Parul® University

DIGITAL LEARNING CONTENT

Requirements Engineering

Chapter-3:Requirements Engineering

Prof. Arpita Vaidya
Assistant Professor
Department of Computer Science and Engineering

Parul® University

NAAC A++

Content

INDEX

Problem Recognition

Problem recognition in software engineering is the **process of identifying a problem and understanding its nature before attempting to solve it**.

Problem recognition is the initial stage in the software development lifecycle where a **need**, **challenge**, or **opportunity** is identified, which requires a software solution.

It involves understanding and defining the issues or requirements that the software is intended to address. **This phase is crucial as it sets the foundation for all subsequent stages, including requirements gathering, system design, development, and testing.**

Problem Recognition

Key points for problem recognition-

1. **Ask questions**: Ask questions to understand the problem, such as - what the system is supposed to do and what the pain points are.
2. **Identify constraints and goals:** Consider what limits exist, such as - time, budget and technology, and what the desired outcome is.
3. **Involve stakeholders:** Consider the needs and concerns of the people who will be affected by the problem. These people are known as stakeholders.

Problem Recognition

4. **Consider multiple solutions:** Consider multiple solutions to the problem and weigh the pros and cons of each approach.
5. **Prioritizing Requirements:** Requirements are prioritized based on their importance to stakeholders and their impact on achieving project goals.
6. **Documenting the Problem Statement:** The document outlines the context of the project, the identified issues or opportunities, the goals and objectives, and the initial set of requirements to be addressed.

Problem Recognition

7. **Consider multiple solutions:** Consider multiple solutions to the problem and weigh the pros and cons of each approach.
8. **Prioritizing Requirements:** Requirements are prioritized based on their importance to stakeholders and their impact on achieving project goals
9. **Requirements Analysis-** Once requirements are collected, they need to be analyzed to ensure they are clear, complete, consistent, and feasible.
10. **Requirements Validation-** Validation involves checking requirements for correctness, consistency, and completeness through techniques such as reviews, prototyping, simulations, and demonstrations.

Problem Recognition

11. **Requirements Management:** This involves maintaining traceability of requirements, establishing baselines, controlling changes and ensuring that all stakeholders are informed about the status and evolution of requirements.
12. **Consider multiple solutions:** Consider multiple solutions to the problem and weigh the pros and cons of each approach.
13. **Prioritizing Requirements:** Requirements are prioritized based on their importance to stakeholders and their impact on achieving project goals
14. **Requirements Analysis-** Once requirements are collected, they need to be analyzed to ensure they are clear, complete, consistent, and feasible.

What is Requirement?

Image source : Google

What is a Stakeholder?

Image source : Google

Requirement Practices: Loopholes

- Not understanding the requirements that we take from the customer.
- Record requirements in a unorganized manner
- Not spending enough time verifying the record
- Do not allow mechanisms to control the change
- Lack of importance given to the software that the user wants

Requirement Engineering Tasks

- **Seven distinct tasks**
  1. Inception
  2. Elicitation
  3. Elaboration
  4. Negotiation
  5. Specification
  6. Validation
  7. Requirements Management

Inception

- Defines the scope and nature of the problem.
- Context-free questions are asked by software engineer.
- Establish a basic understanding of the problem, the people, nature of the solution and the effectiveness of primary communication and bridge between the customer and the developer.

The First Set of Questions

These questions look on the clients, stakeholders, goals, and the advantages

- **Who is responsible for the request of the work?**
- **Who will utilize the solution?**
- **What will be the financial advantage of a successful solution?**
- **Is there other source for the solution that you require?**

The Next Set of Questions

These questions make the requirements engineer gain a more understanding of the problem and let the customer to say his or her views about a solution

- How would you characterize "good" output that might be generated by a successful solution?
- What problem(s) will this solution address?
- Are you able to show me (or describe) the business environment during which the answer are going to be used?
- Will special performance issues or constraints affect the way the answer is approached?

The Final Set of Questions

These questions look on the effectiveness of the communication itself

- Are you the proper person to answer these questions? Are your answers "official"?
- Are my questions relevant to the matter that you simply have?
- Am I asking too many questions?
- Can anyone else provide additional information?
- Should I be asking you anything else?

