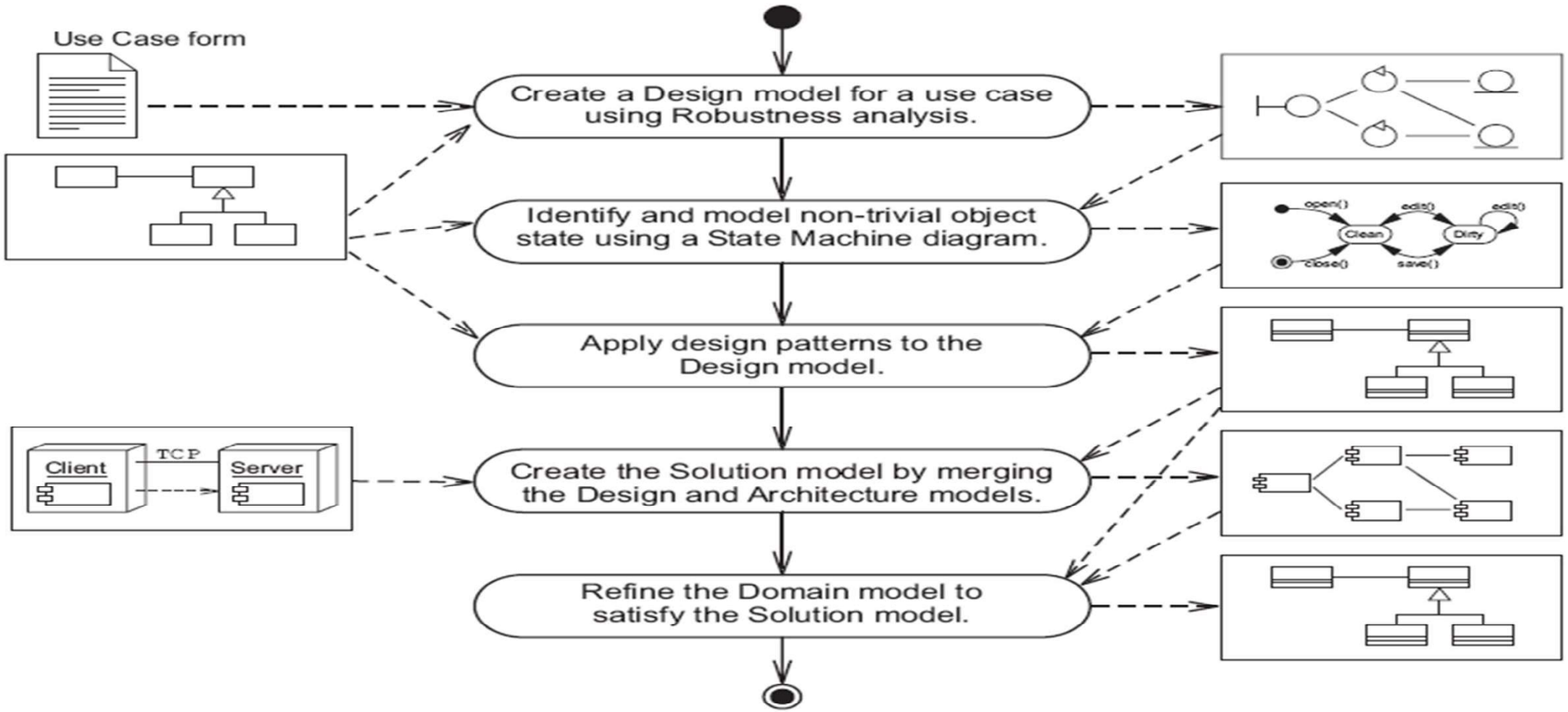
Exploring the Design Workflow

Workflow	Purpose	Description
Requirements Gathering	Determine *what* the system must do	
Requirements Analysis	Model the existing business processes	
Architecture	Model the high-level system structure to satisfy the NFRs	

Exploring the Design Workflow

Workflow	Purpose	Description
Design	Model *how* the system will support the use cases	• Create a Design model for a use case using Interaction diagrams • Identify and model objects with non-trivial states using a State Machine diagram • Apply design patterns to the Design model • Create a Solution model by merging the Design and Architecture models • Refine the Domain model

