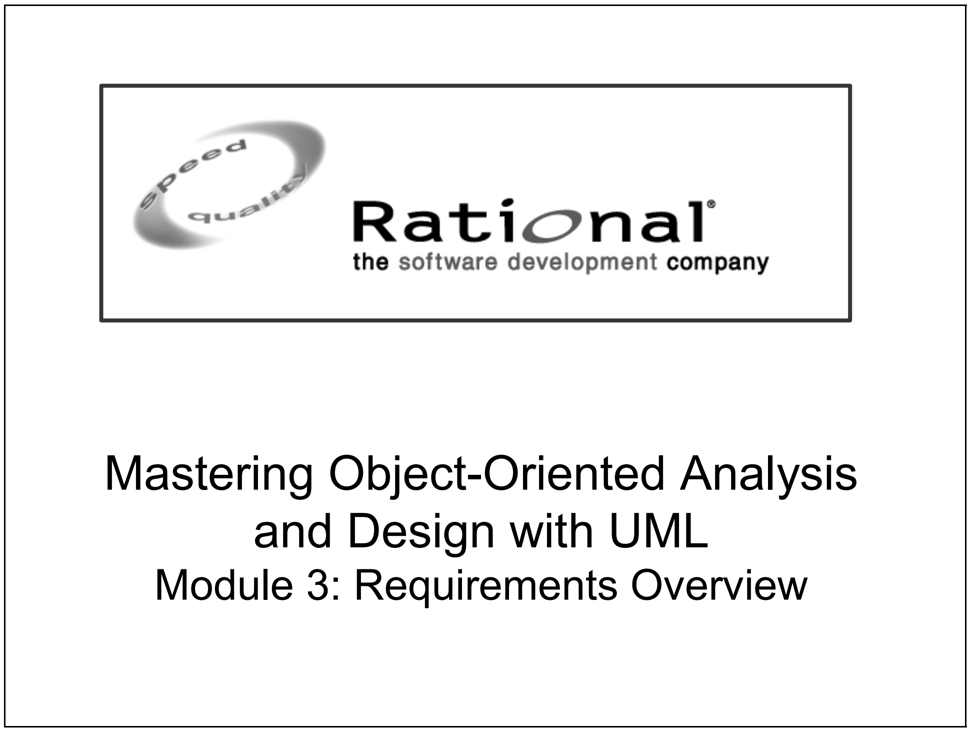
▸ ▸ ▸ Module 3 Requirements Overview

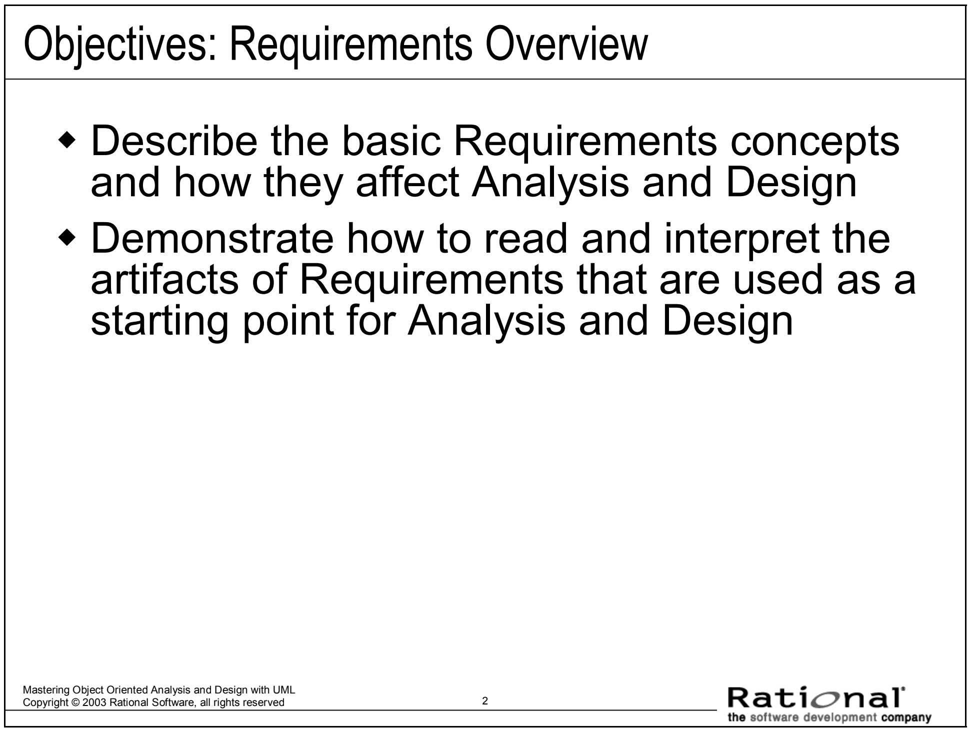


Rational
the software development company

Mastering Object-Oriented Analysis and Design with UML
Module 3: Requirements Overview

Topics

Objectives: Requirements Overview



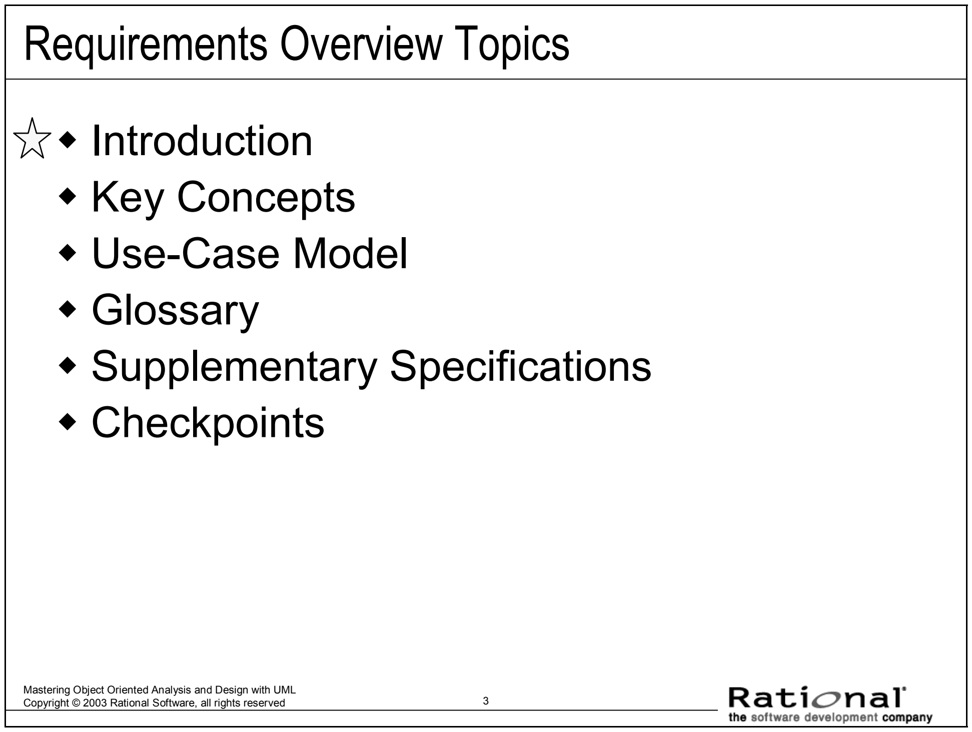
Objectives: Requirements Overview

- Describe the basic Requirements concepts and how they affect Analysis and Design
- Demonstrate how to read and interpret the artifacts of Requirements that are used as a starting point for Analysis and Design

This Requirements Overview module provides an overview of the activities that immediately precede Analysis and Design. It is meant to describe the interface between the Requirements and the Analysis and Design discipline.

This Requirements Overview module will provide enough information to give you an appreciation for the Requirements discipline, and enable you to read and interpret the Requirements artifacts that serve as the starting point for the Analysis and Design activities.

Introduction



Requirements Overview Topics

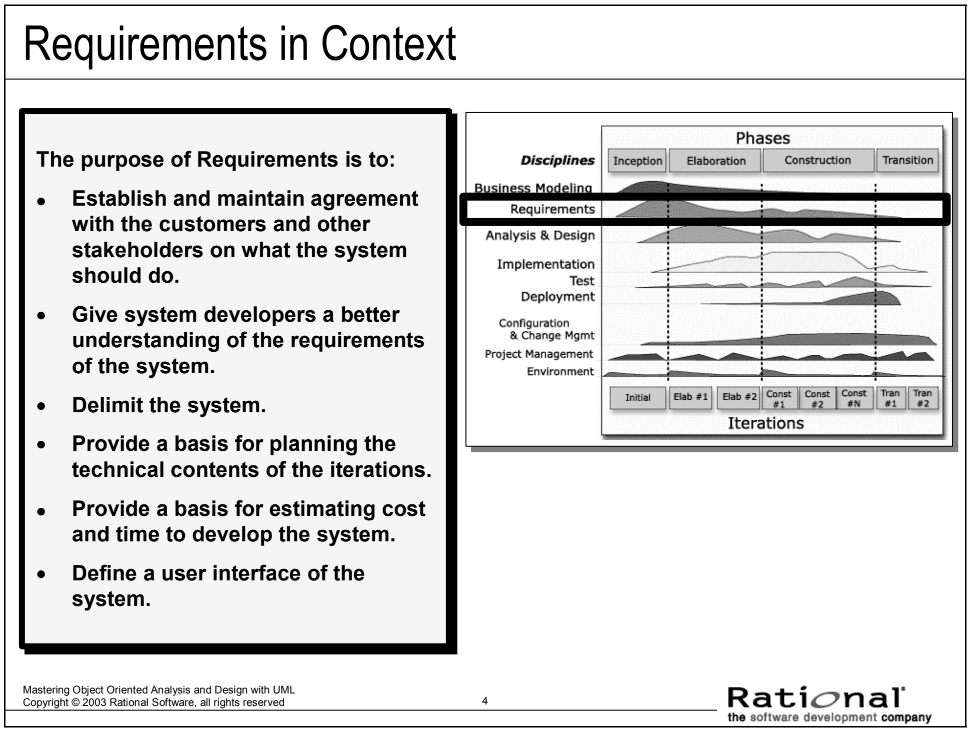
☆ ◆ Introduction
◆ Key Concepts
◆ Use-Case Model
◆ Glossary
◆ Supplementary Specifications
◆ Checkpoints

Mastering Object Oriented Analysis and Design with UML
Copyright © 2003 Rational Software, all rights reserved 3

Rational the software development company

We will start with an introduction to the Requirements discipline, followed by a review of the key concepts in use-case modeling. Then we will look briefly at each of the Requirements' artifacts and discuss how to read and interpret their contents. We will close by reviewing a series of checklists that will assist you in assessing the quality and completeness of the Requirements artifacts.

