Object-Oriented Analysis and Design using JAVA

B.Tech (CSE/IT) 5th SEM
2021-2022

Lecture-8: Requirement elicitation

In previous lectures following topics regarding the software development process have been covered:

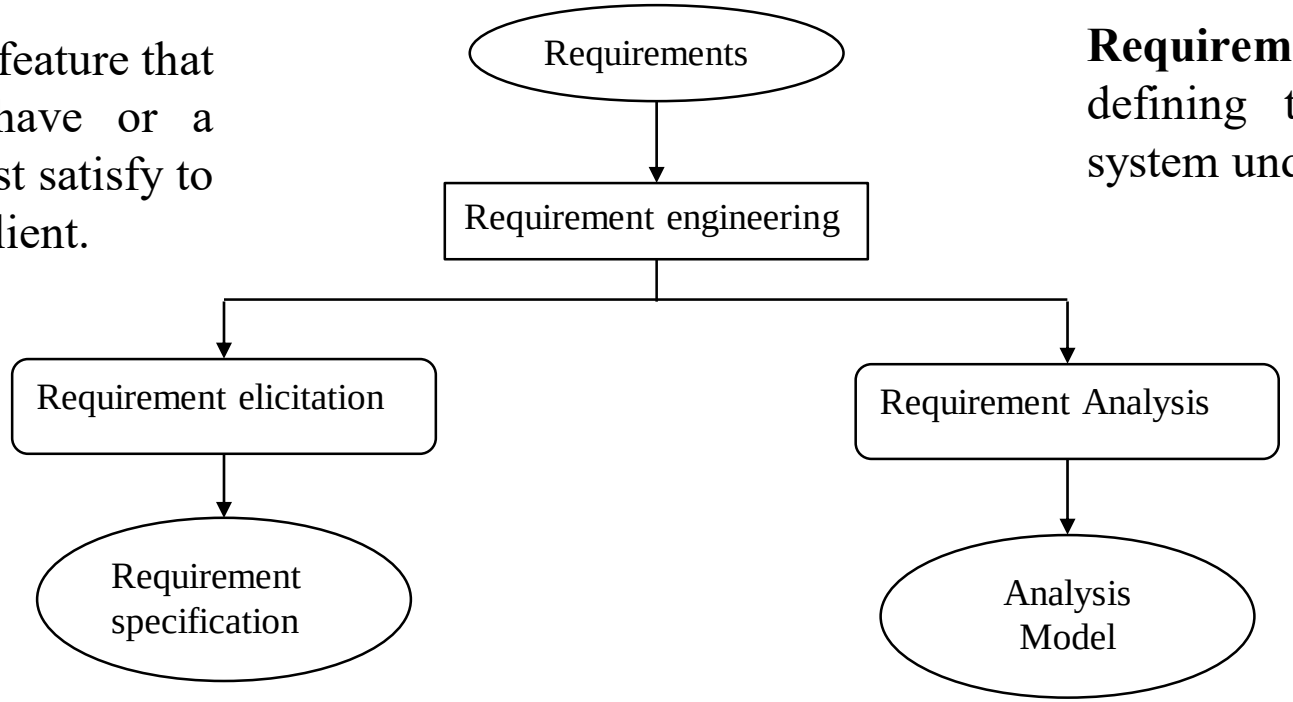
1. Introduction of software development life cycle (SDLC)-with their various phases-Requirement elicitation, Analysis, design, implementation, testing..
2. Description of waterfall model, V-model, spiral model and some brief descriptions about incremental, iterative models.
3. Introduction of object-oriented software development process-with their various phases-requirement elicitation, analysis, system design, object design, implementation, and testing.
4. Detailed explanation of Rational unified process (RUP) model with respect to the object-oriented methodology.