Activities and Artifacts of the Design



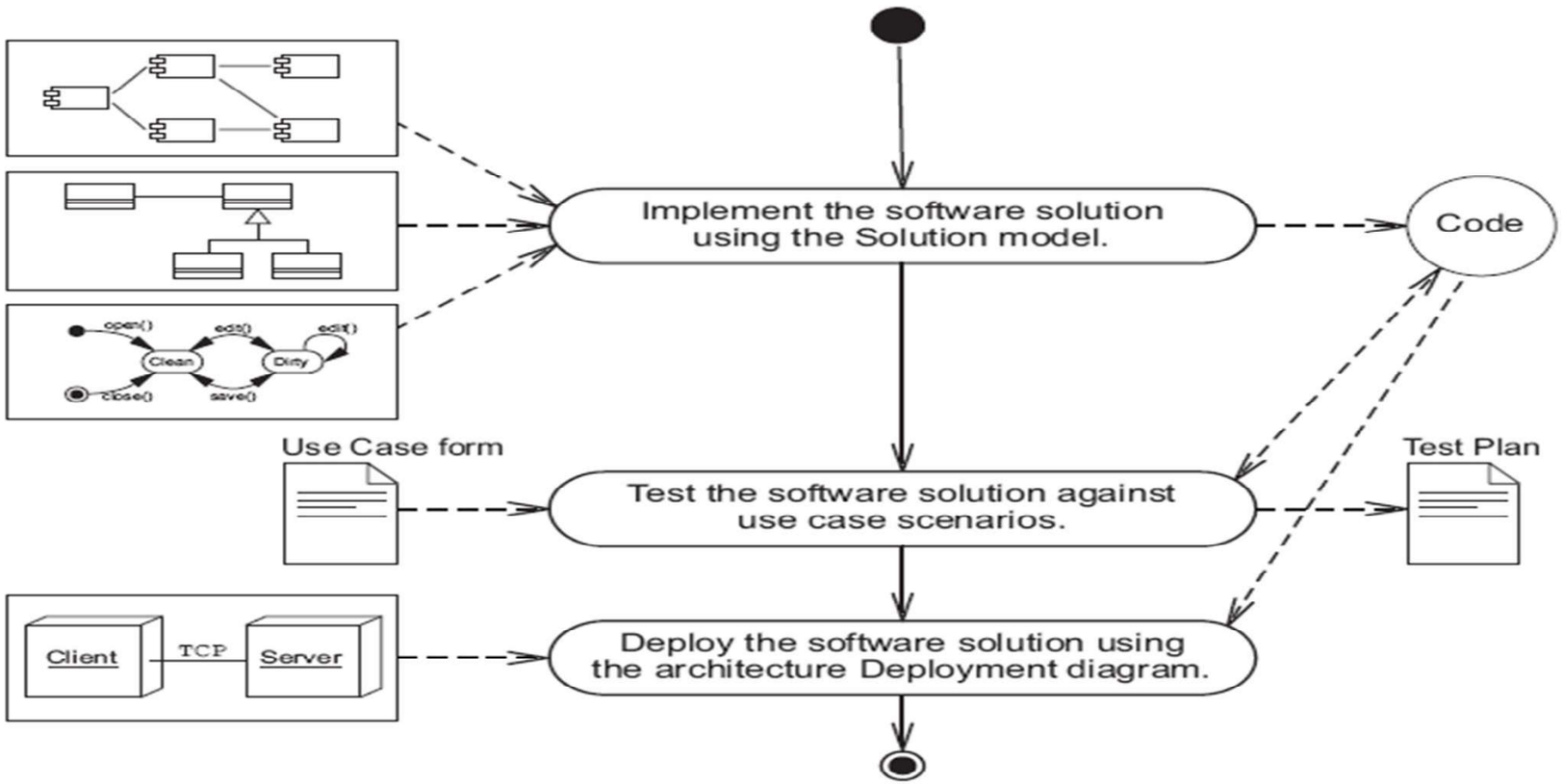
Exploring the Implementation, Testing, and

Workflow	Purpose	Description
Requirements Gathering	Determine *what* the system must do	
Requirements Analysis	Model the existing business processes	
Architecture	Model the high-level system structure to satisfy the NFRs	
Design	Model *how* the system will support the use cases	