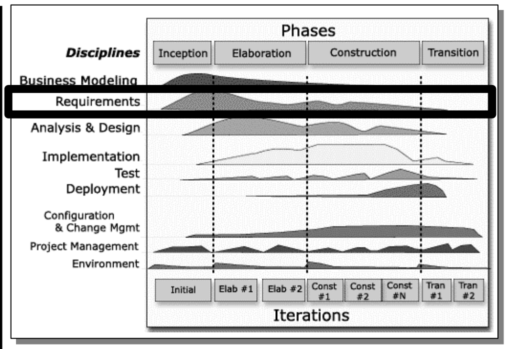
Requirements in Context



Requirements in Context

The purpose of Requirements is to:

- **Establish and maintain agreement with the customers and other stakeholders on what the system should do.**
- **Give system developers a better understanding of the requirements of the system.**
- **Delimit the system.**
- **Provide a basis for planning the technical contents of the iterations.**
- **Provide a basis for estimating cost and time to develop the system.**
- **Define a user interface of the system.**



Mastering Object Oriented Analysis and Design with UML
Copyright © 2003 Rational Software, all rights reserved 4

Rational the software development company

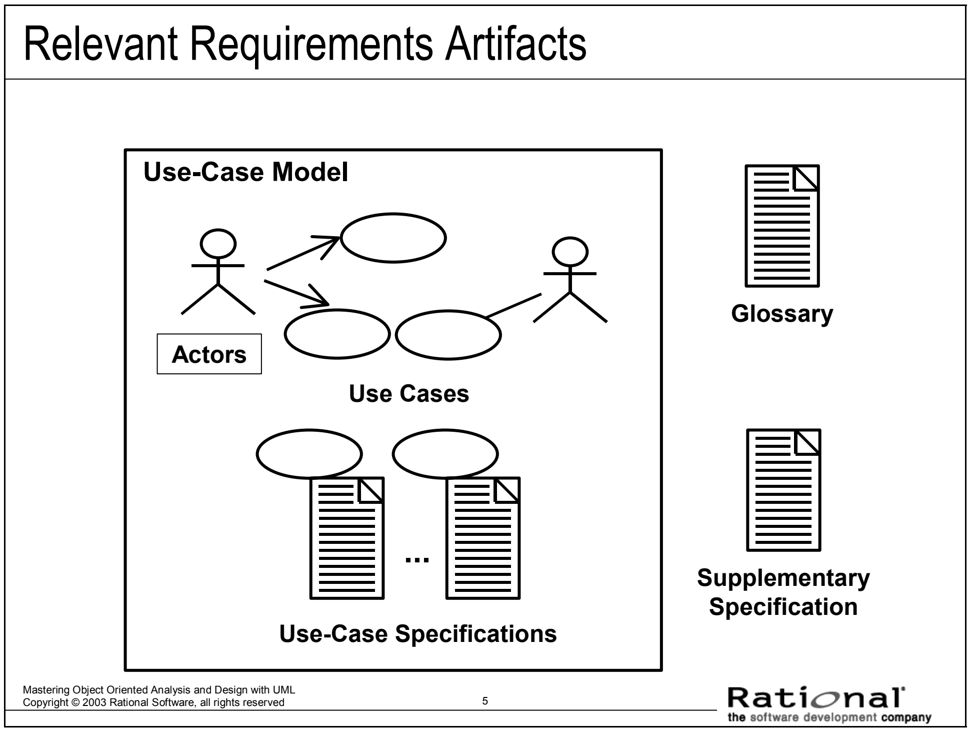
The Business Modeling discipline provides organizational context for the system. This is the context in which the requirements are defined and analyzed.

The purpose of the Requirements discipline is:

- To establish and maintain agreement with the customers and other stakeholders on what the system should do.
- To provide system developers with a better understanding of the system requirements.
- To define the boundaries of (delimit) the system.
- To provide a basis for planning the technical contents of iterations.
- To provide a basis for estimating cost and time to develop the system.
- To define a user-interface for the system, focusing on the needs and goals of the users.

The Analysis and Design discipline gets its primary input (the Use-Case Model and the Glossary) from Requirements. Flaws in the Use-Case Model can be discovered during Analysis and Design; change requests are then generated, and applied to the Use-Case Model.

Relevant Requirements Artifacts


Relevant Requirements Artifacts

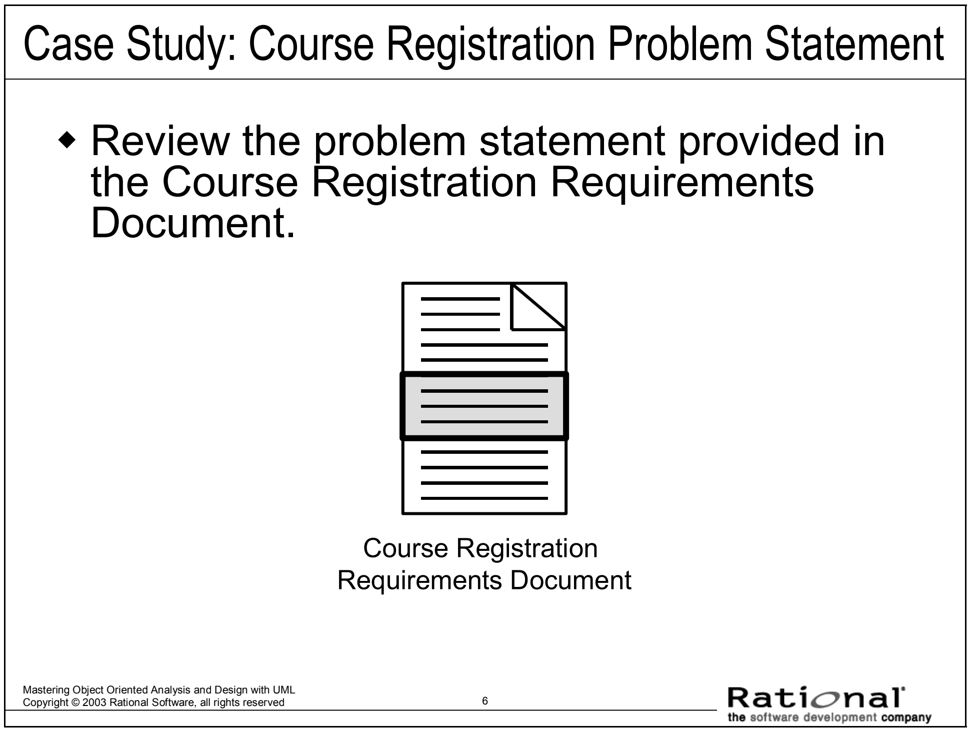
The **Use-Case Model** describes what the system will do. The Use-Case Model serves as a contract between the customer, the users, and the system developers. It allows customers and users to validate that the system will become what they expected and allows system developers to ensure that what they build is what is expected. The Use-Case Model consists of use cases and actors. Each use case in the model is described in detail, showing step-by-step how the system interacts with the actors and what the system does in the use case. The Use-Case Specification is the document where all of the use-case properties are documented (for example, brief description and use-case flows of events).

Note: The OOAD course requirements documentation includes Use-Case Specifications because it is the textual description that will drive Analysis and Design activities. (Use-case specifications only include the textual use-case properties.)

The **Glossary** defines a common terminology for all models and contains textual descriptions of the required system.

The **Supplementary Specification** contains those requirements that do not map to a specific use case (for example, nonfunctional requirements). The Supplementary Specification is an important complement to the Use-Case Model. Together they capture all requirements (functional and nonfunctional) that need to be described for a complete System Requirements Specification.

Case Study: Course Registration Problem Statement



Case Study: Course Registration Problem Statement

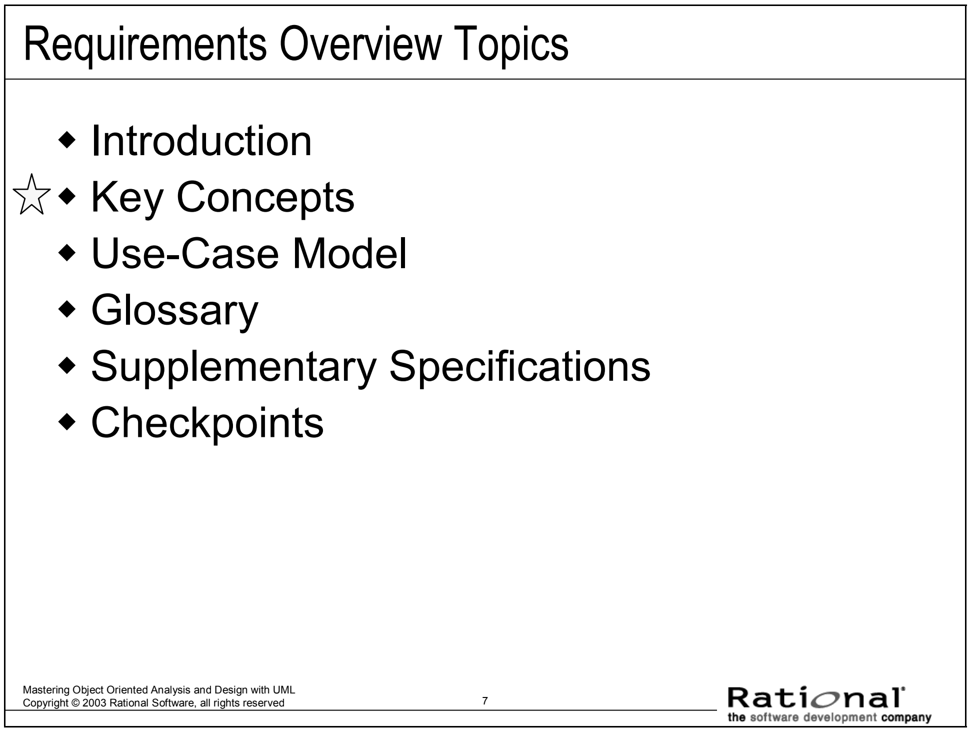
- Review the problem statement provided in the Course Registration Requirements Document.

Course Registration
Requirements Document

Before we discuss the details of the artifacts that drive the Analysis and Design discipline, it is important that you understand the problem domain that all of the course exercises will be based on.

With regards to the formal Requirements artifacts, the Problem Statement is part of the Vision document. The other sections have been omitted for scoping reasons. For more information on the Vision document, see the Requirements discipline of the Rational Unified Process.

Key Concepts

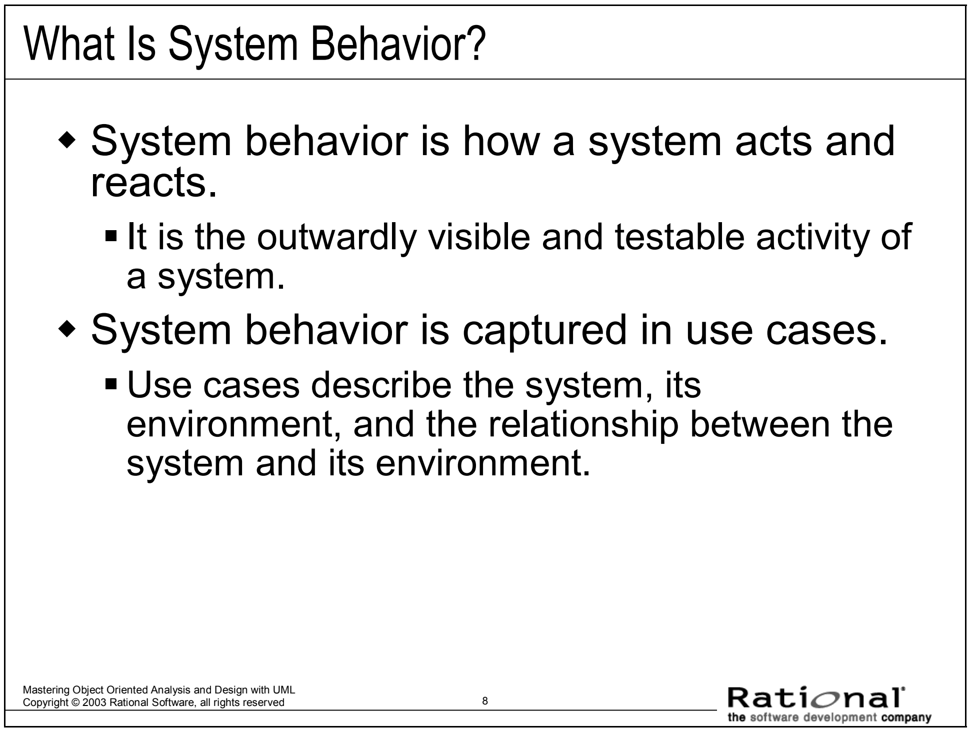


Requirements Overview Topics

- Introduction
- ☆ Key Concepts
- Use-Case Model
- Glossary
- Supplementary Specifications
- Checkpoints

In this section, we will discuss the key concepts of use-case modeling — the actor and the use case, as well as what relationships can exist between them. We will also look at the artifacts that make up the Use-Case Model: Use-Case Specifications, and activity diagrams.

What Is System Behavior?