In this lecture following topic will be covered

Requirement elicitation

Introduction

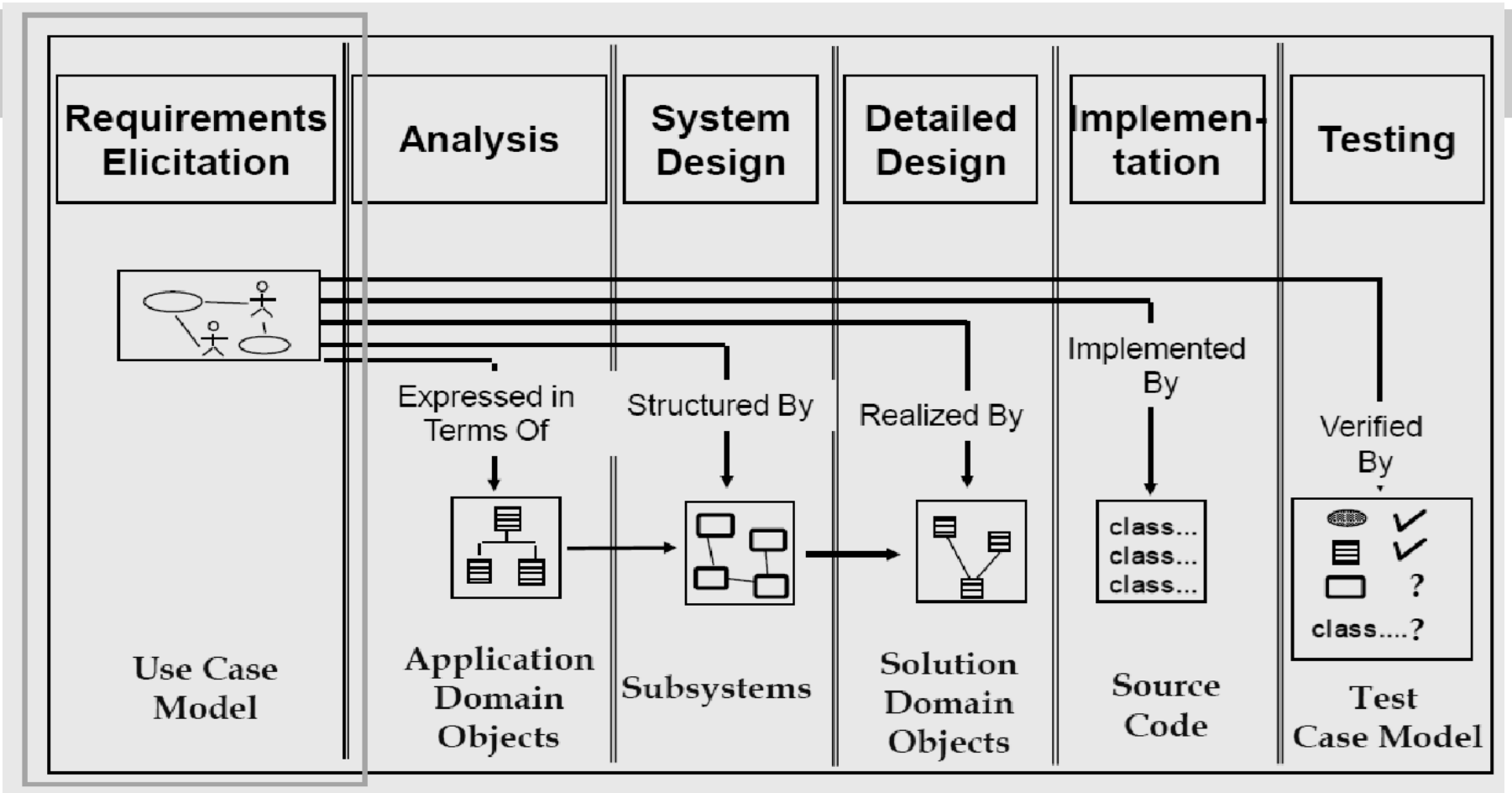
A requirement is a feature that the system must have or a constraint that it must satisfy to be accepted by the client.



Requirements engineering aims at defining the requirements of the system under construction

Specification of the system that the client understands

An analysis model that the developers can unambiguously interpret


Requirements Elicitation
Analysis
System Design
Detailed Design
Implemen-tation
Testing
Expressed in Terms Of
Structured By
Realized By
Implemented By
Verified By
class...
class...
class...
class....?
Use Case Model
Application Domain Objects
Subsystems
Solution Domain Objects
Source Code
Test Case Model