Exploring the Implementation, Testing, and

Workflow	Purpose	Description
Implementation, Testing, and Deployment	Implement, test, and deploy the system	• Implement the software • Perform testing • Deploy the software to the production environment

Activities and Artifacts of the Implementation, Testing, and

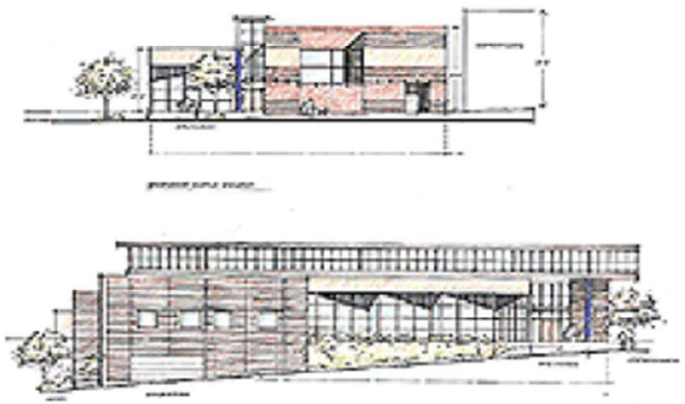


Exploring the Benefits of Modeling Software

- The inception of every software project starts as an idea in someone's mind.
- To construct a realization of that idea, the development team must create a series of conceptual models that transform the idea into a production system.

What is a Model?

- “A model is a simplification of reality.”



- Amodel is an abstract conceptualization of some entity (such as a building) or a system (such as software).
- Different views show the model from different perspectives.

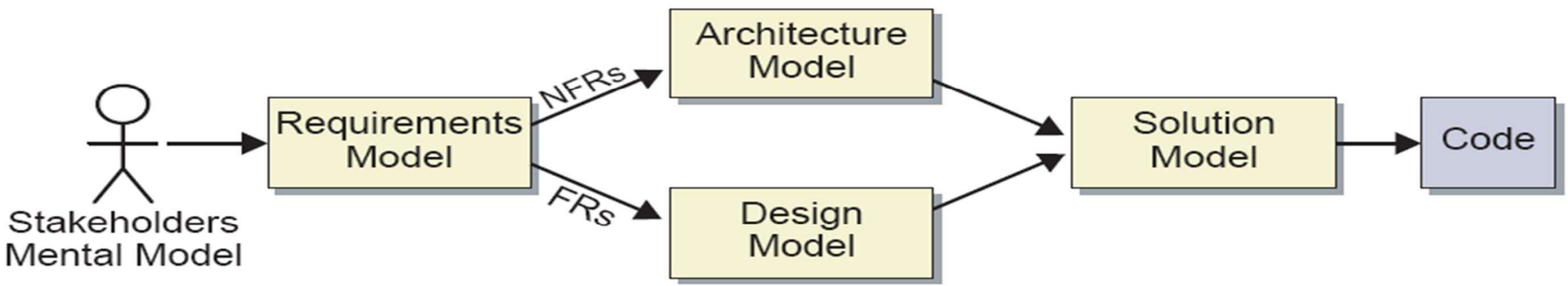
Why Model Software?

"We build models so that we can better understand the system we are developing."

Specifically, modeling enables us to:

- Visualize new or existing systems
- Communicate decisions to the project stakeholders
- Document the decisions made in each OOSD workflow
- Specify the structure (static) and behavior (dynamic) elements of a system
- Use a template for constructing the software solution

- Software development can be viewed as a series of transformations from the Stakeholder's mental model to the actual code:



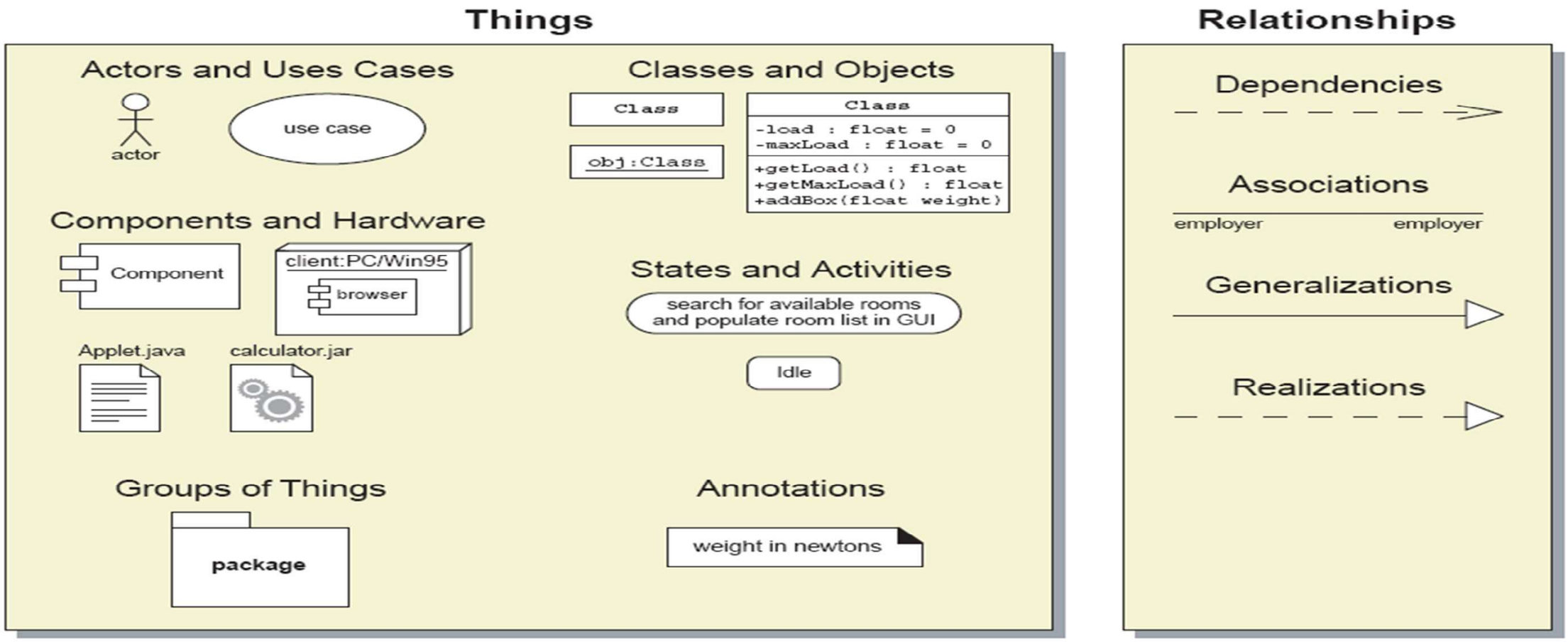
Defining the UML

"The Unified Modeling Language (UML) is a graphical language for visualizing, specifying, constructing, and documenting the artifacts of a software-intensive system."

Using the UML, a model is composed of:

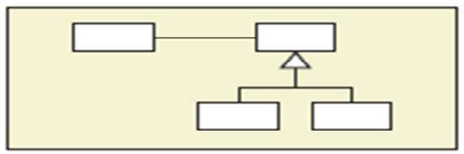
- Elements (things and relationships)
- Diagrams (built from elements)
- Views (diagrams showing different perspectives of a model)