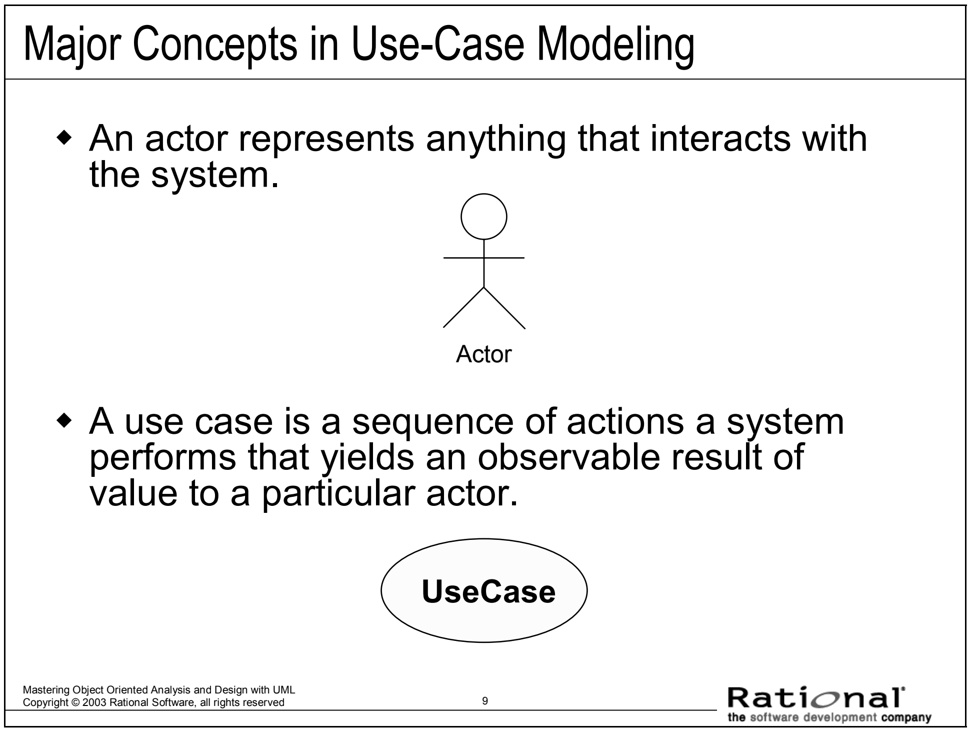


What Is System Behavior?

- System behavior is how a system acts and reacts.
  - It is the outwardly visible and testable activity of a system.
- System behavior is captured in use cases.
  - Use cases describe the system, its environment, and the relationship between the system and its environment.

- No system exists in isolation. Every system interacts with people or automated systems for some purpose. These interactions result in some sort of predictable result. This predictable result is system behavior.
- Use cases are the mechanism for capturing the desired behavior for the system that is under development, but they do not specify how the behavior is to be implemented.
- The UML specifies a model for communicating system behavior — the Use-Case Model.

Major Concepts in Use-Case Modeling



Major Concepts in Use-Case Modeling

- An actor represents anything that interacts with the system.

Actor

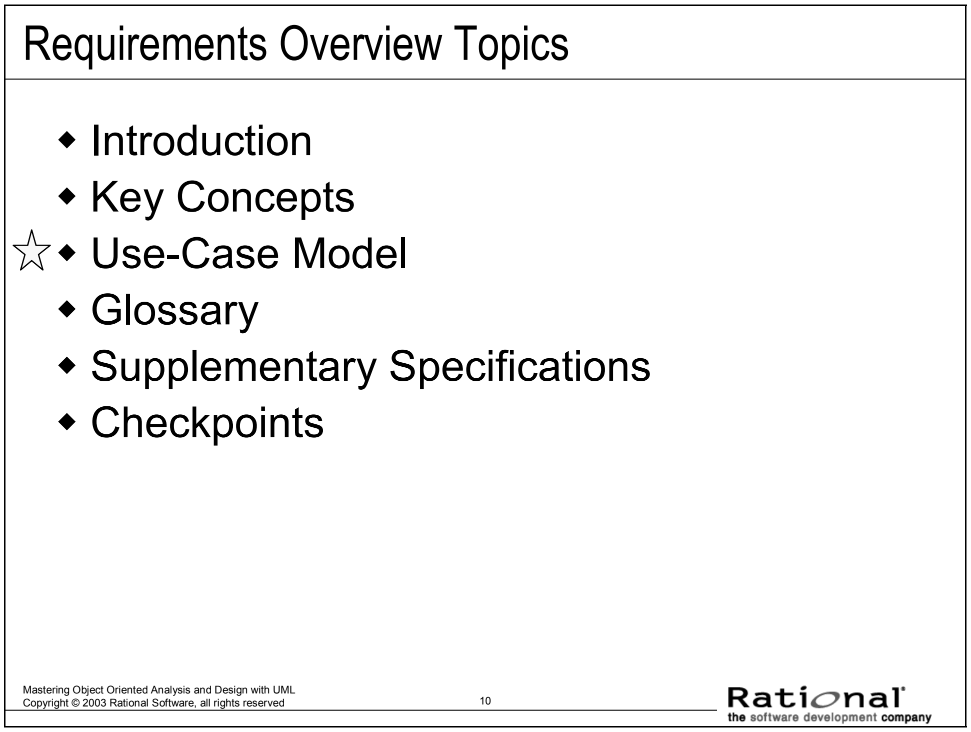
- A use case is a sequence of actions a system performs that yields an observable result of value to a particular actor.

UseCase

An **actor** represents a coherent set of roles that users of the system play when interacting with these use cases. Typically, an actor represents a role that a human, a hardware device, or even another system plays with a system.

A **use case** is a sequence of actions a system performs to yield an observable result that is of value to a particular actor. A use case describes *what* a system does, but it does not specify *how* it does it.

Use-Case Model



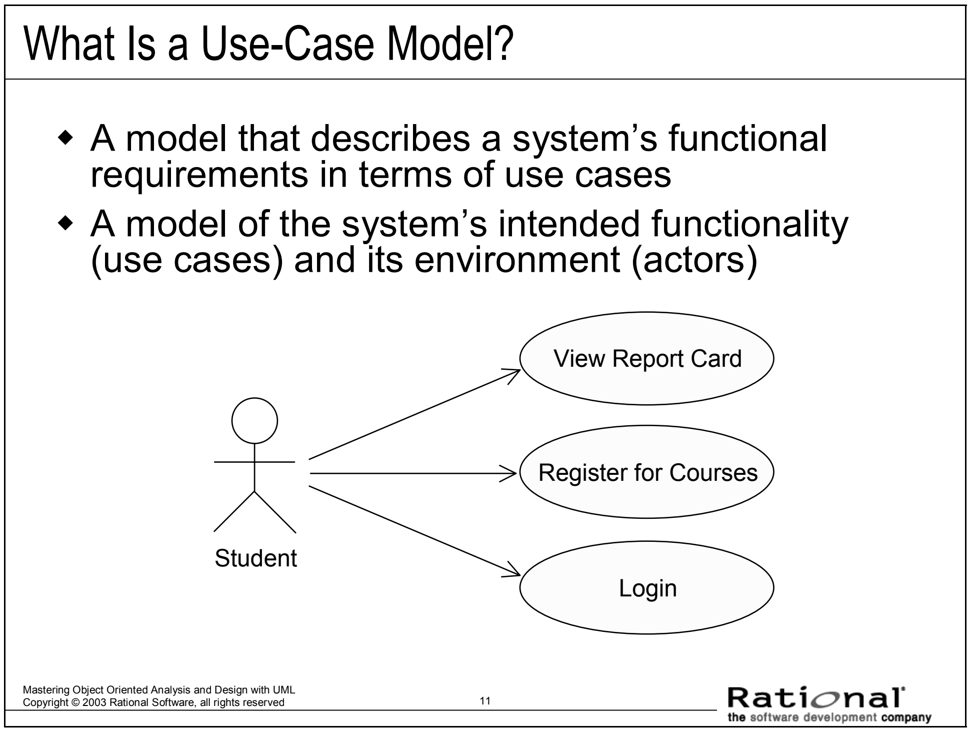
Requirements Overview Topics

- Introduction
- Key Concepts
- ☆ Use-Case Model
- Glossary
- Supplementary Specifications
- Checkpoints

10

Rational®
the software development company

What Is a Use-Case Model?



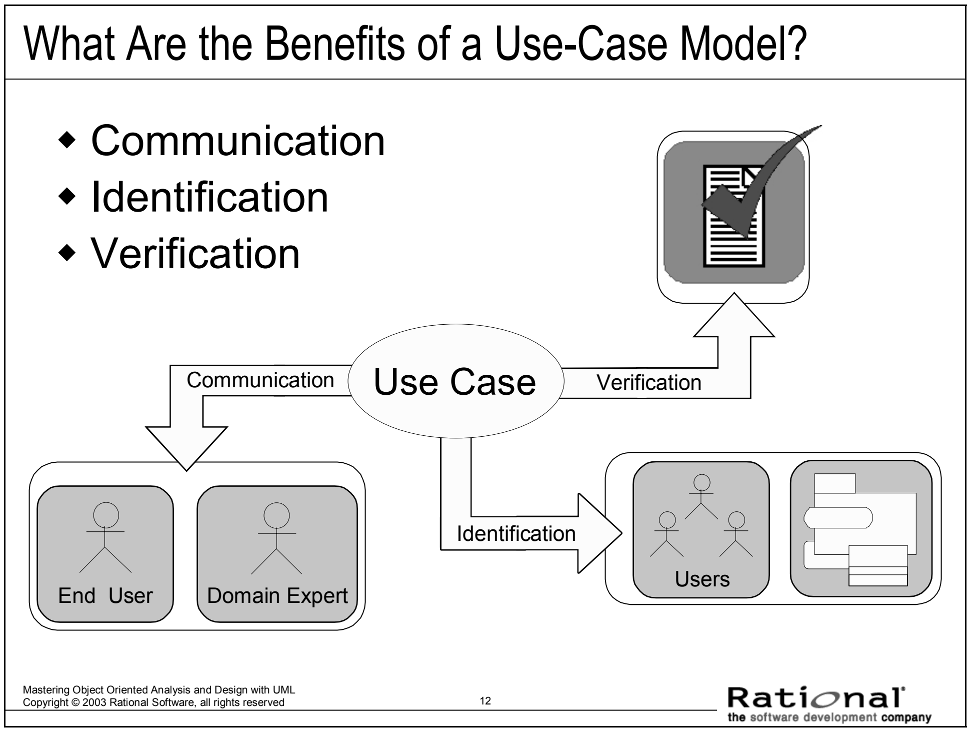
What Is a Use-Case Model?

- A model that describes a system's functional requirements in terms of use cases
- A model of the system's intended functionality (use cases) and its environment (actors)

View Report Card
Register for Courses
Login
Student

- A **Use-Case Model** describes a system's functional requirements in terms of use cases. It is a model of the system's intended functionality and its environment. The Use-Case Model serves as a contract between the customer and the developers. Because it is a very powerful planning instrument, the Use-Case Model is generally used in all phases of the development cycle.
- When the customer approves the Use-Case Model, you know the system is what the customer wants. You can also use the model to discuss the system with the customer during development.
- Potential users use the Use-Case Model to better understand the system.
- Designers use it as a basis for their work and to get a system overview.
- Testers use it to plan testing activities (use case and integration testing) as early as possible.
- Those developing the next version of the system use it to understand how the existing version works.
- Documentation writers use the use cases as a basis for writing the system user guides.
- The architect uses the Use-Case Model to identify architecturally significant functionality.
- The manager uses it to plan and follow up on use-case modeling and subsequent design.

What Are the Benefits of a Use-Case Model?



What Are the Benefits of a Use-Case Model?

- Communication
- Identification
- Verification

There are many ways to model a system, each of which may serve a different purpose. However, the most important role of a Use-Case Model is to communicate the system's behavior to the customer or end user. Consequently, the model must be easy to understand.

Communication with the end users and domain experts:

- Provides buy-in at an early stage of system development.
- Ensures a mutual understanding of the requirements.