Elicitation

- **Take the requirements from the customers.**
- **Eliciting requirements is not easy because of:**
  1. Problems of scope in identifying the boundaries of the system or specifying an excessive amount of technical detail instead of overall system objectives
  2. Problems of understanding what's wanted, what the matter domain is, and what the computing environment can handle

     (Information that's believed to be "obvious" is usually omitted)
  3. Problems of volatility because the wants change over time

Collaborative Requirements Gathering

- Meetings are conducted and attended by both software engineers, customers, and other interested stakeholders
- Rules for preparation and participation are established
- An agenda is usually recommended that's formal enough to hide all details but informal enough to encourage the free flow of ideas
- A "facilitator" (customer, developer, or outsider) controls the meeting
- A "definition mechanism" is employed like work sheets, flip charts, wall stickers, electronic bulletin board, chat room, or another virtual forum
- The goal is to spot the matter , propose elements of the answer , negotiate different approaches.

Quality Function Deployment

- **This is a technique that converts the requirementss of the customer into technical needs for software**
- **It identifies three types of requirements:**
  1. Normal requirements: These requirements are the objectives and goals stated for a product or system during meetings with the customer
  2. Expected requirements: These requirements are implicit to the merchandise or system and should be so fundamental that the customer doesn't explicitly state them
  3. Exciting requirements: These requirements are for features that transcend the customer's expectations and convince be very satisfying

Elaboration

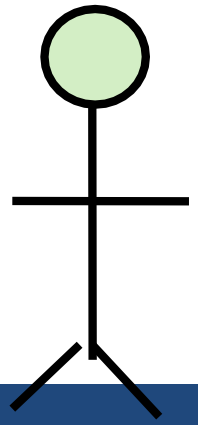
- **During elaboration, the software engineer takes the information obtained during inception and elicitation and begins to expand and refine it.**
- **It is an analysis modeling task:**
  1. Use cases are developed
  2. Domain classes are identified along with their attributes and relationships
  3. State machine diagrams are used to capture the life on an object

Use cases

- **Step One – Define the set of actors that will be involved in the story.**
    1. Actors are people, devices, or other systems that use the system or product within the context of the function and behavior that is to be described
    2. Actors are anything that communicate with the system or product and that are external to the system itself
- **Step Two – Develop use cases, where each one answers a set of questions**

Use cases - Actors

- **An Actor is outside or external the system. It can be a:**
  1. Human
  2. Peripheral Device (Hardware)
  3. External System or Subsystem
  4. Time or time-based event
- **Represented by the following figure:**



Use cases - Relationships

- Represent communication between actor and use case
- Depicted by line or double-headed arrow line
- Also called association relationship

Make Appointment

Negotiation

- During negotiation, the programmer reconciles the conflicts between what the customer wants and what are often achieved given limited business resources.
- Requirements are ranked (i.e., prioritized) by the purchasers , users, and other stakeholders
- Risks related to each requirement are identified and analyzed
- Using an iterative approach, requirements are eliminated, combined and/or modified in order that each party achieves some measure of satisfaction

The Art of Negotiation

- Recognize that it is not competition
- Map out a strategy
- Listen actively
- Focus on the other party's interests
- Don't let it get personal
- Be creative
- Be ready to commit

Specification

- A specification is that the final work product produced by the wants engineer
- it's normally within the sort of a software requirements specification
- It is the inspiration for subsequent software engineering activities
- It formalizes the informational, functional, and behavioral requirements of the proposed software in both a graphical and textual format
- It describes the function and performance of a computer-based system and therefore the constraints which will govern its development

Validation

- **During validation, the products generated as a result of requirements engineering are examined for quality.**
- **The specification is examined to ensure that**
  1. all software requirements are stated unambiguously
  2. inconsistencies, omissions, and errors are detected and corrected
  3. the work products conform to the standards established for the method , the project, and therefore the product
- **The formal technical review is the basic requirements validation mechanism**

Requirements Management

- During requirements management, the project team performs a group of activities to spot , control, and track requirements and changes to the wants at any time because the project proceeds
- Each requirement is assigned a singular identifier
- The wants are then placed into one or more traceability tables
- These tables could also be stored during a database that relate features, sources, dependencies, subsystems, and interfaces to the wants
- A requirements traceability table is additionally placed at the top of the software requirements specification

Requirements Engineering Process

• A Requirement Engineering may be a process during which various activities like discovery, analysis and validation of system requirements are done.

• It begins with feasibility study of the system and finishes up with requirement validation.

• This process may be a three stage activity where the activities are arranged within the iterative manner

• within the early stage of this process most of the time is spent on understanding the system by understanding the high-level non functional requirements and user requirements.

Software Requirement Specification - SRS

- The software requirements specification document enlists all necessary requirements that are required for the project development.
- To derive the wants we'd like to possess clear and thorough understanding of the products to be developed.
- this is often prepared after detailed communications with the project team and customer.