UML Elements

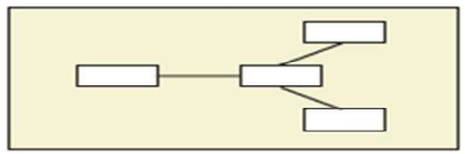


UML Diagrams

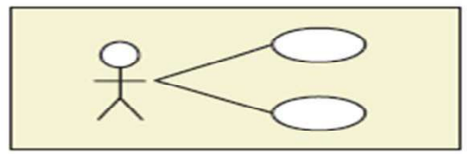
Class



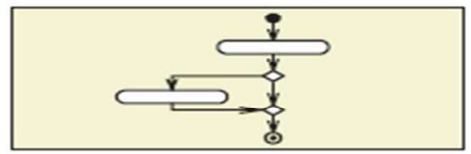
Object



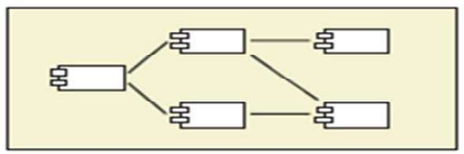
Use Case



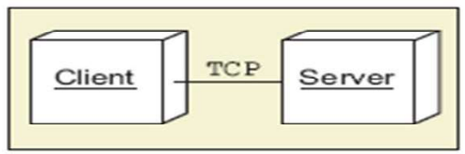
Activity



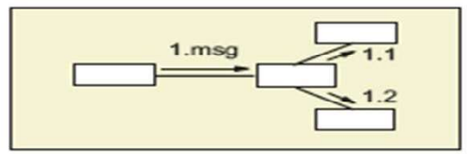
Component



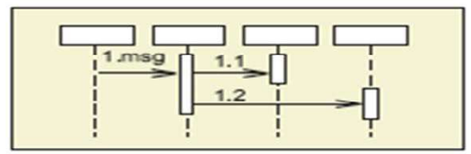
Deployment



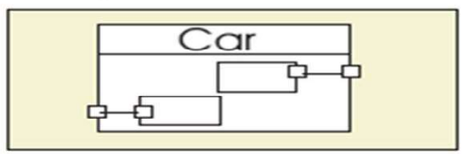
Communication



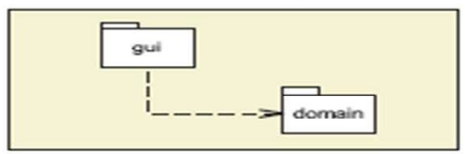
Sequence



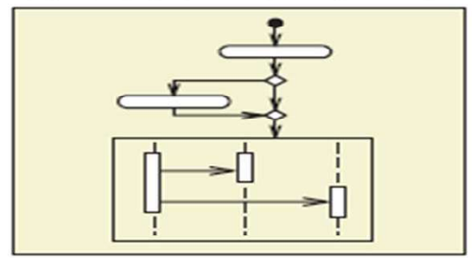
Composite Structure



Package

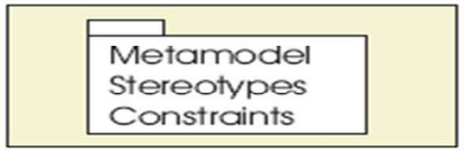


Interaction Overview

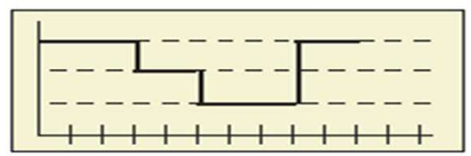


State Machine

Profile



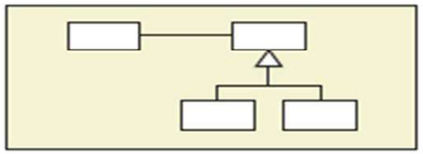
Timing



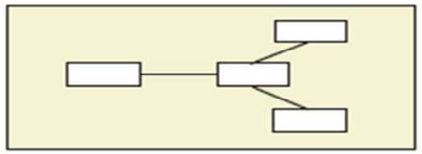
UML Diagram Categories

Structural