Identification of system users and what the system should do:

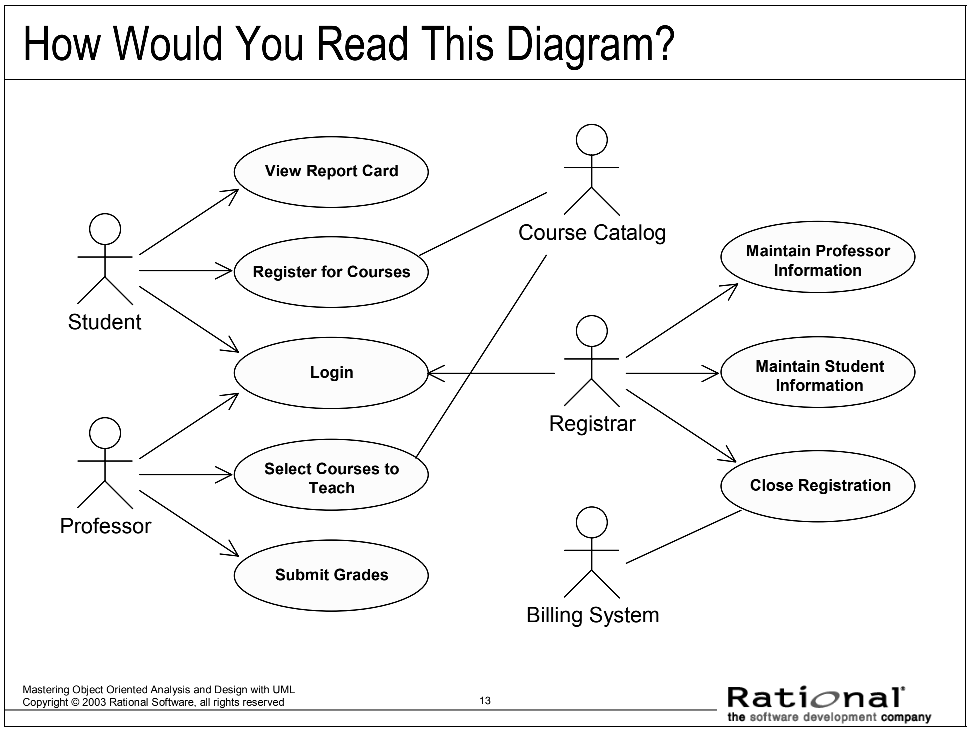
- Establishes the requirements for the system interfaces.

Verification that all requirements have been captured:

- Ensures that the development team understands the requirements.

The Use-Case Model is also used to identify the **actors** that interact with the system. Because they represent system users, actors help delimit the system and give a clearer picture of what it is supposed to do. Use cases are developed on the basis of the actors' needs, ensuring that the system will turn out to be what the users expected.

How Would You Read This Diagram?

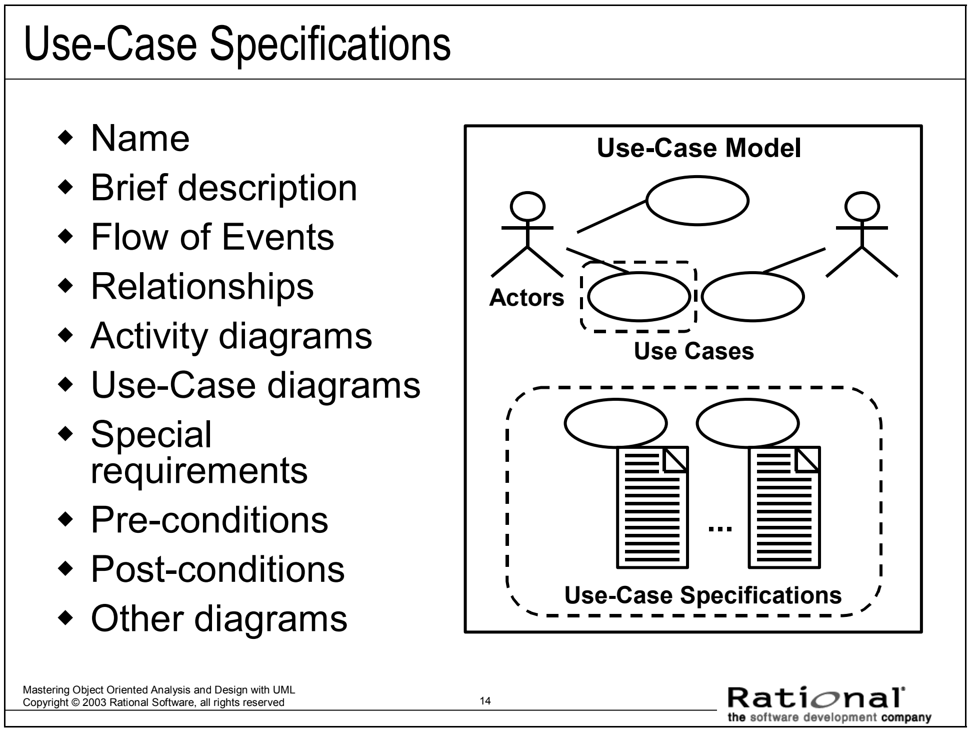


How Would You Read This Diagram?

Answer the following questions:

1. Which use cases will a student be able to perform? A professor? The Course Catalog?
2. If Charlie is a student and professor, which use cases will he be able to execute?
3. Describe the functionality of this system.
4. Describe the actor relationships for the Close Registration and Select Courses To Teach use cases.
5. What doesn't this model say?
6. Which use case needs to run first — Register for Courses or View Report Card?

Use-Case Specifications



Use-Case Specifications

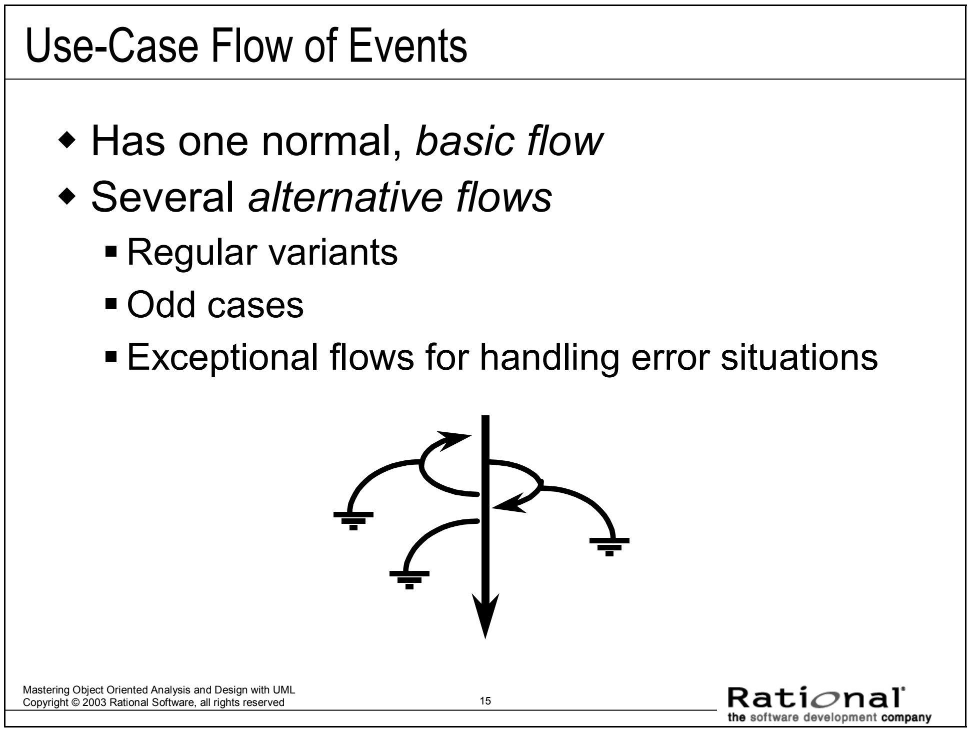
- Name
- Brief description
- Flow of Events
- Relationships
- Activity diagrams
- Use-Case diagrams
- Special requirements
- Pre-conditions
- Post-conditions
- Other diagrams

Use-Case Model
Actors
Use Cases
...
Use-Case Specifications

The use case has a set of properties as shown in the graphic. The use-case properties may be documented in use-case specifications, which can include the items listed below:

- **Brief description** describes the role and purpose of the use case.
- **Flow of events** are textual descriptions of what the system does with regard to the use case. There can be multiple flows of events — for example, a basic flow and alternative flows.
- **Relationships** are communicates-associations. The use case includes and extends relationships that the use case participates in.
- **Activity diagrams** can be used to illustrate the structure of the flow of events.
- **Use-case diagrams** can be used to show the relationships involving the use case.
- **Special requirements** is a textual description that collects all use-case requirements, like nonfunctional requirements, that are not considered in the Use-Case Model, yet need to be taken care of during design or implementation.
- **Pre-conditions** define a constraint on the system regarding when the use case may start.
- **Post-conditions** define a constraint on the system that applies after the use case has terminated.
- **Other diagrams** can be used to illustrate the use case, like hand-drawn sketches or screen captures from an user-interface prototype.

Use-Case Flow of Events



Use-Case Flow of Events

- Has one normal, *basic flow*
- Several *alternative flows*
  - Regular variants
  - Odd cases
  - Exceptional flows for handling error situations

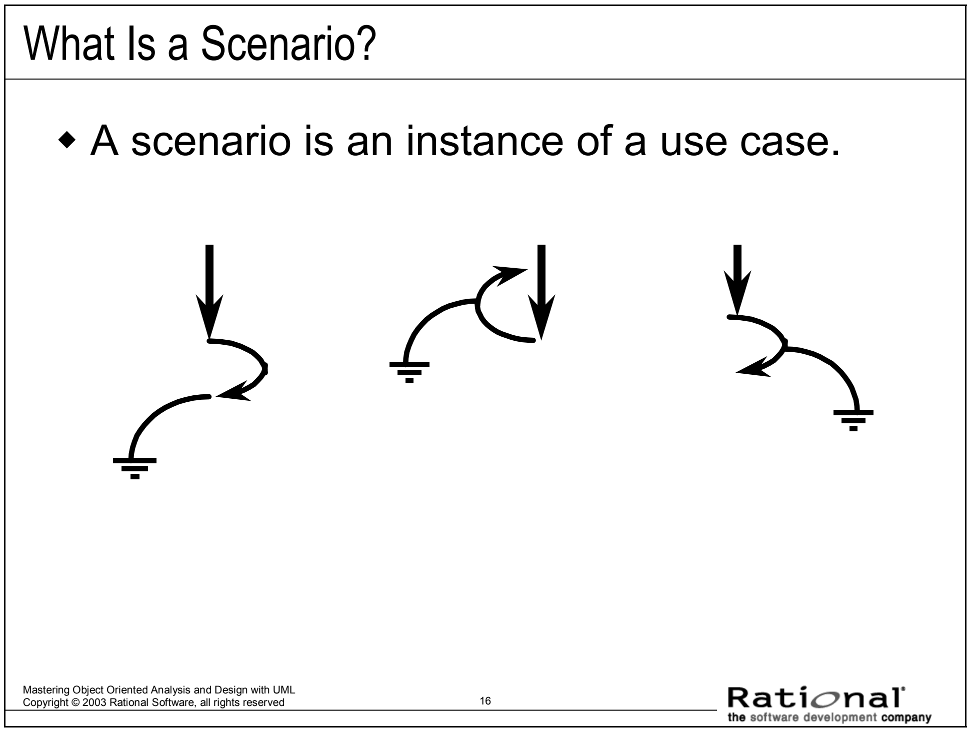
A use case **flow of events**:

- Contains the most important information derived from use-case modeling work.
- Should describe the use case's flow clearly enough for an outsider to easily understand it.
- Should present **what** the system does, not how the system is designed to perform the required behavior.

Guidelines for the flow of events. Specify that the content must:

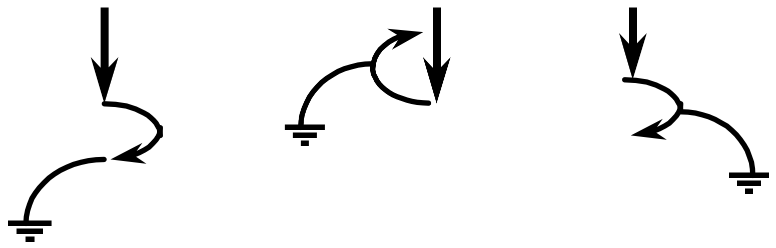
- Detail the flow of events. All "what" questions should be answered. Remember that test designers will use this text to identify test cases.
- Describe how the use case starts and ends.
- Describe the flow of events, not only the functionality. To reinforce this, start every action with "When the actor. . . ."
- Describe only the events that belong to the use case and not what happens in other use cases or outside of the system.
- Describe the data exchanged between the actor and the use case.
- Avoid describing the details of the user interface unless they are needed to provide an understanding the behavior of the system.
- Avoid vague terminology such as "for example", "etc.," and "information."

What Is a Scenario?



What Is a Scenario?

- A scenario is an instance of a use case.



A scenario is an instance of a use case. It is one flow through a use case.

Each use case has a web of flow of events with a scenario being an instance of a particular flow of events. The scenario may involve the basic flow and any number of alternative flows in any number of combinations.