An Example of SRS

Document Title
Author(s)
Affiliation
Address
Date
Document Version

1. Introduction
   1.1. Purpose of this document
   1.2. Scope of this document
   1.3. overview
2. General Description
3. Functional Requirements
   3.1. Description
   3.2. Criticality
   3.3. Technical issues
   3.4. Cost and Schedule
   3.5. Risks
   3.6. Dependencies with other requirements
   3.7. Any other appropriate.
4. Interface Requirements
   4.1. User Interface
   4.1.1. GUI
   4.1.2. CLI
   4.1.3. API
   4.2. Hardware interfaces
   4.3. Communications interfaces
   4.4. Software interfaces.
5. Performance Requirements
6. Design Constraints
7. Other Non-Functionality Attributes
   7.1. Security
   7.2. Binary compatibility
   7.3. Reliability
   7.4. Maintainability
   7.5. Portability
   7.6. Extensibility
   7.7. Reusability
   7.8. Application compatibility
   7.9. Resource utilization
   7.9.1. Serviceability
8. Operational Scenarios
9. Preliminary Schedule
10. Preliminary Budget
11. Appendices

Requirements Validation

- When any model of software is made at that point it's examined for inconsistency, ambiguity , Error etc.
- the wants are often prioritized by the stake holders.
- It grouped with in requirement package which will be implemented by software packages.
- it's one process during which will check about gathered requirements whether it represents an equivalent system or not.
- Here any quite generated errors are often fixed because fixing an requirement errors after delivery may cost up to 100.

Requirements Validation (Contd.)

- **Requirement checking can be done in following manner:**

1. **Validity Checks**
2. **Consistency Checks**
3. **Completeness Checks**

Validity Checks

- A user might imagine that a system is required to perform certain functions.
- In validity check analysis may identify additional or different functions that are required.
- Systems have diverse stakeholders with distinct needs, and any set of requirements is inevitably a compromise across the stakeholder community.

Consistency Checks

- Requirements within the document shouldn't conflict.
- There should be no contradictory constraints or descriptions of an equivalent system function.

Completeness Checks

- **Realism checks-**
  1. Using knowledge of existing technology, the wants should be checked to make sure that they might actually be implemented.
  2. These checks should also appreciate of the budget and schedule for the system development.
- **Verifiability-**
  3. To reduce the potential for dispute between customer and contractor
  4. System requirements should even be written in order that you ought to be ready to write a group of tests which will demonstrate that the delivered system meets each specified requirement.

Requirement Validation Techniques

- **Requirements Reviews-**

1. both the customer and contractor staff should be involved in reviews.
2. Reviews could also be formal (with completed documents) or informal.
3. Good communications should happen between developers, customers and users such a healthy communication helps to resolve problems at an early stage.

- **Prototyping-**

1. The requirements can be examined through executable model of system

Requirement Validation Techniques (Contd.)

- **Test Case Generation-**

1. If the tests for the wants are devised as a part of the validation process, this often reveals requirements problems.
2. If a test is difficult or impossible to style , this usually means the wants are going to be difficult to implement and will be reconsidered.
3. Developing tests from the user requirements before any code is written is an integral a part of extreme programming.

PPT Content Resources Reference :

1. **Book Reference**

   Software Engineering: A Practitioner's Approach, by R.S.Pressman published by TMH.

2. **Reference Books:**

- Software Engineering, 8th Edition by Sommerville, Pearson.
- Software Engineering 3rd Edition by Rajiv Mall, PHI.
- An Integrated Approach to Software Engineering by Pankaj Jalote Wiley India, 2009.

3. **Website Resource :**

- NPTEL - Software Engineering
- MIT OpenCourseWare
- Coursera – Software Engineering Specializations
- edX – Software Development Courses

4. **Sources :**

- geeksforgeeks.org
- tutorialspoint.com
- javatpoint.com

5. **Article : ieeexplore.ieee.org**

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/

Parul® University

DIGITAL LEARNING CONTENT

Requirements Engineering

Chapter-3: Software Research Specification

Prof. Arpita Vaidya
Assistant Professor
Department of Computer Science and Engineering

Parul® University

NAAC A++

INDEX

Content

Software Requirement Specifications

A Software requirement specification (SRS) is a **document** that mentioned **complete description about how the system is expected to perform, behavior of system, functional and non-functional requirements of the system.**

SRS is a formal report, that enables the customers to review whether it (SRS) is according to their requirements.

User of SRS

1. Client
2. Development team
3. Maintenance team
4. Technical writers

Need of SRS Document

❖It structures and formalizes all project requirements.

❖It helps the development team build a product that exactly meets customer and target audience needs.

❖It provides all team members with the necessary information while working on the project.

❖It minimize the possible misunderstanding between the members of the development team and the customer.

❖It clearly define the scope of work and that allows developers to plan particular iterations and release dates.