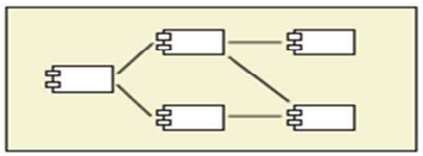
Class



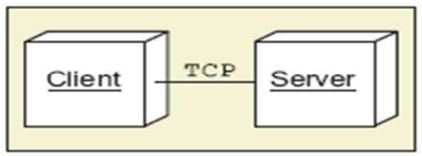
Object



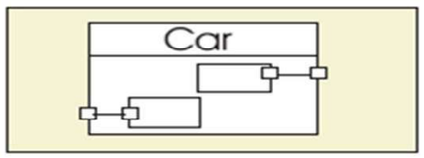
Component



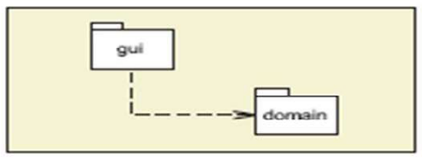
Deployment



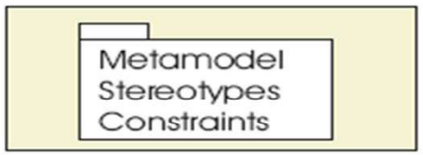
Composite Structure



Package

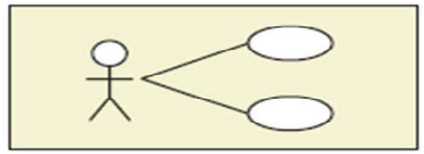


Profile

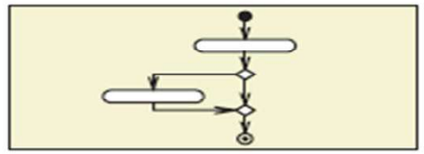


Behavioral

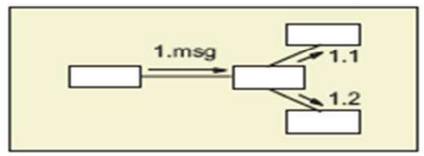
Use Case



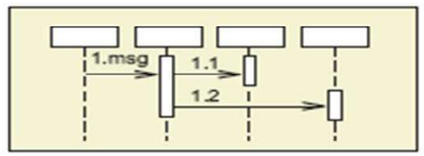
Activity



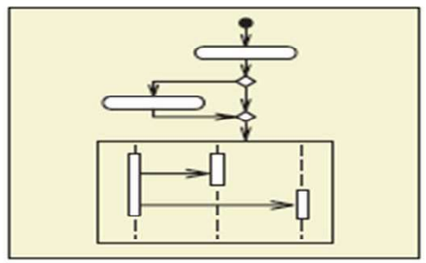
Communication



Sequence

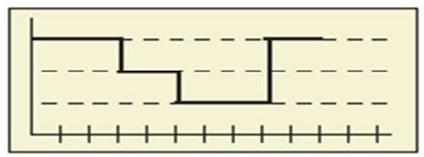


Interaction Overview



State Machine

Timing



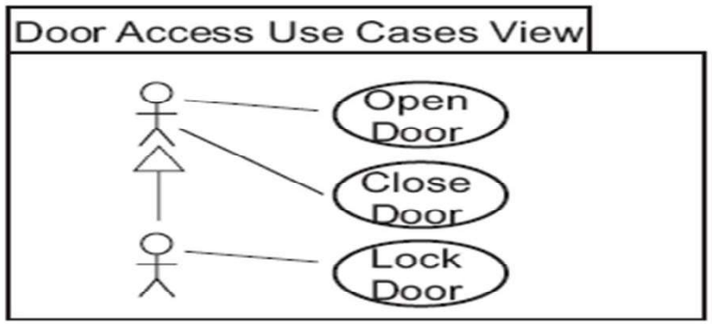
Common UML Elements and Connectors

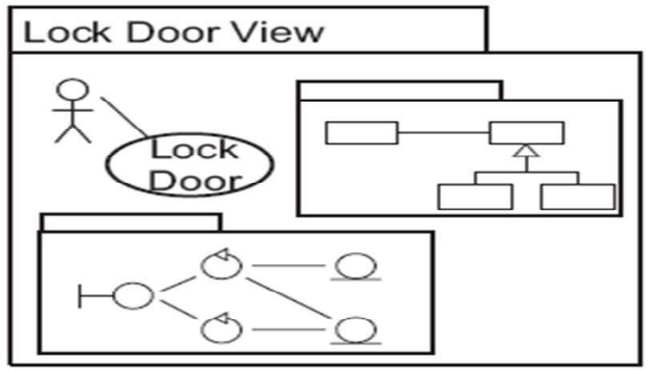
UML has a few elements and connectors that are common across UML diagrams. These include:

- Package
- Note
- Dependency
- Stereotypes

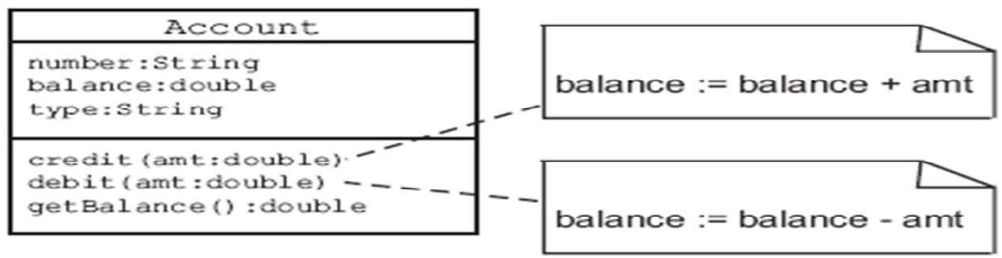
Packages and Notes

Package

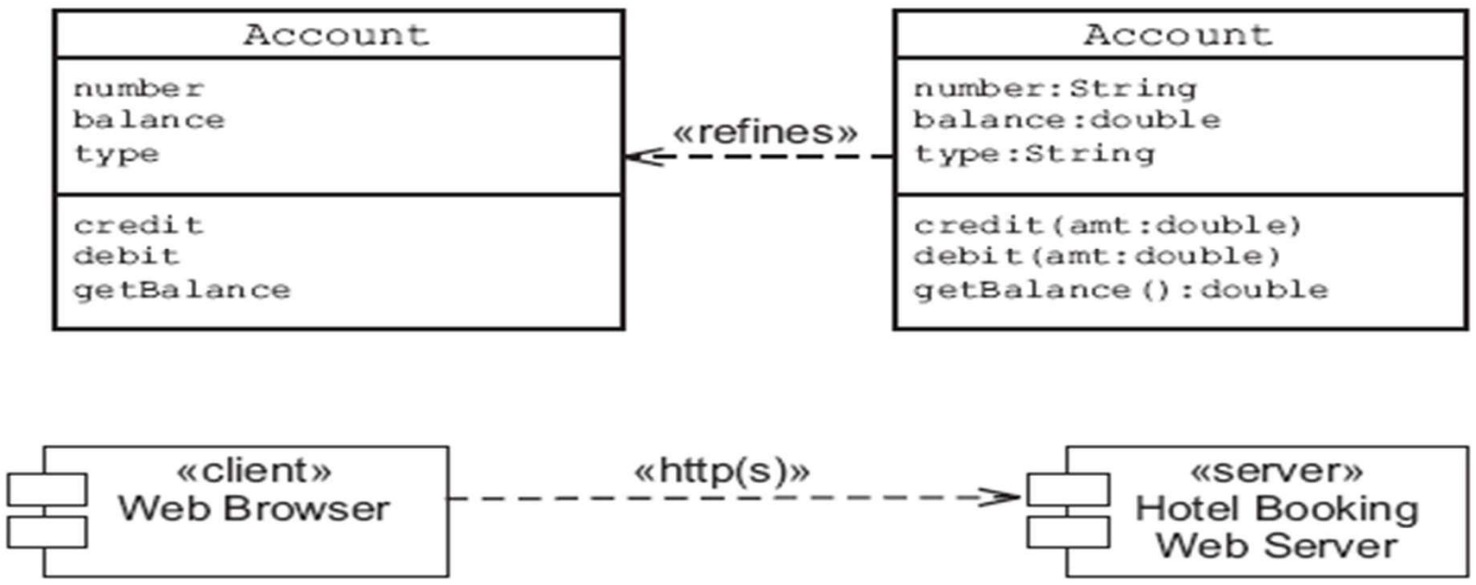




Notes



Dependency and Stereotype



What UML Is and Is Not

UML is not:	**But it:**
Used to create an executable model	Can be used to generate code skeletons
A programming language	Maps to most OO languages
A methodology	Can be used as a tool within the activities of a methodology

UML Tools

UML itself is a tool. You can create UML diagrams on paper or a white board. However, software tools are available to:

- Provide computer-aided drawing of UML diagrams
- Support (or enforce) semantic verification of diagrams
- Provide support for a specific methodology
- Generate code skeletons from the UML diagrams
- Organize all of the diagrams for a project
- Automatic generation of modeling elements for design patterns, Java™ Platform, Enterprise Edition (Java™ EE platform) components, and so on.

Summary

In this lesson, you should have learned the following:

- Describe the Object-Oriented Software Development (OOSD) process
- Describe how modeling supports the OOSD process
- Describe the benefits of modeling software
- Explain the purpose, activities, and artifacts of the following OOSD workflows: Requirements Gathering, Requirements Analysis, Architecture, Design, Implementation, Testing, and Deployment

Practice : Overview

This practice covers the following topics:

- Introducing Modeling and Software Development Process