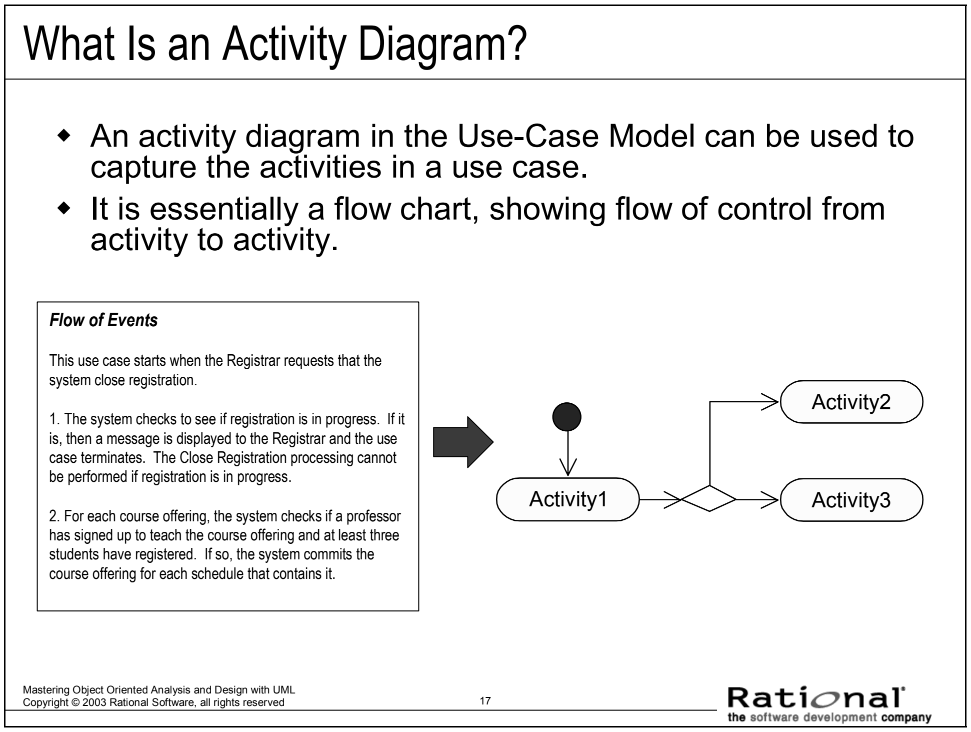
In the example, the bold lines highlight some possible scenarios for the basic and alternative flows previously described.

How many scenarios are needed?

As many as one needs to understand the system being developed. You must elaborate the scenarios of the interesting and high-risk use cases. Scenarios can be used to understand, as well as to validate, the use-case flows of events. Some people write scenarios first and extract use cases, while others find use cases first and validate those use cases by writing scenarios.

Scenarios make excellent test cases.

What Is an Activity Diagram?



What Is an Activity Diagram?

- An activity diagram in the Use-Case Model can be used to capture the activities in a use case.
- It is essentially a flow chart, showing flow of control from activity to activity.

Flow of Events

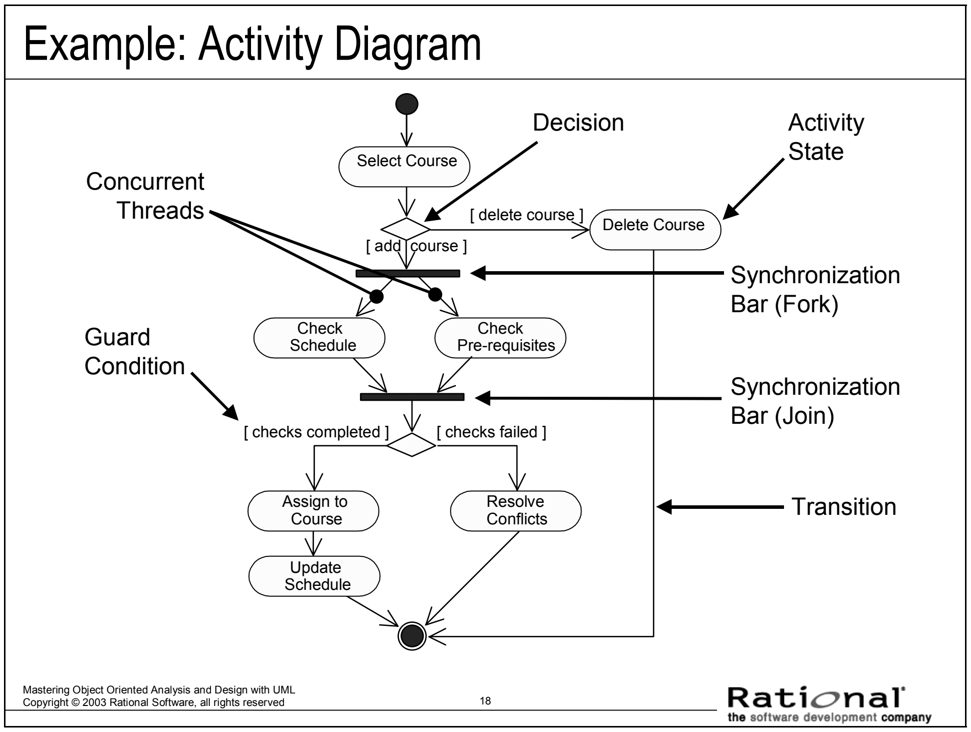
This use case starts when the Registrar requests that the system close registration.

1. The system checks to see if registration is in progress. If it is, then a message is displayed to the Registrar and the use case terminates. The Close Registration processing cannot be performed if registration is in progress.

2. For each course offering, the system checks if a professor has signed up to teach the course offering and at least three students have registered. If so, the system commits the course offering for each schedule that contains it.

- The workflow of a use case describes what needs to be done by the system to provide the value that the served actor is looking for.
- It consists of a sequence of activities that, together, produce something for the actor.
- The workflow often consists of a basic flow and one or several alternative flows.
- The structure of the workflow can be described graphically with the help of an activity diagram.

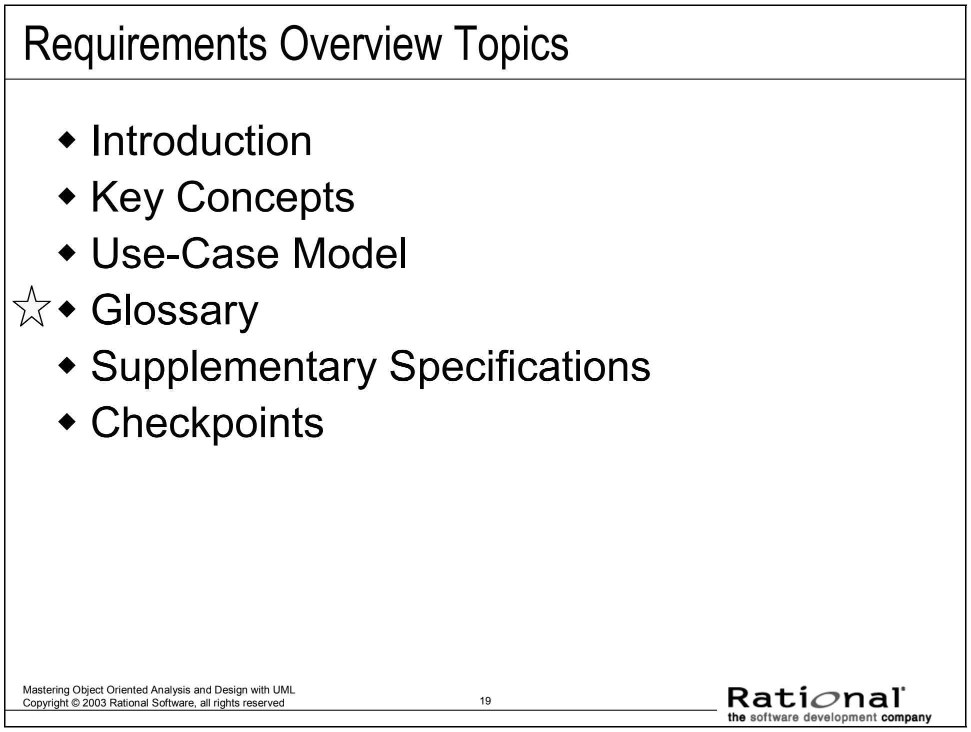
Example: Activity Diagram



An activity diagram may include the following elements:

- **Activity states** represent the performance of an activity or step within the workflow.
- **Transitions** show what activity state follows after another.
- **Decisions** evaluate conditions defined by guard conditions. These guard conditions determine which of the alternative transitions will be made and, thus, which activities are performed. You may also use the decision icon to show where the threads merge again. Decisions and guard conditions allow you to show alternative threads in the workflow of a use case.
- **Synchronization bars** show parallel sub-flows. They allow you to show concurrent threads in the workflow of a use case.

Glossary



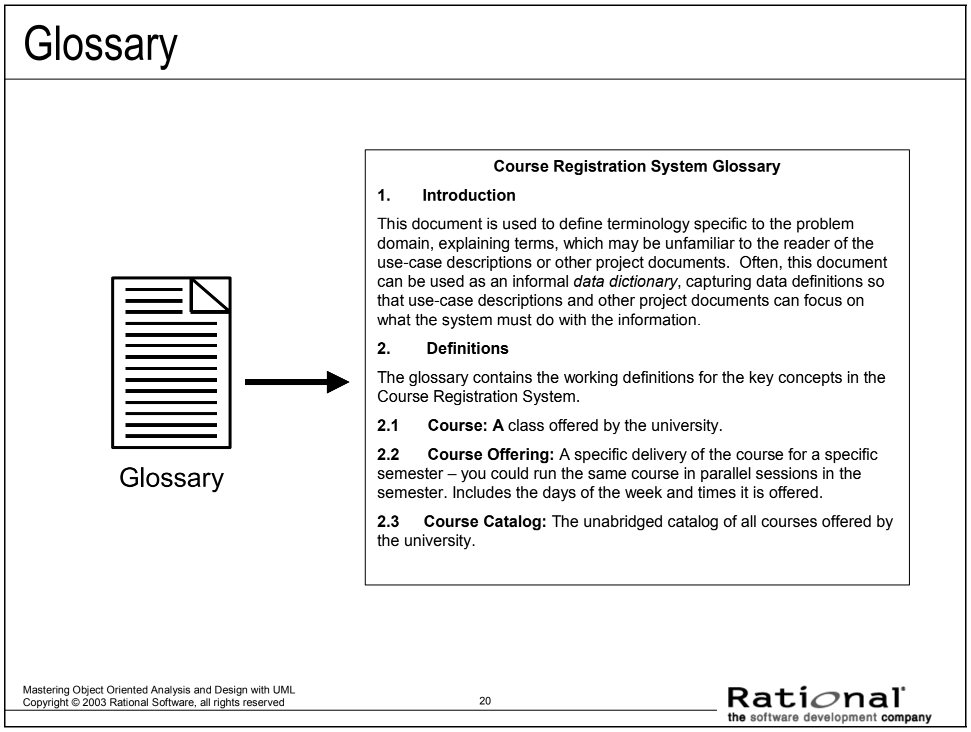
Requirements Overview Topics

- Introduction
- Key Concepts
- Use-Case Model
- ☆ Glossary
- Supplementary Specifications
- Checkpoints

19

Rational
the software development company

Glossary



Glossary

Course Registration System Glossary

1. Introduction

This document is used to define terminology specific to the problem domain, explaining terms, which may be unfamiliar to the reader of the use-case descriptions or other project documents. Often, this document can be used as an informal *data dictionary*, capturing data definitions so that use-case descriptions and other project documents can focus on what the system must do with the information.

2. Definitions

The glossary contains the working definitions for the key concepts in the Course Registration System.

2.1 Course: A class offered by the university.

2.2 Course Offering: A specific delivery of the course for a specific semester – you could run the same course in parallel sessions in the semester. Includes the days of the week and times it is offered.

2.3 Course Catalog: The unabridged catalog of all courses offered by the university.

The **Glossary** defines important terms used in the project.

There is one Glossary for the system. This document is important to many developers, especially when they need to understand and use the terms that are specific to the project. The Glossary is used to facilitate communications between domain experts and developers.

The Glossary is developed primarily during the Inception and Elaboration phases, because it is important to agree on a common terminology early in the project. In Inception and Elaboration, it is used by domain experts (for example, business analysts) to explain all the domain-specific terminology used in their use cases. In Elaboration and Construction, developers use the Glossary to explain technical terms used in the other four models.