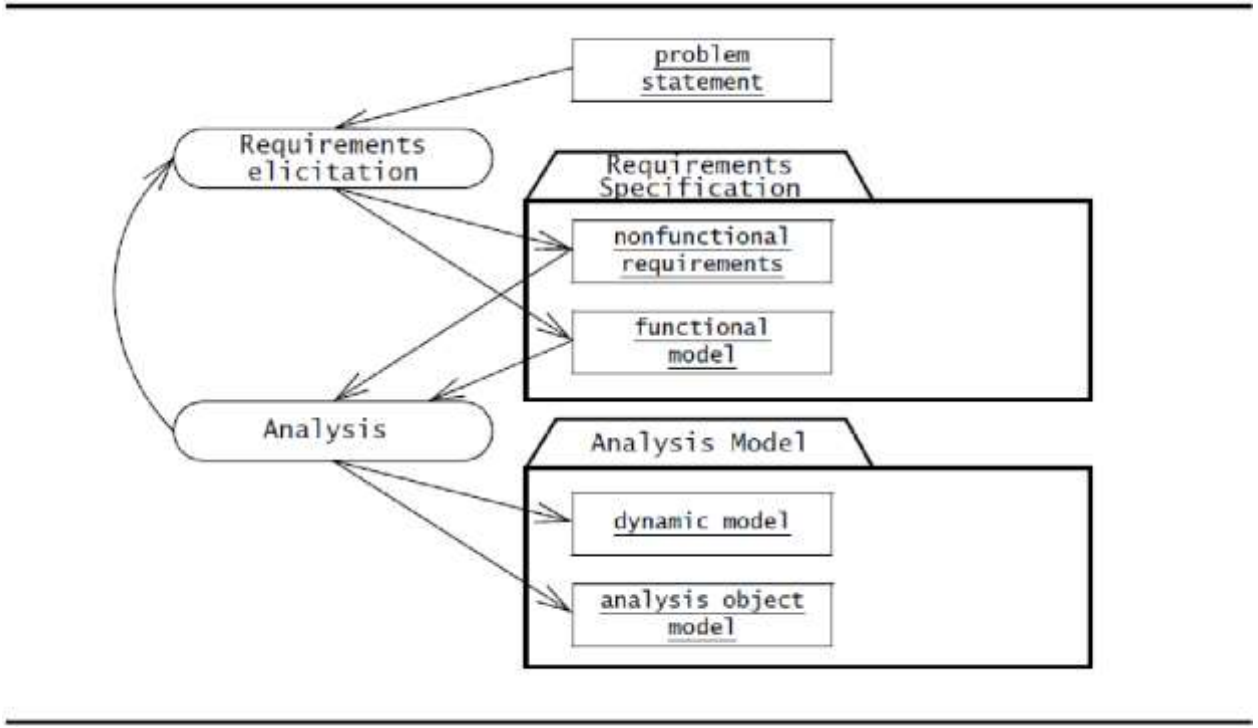
- **Requirements elicitation** focuses on describing the purpose of the system. The client, the developers, and the users identify a problem area and define a system that addresses the problem. Such a definition is called a **requirements specification** and serves as a contract between the client and the developers.

- The requirements specification is structured and formalized during analysis to produce an **analysis model.** Both requirements specification and analysis model represent the same information. They differ only in the language and notation they use; the requirements specification is written in natural language, whereas the analysis model is usually expressed in a formal or semiformal notation.

- The requirements specification supports the communication with the client and users. The analysis model supports the communication among developers. They are both models of the system in the sense that they attempt to represent accurately the external aspects of the system. Given that both models represent the same aspects of the system, requirements elicitation and analysis occur concurrently and iteratively.

Requirements elicitation and analysis focus only on the user's view of the system. For example, the system functionality, the interaction between the user and the system, the errors that the system can detect and handle, and the environmental conditions in which the system functions are part of the requirements.

The system structure, the implementation technology selected to build the system, the system design, the development methodology, and other aspects not directly visible to the user are not part of the requirements

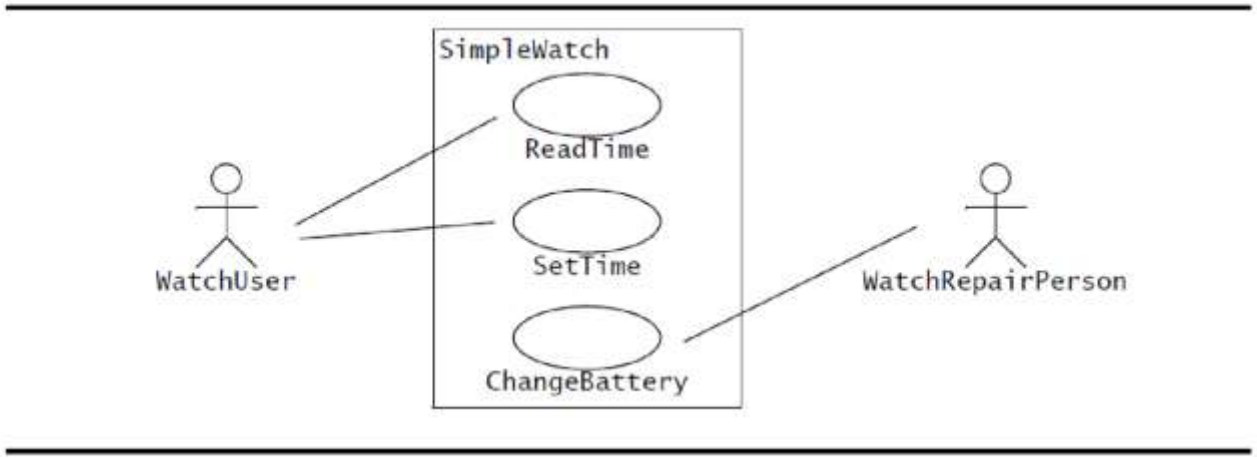


Use Case Diagrams –brief introduction (Detailed description will be provided in UML diagram lecture)

- Use cases are used during requirements elicitation and analysis to represent the functionality of the system.
- Use cases focus on the behavior of the system from an external point of view.
- A use case describes a function provided by the system that yields a visible result for an actor.
- An actor describes any entity that interacts with the system (e.g., a user, another system, the system's physical environment).
- The identification of actors and use cases results in the definition of the boundary of the system, that is, in differentiating the tasks accomplished by the system and the tasks accomplished by its environment.
- The actors are outside the boundary of the system, whereas the use cases are inside the boundary of the system.