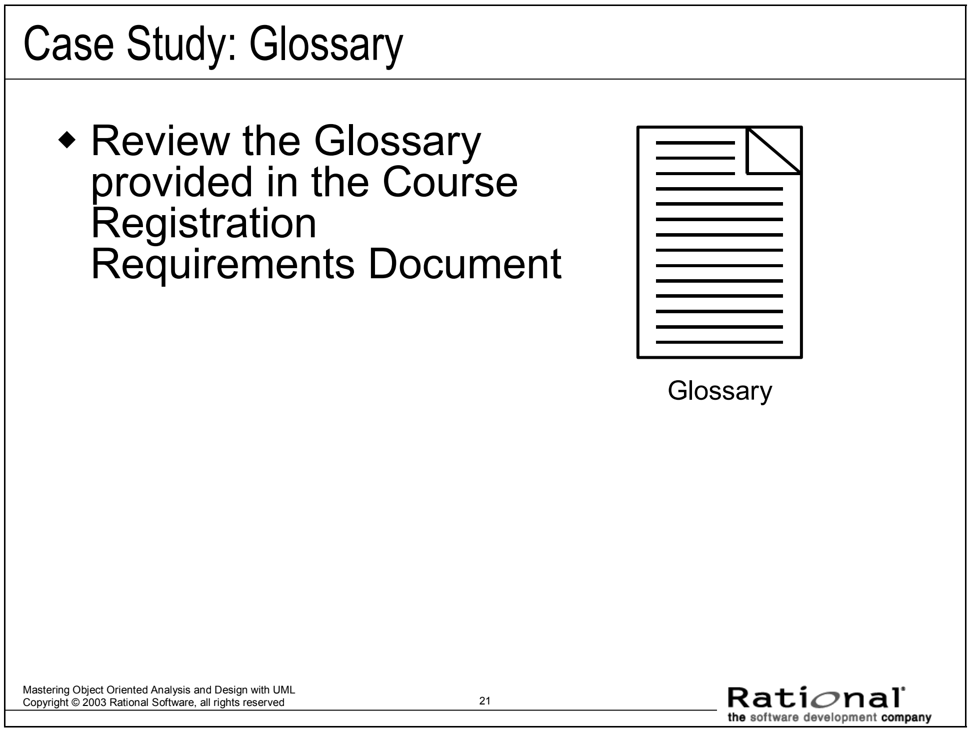
A system analyst is responsible for the integrity of the Glossary, ensuring that it is produced in a timely manner and is continuously kept consistent with the results of development.

The above is just a sample outline for the Glossary. Not all of these elements need to be in it. A project needs to establish the template to be used on that particular project.

Introduction: Provides a brief description of the Glossary and its purpose.

Terms: Define the term in as much detail as necessary to completely and unambiguously characterize it.

Case Study: Glossary

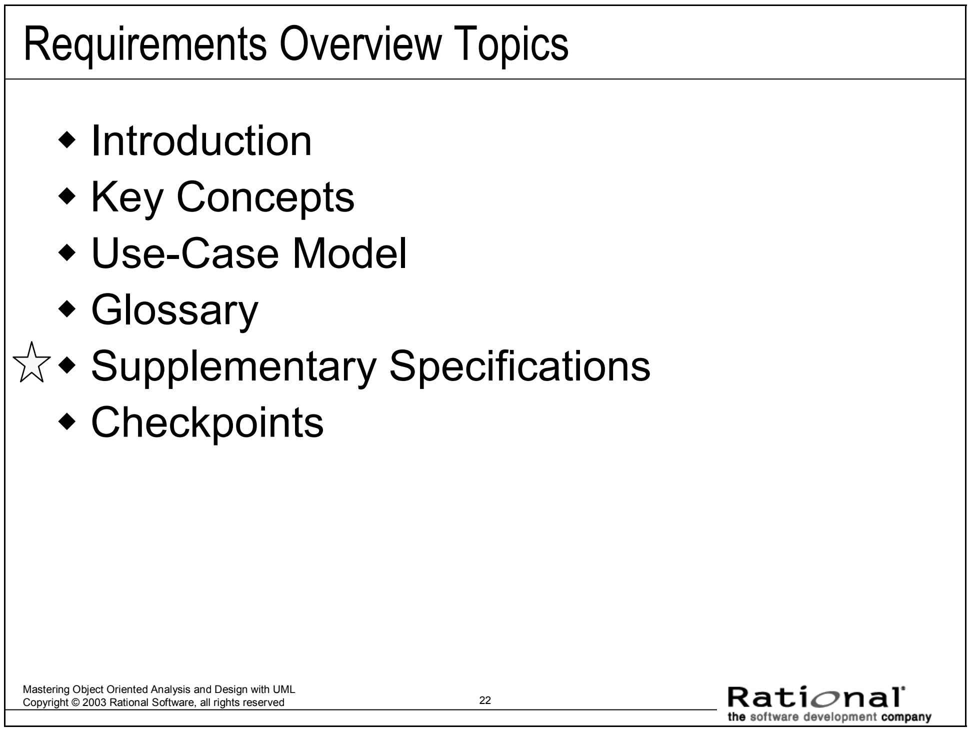


Case Study: Glossary

- Review the Glossary provided in the Course Registration Requirements Document

The idea is not to go over the Glossary in vivid detail, but to demonstrate how to read it, where to look for information you will need during the Analysis and Design activities, as well as how to detect if it is insufficient.

Supplementary Specifications



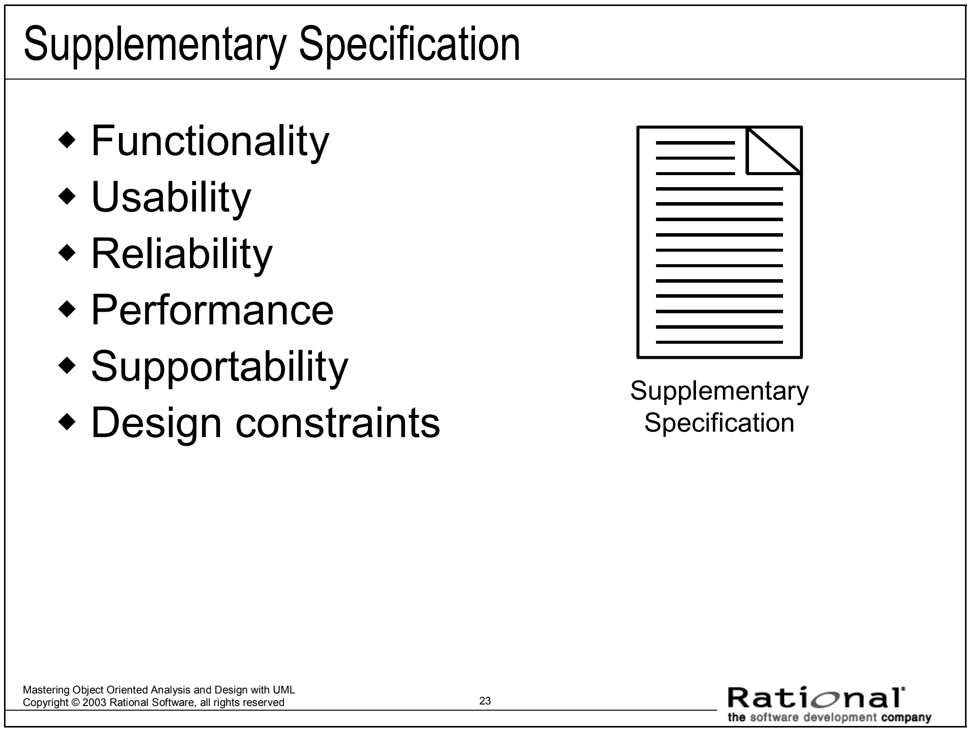
Requirements Overview Topics

- Introduction
- Key Concepts
- Use-Case Model
- Glossary
- ☆ Supplementary Specifications
- Checkpoints

Mastering Object Oriented Analysis and Design with UML
Copyright © 2003 Rational Software, all rights reserved 22

Rational the software development company

Supplementary Specification



Supplementary Specification

- Functionality
- Usability
- Reliability
- Performance
- Supportability
- Design constraints

Supplementary Specification

The nonfunctional requirements and functional requirements not captured by the use cases are included in the Supplementary Specifications. The Supplementary Specifications include constraints on the implementation.

Functionality: List of the functional requirements that are general to many use cases.

Usability: Requirements that relate to, or affect, the usability of the system. Examples include ease-of-use requirements or training requirements that specify how readily the system can be used by its actors.

Reliability: Any requirements concerning the reliability of the system. Quantitative measures such as mean time between failure or defects per thousand lines of code should be stated.

Performance: The performance characteristics of the system. Include specific response times. Reference related use cases by name.

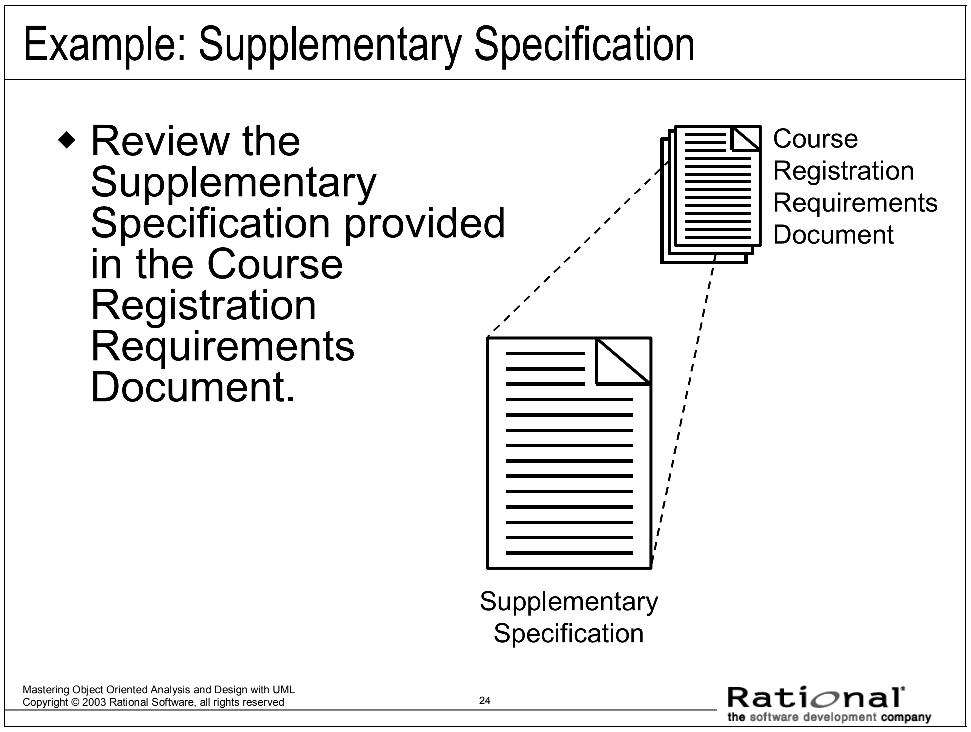
Supportability: Any requirements that will enhance the supportability or maintainability of the system being built.

Design Constraints: Any design constraints on the system being built.

Supplementary Specifications go hand-in-hand with the Use-Case Model, implying that:

- They are initially considered in the Inception phase as a complement to defining the scope of the system.
- They are refined in an incremental fashion during the Elaboration and Construction phases.

Example: Supplementary Specification



Example: Supplementary Specification

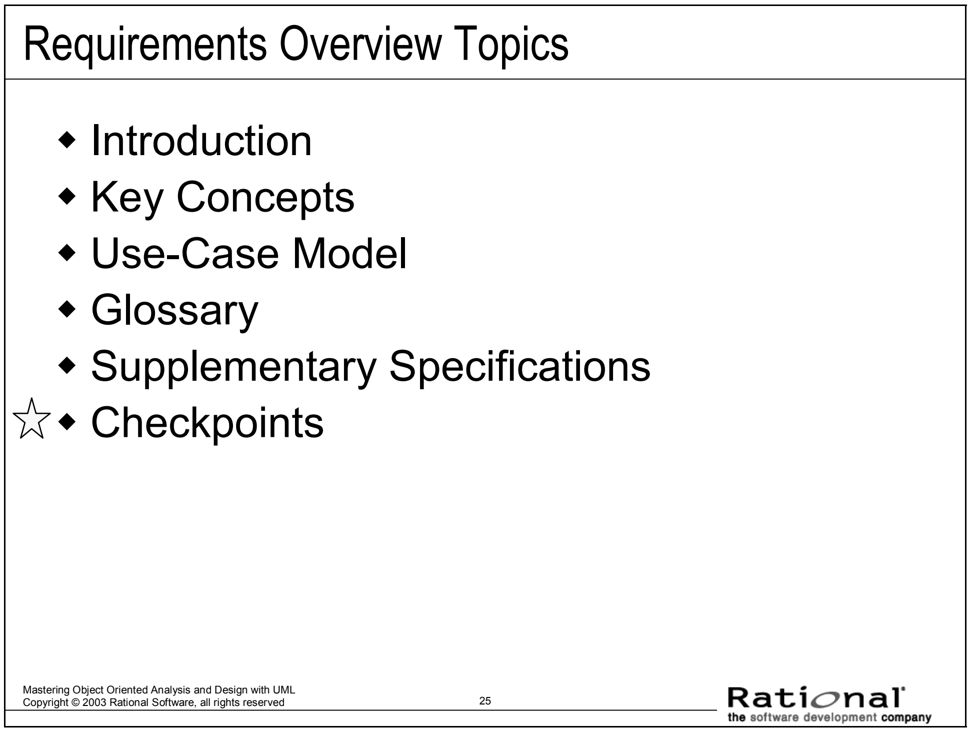
- Review the Supplementary Specification provided in the Course Registration Requirements Document.

Course Registration Requirements Document
Supplementary Specification

Mastering Object Oriented Analysis and Design with UML
Copyright © 2003 Rational Software, all rights reserved 24

The idea is not to go over the Supplementary Specification in vivid detail, but to demonstrate how to read it, where to look for information you will need during the Analysis and Design activities, as well as how to detect if it is insufficient.

Requirements Overview Topics



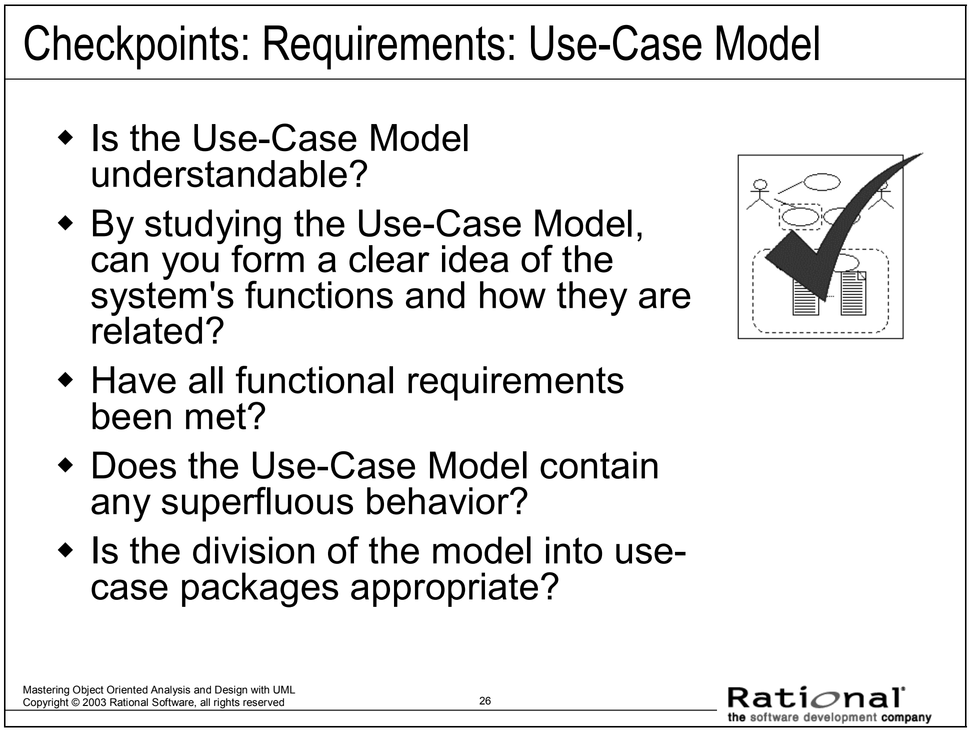
Requirements Overview Topics

- Introduction
- Key Concepts
- Use-Case Model
- Glossary
- Supplementary Specifications
- ☆ Checkpoints

Mastering Object Oriented Analysis and Design with UML
Copyright © 2003 Rational Software, all rights reserved 25

Rational
the software development company

Checkpoints: Requirements: Use-Case Model



Checkpoints: Requirements: Use-Case Model

- Is the Use-Case Model understandable?
- By studying the Use-Case Model, can you form a clear idea of the system's functions and how they are related?
- Have all functional requirements been met?
- Does the Use-Case Model contain any superfluous behavior?
- Is the division of the model into use-case packages appropriate?

Mastering Object Oriented Analysis and Design with UML
Copyright © 2003 Rational Software, all rights reserved 26

Rational the software development company

The above, as well as the remaining checklists in this section, is a sample of the kinds of things to look for when reviewing the Use-Case Model. Explicit, detailed checklists should be established for the project to support the review of the Use-Case Model.

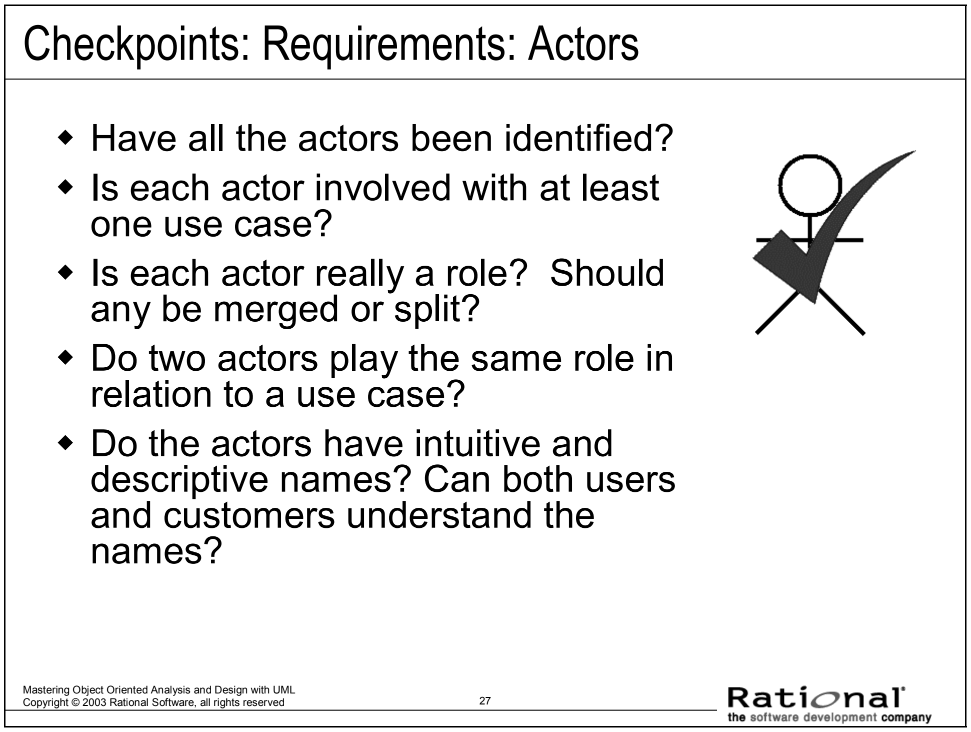
Verify that there are no open questions. The use cases you found must be able to perform all system behaviors; if not, some use cases are missing. If you intentionally left any requirements to be dealt with in the object models, such as nonfunctional requirements, you must mention this. Put the reference in the Supplementary Specification(s) unless the requirement concerns a specific use case, in which case state it in the Special Requirements section of the use case.

The Use-Case Model should not present more functions than were called for in the requirements.

Traceability from the original requirements to the use cases and the Supplementary Specification is critical for project management and impact assessment as well as to make sure nothing has been missed.

The packaging should make the Use-Case Model simple and intuitive to understand and maintain.

Checkpoints: Requirements: Actors



Checkpoints: Requirements: Actors

- Have all the actors been identified?
- Is each actor involved with at least one use case?
- Is each actor really a role? Should any be merged or split?
- Do two actors play the same role in relation to a use case?
- Do the actors have intuitive and descriptive names? Can both users and customers understand the names?

Mastering Object Oriented Analysis and Design with UML
Copyright © 2003 Rational Software, all rights reserved 27

Rational the software development company

Make sure that all the roles in the system's environment have been accounted for and modeled. You will not be able to do this fully until you have found and described all the use cases.

Remove any actors not mentioned in the use-case descriptions or that have no communicates-associations with a use case. However, an actor mentioned in a use-case description is likely to have a communicates-association with that particular use case.

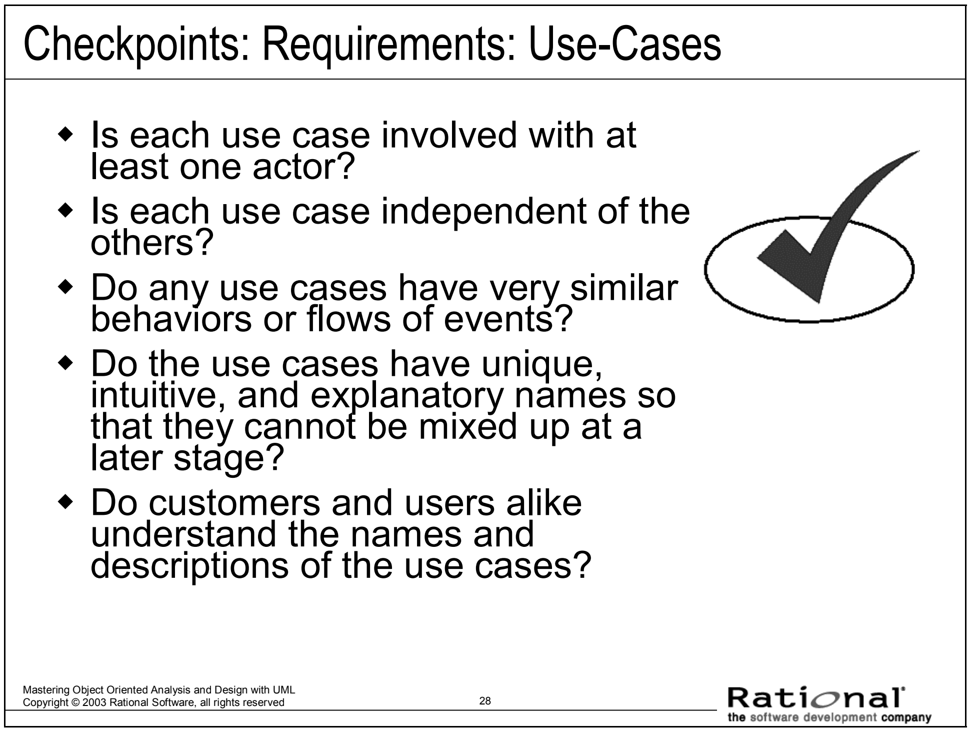
You should be able to name at least two people who would be able to perform as a particular actor. If not, see if the role the actor models is part of another role. If so, you should merge the actors.

If any actors play similar roles in relation to the system, they should be merged into a single actor. If a particular actor will use the system in several completely different ways, or has several completely different purposes for using the use case, then there should probably be more than one actor.

If two actors play the same role in relation to a use case, then actor-generalizations should be used to model their shared behavior.

It is important that actor names correspond to their roles.

Checkpoints: Requirements: Use-Cases



Checkpoints: Requirements: Use-Cases

- Is each use case involved with at least one actor?
- Is each use case independent of the others?
- Do any use cases have very similar behaviors or flows of events?
- Do the use cases have unique, intuitive, and explanatory names so that they cannot be mixed up at a later stage?
- Do customers and users alike understand the names and descriptions of the use cases?

Mastering Object Oriented Analysis and Design with UML
Copyright © 2003 Rational Software, all rights reserved 28

A use case that does not interact with an actor is superfluous. You should remove it.

If two use cases are always activated in the same sequence, you should probably merge them into one use case.