Use Case Diagrams –Example

For example, following figure depicts a use case diagram for a simple watch. The WatchUser actor may either consult the time on their watch (with the ReadTime use case) or set the time (with the SetTime use case). However, only the WatchRepairPerson actor can change the battery of the watch (with the ChangeBattery use case).



A UML use case diagram describing the functionality of a simple watch. The WatchUser actor may either consult the time on her watch (with the ReadTime use case) or set the time (with the SetTime use case). However, only the WatchRepairPerson actor can change the battery of the watch (with the ChangeBattery use case). Actors are represented with stick figures, use cases with ovals, and the boundary of the system with a box enclosing the use cases

Requirements elicitation activities

- ***Identifying actors.*** During this activity, developers identify the different types of users the future system will support.

- ***Identifying scenarios.*** During this activity, developers observe users and develop a set of detailed scenarios for typical functionality provided by the future system.

- ***Identifying use cases.*** Once developers and users agree on a set of scenarios, developers derive from the scenarios a set of use cases that completely represent the future system. Whereas scenarios are concrete examples illustrating a single case, use cases are abstractions describing all possible cases. When describing use cases, developers determine the scope of the system.

- ***Refining use cases.*** During this activity, developers ensure that the requirements specification is complete by detailing each use case and describing the behavior of the system in the presence of errors and exceptional conditions.

- ***Identifying relationships among use cases.*** During this activity, developers identify dependencies among use cases. They also consolidate the use case model by factoring out common functionality. This ensures that the requirements specification is consistent.

- ***Identifying nonfunctional requirements.*** During this activity, developers, users, and clients agree on aspects that are visible to the user, but not directly related to functionality. These include constraints on the performance of the system, its documentation, the resources it consumes, its security, and its quality.

Requirements Elicitation Concepts	
Functional requirements	Nonfunctional requirements
Functional requirements describe the interactions between the system and its environment independent of its implementation. The environment includes the user and any other external system with which the system interacts.	Nonfunctional requirements describe aspects of the system that are not directly related to the functional behavior of the system. Nonfunctional requirements include a broad variety of requirements that apply to many different aspects of the system, from usability to performance.

The FURPS+ model2 used by the Unified Process provides the following categories of non functional requirements

- **Usability** is the ease with which a user can learn to operate, prepare inputs for, and interpret outputs of a system or component.
- **Reliability** is the ability of a system or component to perform its required functions under stated conditions for a specified period of time.
- More recently, this category is often replaced by **dependability**, which is the property of a computer system such that reliance can justifiably be placed on the service it delivers. Dependability includes reliability, **robustness** (the degree to which a system or component can function correctly in the presence of invalid inputs or stressful environment conditions), and **safety** (a measure of the absence of catastrophic consequences to the environment).

- **Performance** requirements are concerned with quantifiable attributes of the system, such as **response time** (how quickly the system reacts to a user input), **throughput** (how much work the system can accomplish within a specified amount of time), **availability** (the degree to which a system or component is operational and accessible when required for use), and **accuracy**.

- **Supportability** requirements are concerned with the ease of changes to the system after deployment, including for example, **adaptability** (the ability to change the system to deal with additional application domain concepts), **maintainability** (the ability to change the system to deal with new technology or to fix defects), and internationalization (the ability to change the system to deal with additional international conventions, such as languages, units, and number formats). FURPS+ model, replaces this category with two categories: **maintainability** and **portability** (the ease with which a system or component can be transferred from one hardware or software environment to another).

Requirements Elicitation Activities

There are many activities are performed during requirements elicitation. These activities map a problem statement into a requirements specification which is represented as a set of actors, scenarios, and use cases. Requirements elicitation activities include:

- Identifying Actors
- Identifying Scenarios
- Identifying Use Cases
- Refining Use Cases
- Identifying Relationships Among Actors and Use Cases
- Identifying Initial Analysis Objects
- Identifying Non functional Requirements

Key references

Bernd Bruegge & Allen H. Dutoit - Object-Oriented Software Engineering: Using UML, Patterns, and Java

Thank You