Use cases that have very similar behaviors or flows of events, or will be similar in the future, should be merged into a single use case. This makes it easier to introduce future changes.

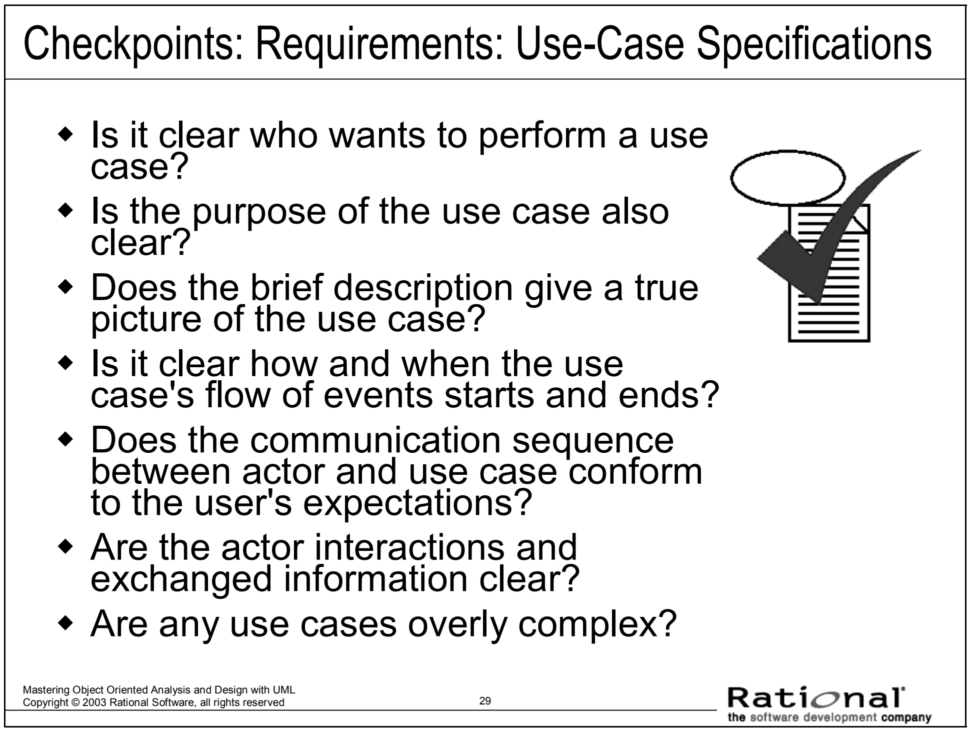
Note: You must involve the users if you decide to merge use cases because the users, who interact with the new, merged use case will probably be affected.

If some part of the flow of events is already part of another use case, the sub-flow should be extracted and then be used by all of the use cases in question.

Note: You must involve the users if you decide to "reuse" the sub-flow, because the users of the existing use case will probably be affected.

Each use-case name must describe the behavior the use case supports. The names and descriptions must be understood by the users and customers.

Checkpoints: Requirements: Use-Case Specifications



Checkpoints: Requirements: Use-Case Specifications

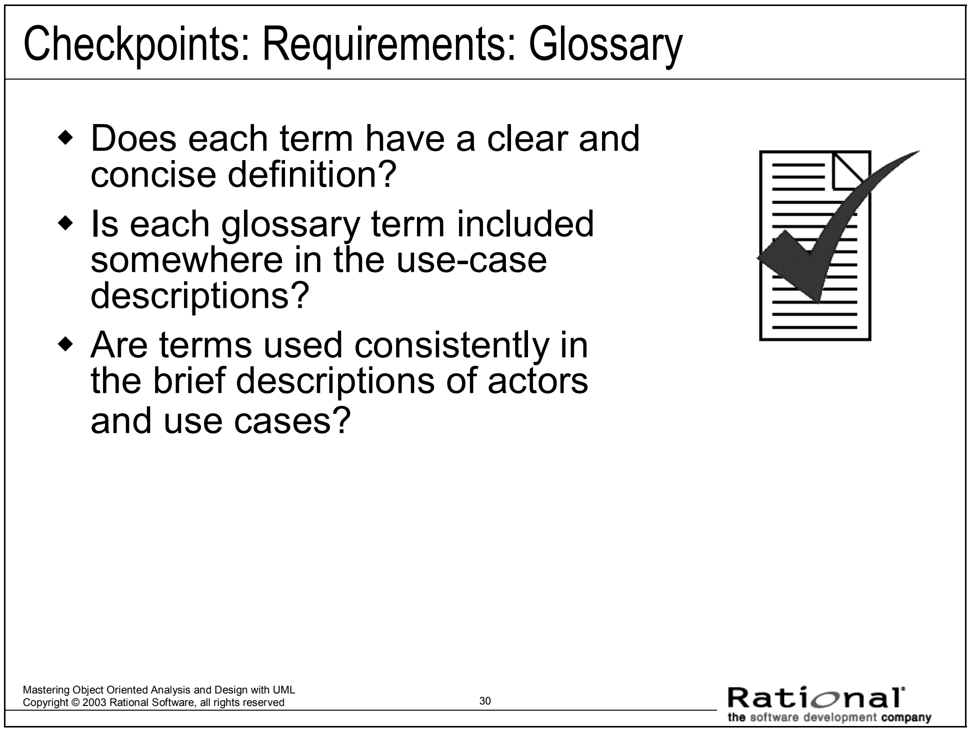
- Is it clear who wants to perform a use case?
- Is the purpose of the use case also clear?
- Does the brief description give a true picture of the use case?
- Is it clear how and when the use case's flow of events starts and ends?
- Does the communication sequence between actor and use case conform to the user's expectations?
- Are the actor interactions and exchanged information clear?
- Are any use cases overly complex?

Include any (nonfunctional) requirements to be handled in the object models in the use-case Special Requirements.

Behavior might exist that is activated only when a certain condition is not met. There should be a description of what will happen in such a case.

If you want your Use-Case Model to be easy to understand, you might have to split up complex use cases. A use case that contains disparate flows of events will be very difficult to understand and to maintain. It is best to divide such use cases into two or more separate ones.

Checkpoints: Requirements: Glossary



Checkpoints: Requirements: Glossary

- Does each term have a clear and concise definition?
- Is each glossary term included somewhere in the use-case descriptions?
- Are terms used consistently in the brief descriptions of actors and use cases?

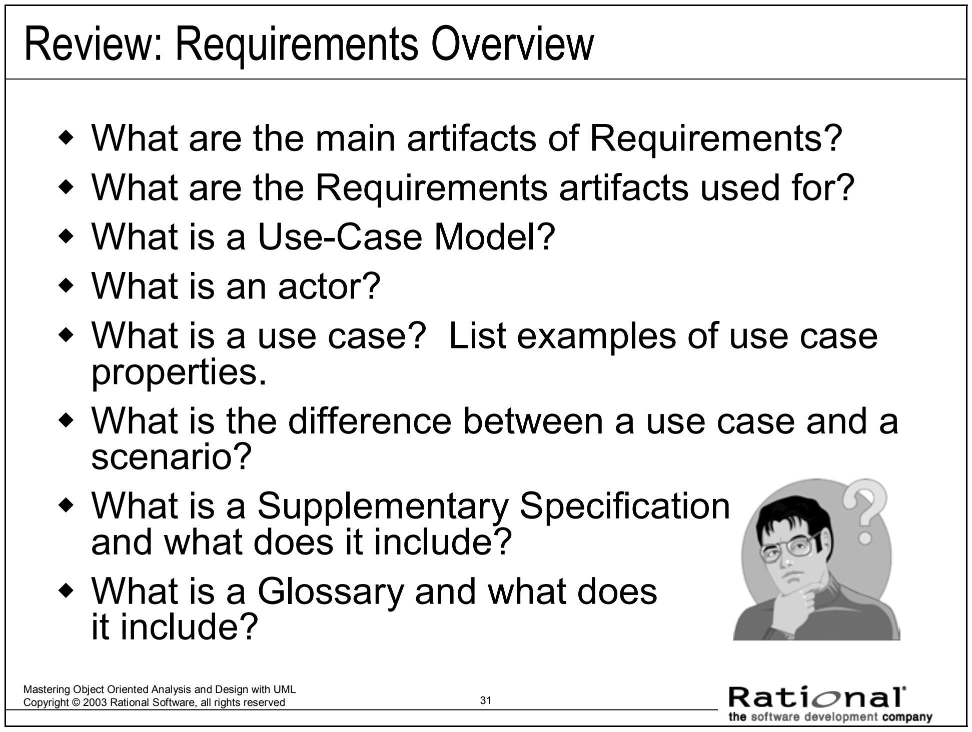
Mastering Object Oriented Analysis and Design with UML
Copyright © 2003 Rational Software, all rights reserved 30

Rational® the software development company

If each Glossary term is not included somewhere in the use-case descriptions, that may imply that a use case is missing or that the existing use cases are not complete. It is more likely, though, that the term is not included because it is not needed, and it should be removed from the Glossary.

A term should represent the same thing in all use-case descriptions.

Review



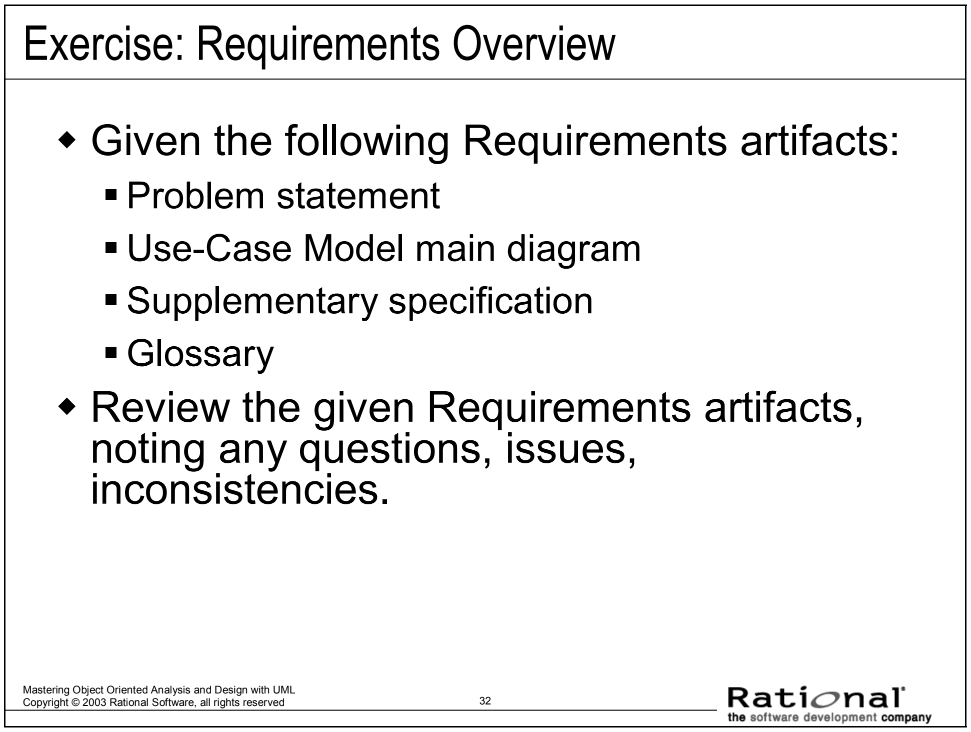
Review: Requirements Overview

- What are the main artifacts of Requirements?
- What are the Requirements artifacts used for?
- What is a Use-Case Model?
- What is an actor?
- What is a use case? List examples of use case properties.
- What is the difference between a use case and a scenario?
- What is a Supplementary Specification and what does it include?
- What is a Glossary and what does it include?

31

Rational the software development company

Exercise: Requirements Overview



Exercise: Requirements Overview

- Given the following Requirements artifacts:
  - Problem statement
  - Use-Case Model main diagram
  - Supplementary specification
  - Glossary
- Review the given Requirements artifacts, noting any questions, issues, inconsistencies.

32

Rational
the software development company

These artifacts will be used as the basis for the remainder of the examples and exercises in this course, so you need to have a good foundation for moving forward. All questions, issues, etc. regarding the Requirements artifacts will be recorded and addressed here.

You will not be reviewing the use-case flow of events at this point. They will be reviewed in detail later in the course.

The Requirements artifacts for this exercise can be found in the Payroll Requirements Document. See the document's table of contents for specific page numbers.