❖It allows estimating the required development budget.

Characteristics of good SRS

1. **Correctness:** provide accurate functional and non-functional requirements as discussed with client.
2. **Completeness:** Should complete all essential features like functionality, performance, features, design, constraints & external interfaces.
3. **consistency:** Same abbreviation, tabular format, diagram format, naming format follow.
4. **Unambiguousness:** Should not be any confusion regarding understanding of the requirements. Everything mentioned uniquely and clearly.
5. **Ranking for importance and stability:** Every requirement is important. But some are urgent and must be fulfilled before other requirements and some could be delayed.

Characteristics of good SRS

6. **Modifiability:** It is capable of quickly obtain changes to the system without affecting other.
7. **Verifiability:** the requirements are verified with the help of reviewers & stakeholders.
8. **Traceability :** Every requirements having unique number that is easy to use in future development.
9. **Design Independence:** There should be an option to select from multiple design alternatives for the final system. SRS should not contain any implementation details.
10. **Design Independence:** An SRS should be written in such a method that it is simple to generate test cases and test plans from the report.

An Example of SRS

Document Title
Author(s)
Affiliation
Address
Date
Document Version

1. Introduction
   1.1. Purpose of this document
   1.2. Scope of this document
   1.3. overview
2. General Description
3. Functional Requirements
   3.1. Description
   3.2. Criticality
   3.3. Technical issues
   3.4. Cost and Schedule
   3.5. Risks
   3.6. Dependencies with other requirements
   3.7. Any other appropriate.
4. Interface Requirements
   4.1. User Interface
   4.1.1. GUI
   4.1.2. CLI
   4.1.3. API
   4.2. Hardware interfaces
   4.3. Communications interfaces
   4.4. Software interfaces.
5. Performance Requirements
6. Design Constraints
7. Other Non-Functionality Attributes
   7.1. Security
   7.2. Binary compatibility
   7.3. Reliability
   7.4. Maintainability
   7.5. Portability
   7.6. Extensibility
   7.7. Reusability
   7.8. Application compatibility
   7.9. Resource utilization
   7.9.1. Serviceability
8. Operational Scenarios
9. Preliminary Schedule
10. Preliminary Budget
11. Appendices

PPT Content Resources Reference :

1. **Book Reference**

   Software Engineering: A Practitioner's Approach, by R.S.Pressman published by TMH.

2. **Reference Books:**

- Software Engineering, 8th Edition by Sommerville, Pearson.
- Software Engineering 3rd Edition by Rajiv Mall, PHI.
- An Integrated Approach to Software Engineering by Pankaj Jalote Wiley India, 2009.

3. **Website Resource :**

- NPTEL - Software Engineering
- MIT OpenCourseWare
- Coursera – Software Engineering Specializations
- edX – Software Development Courses

4. **Sources :**

- geeksforgeeks.org
- tutorialspoint.com
- javatpoint.com

5. **Article : ieeexplore.ieee.org**

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/

Requirements validation

Requirements Validation Techniques

1. Reviews and Inspections
2. Prototyping
3. Test Case Generation
4. Modeling and Simulation

Challenges in Requirements Validation

1. Difficulty in defining complete and accurate requirements.
2. Stakeholder involvement and collaboration.
3. Balancing cost and quality.
4. Evolving requirements

Requirements validation

Best Practices for Requirements Validation

1. Involve stakeholders early and continuously.
2. Use multiple validation techniques.
3. Document validation activities.
4. Iterative and incremental approach.
5. Prioritize requirements for validation

Requirements Analysis

Requirement analysis is a crucial stage in software development where the needs and expectations of stakeholders are identified and documented.

The requirements are typically categorized into **two types**:

1. functional requirements
2. non-functional requirements

Requirements Analysis

1. **Functional requirements-**

   Functional requirements **focus on** the core **features** and **operations** that the system needs to support.

   Example: For an online banking application- a functional requirement might be: "The system must allow users to transfer funds between accounts."

Requirements Analysis

2. **Non- functional requirements-**

Non-functional requirements address the **quality** and **performance of the system**. They include criteria such as speed, security, scalability, and user experience, and describe how the system should perform under various conditions.

Example: A non-functional requirement for the same banking system might be:

"The application should be able to handle 1,000 transactions per minute without performance issues."

Requirements Analysis

Importance of Requirement Analysis in Software Development

- Clear Understanding of Stakeholder Needs
- Avoiding Scope Creep
- Improved Communication and Collaboration
- Reduced Development Costs
- Minimized Risks and Errors
- Better Project Planning
- Improved Quality and User Satisfaction

Requirements Analysis

Steps involved in the Requirement Analysis Process

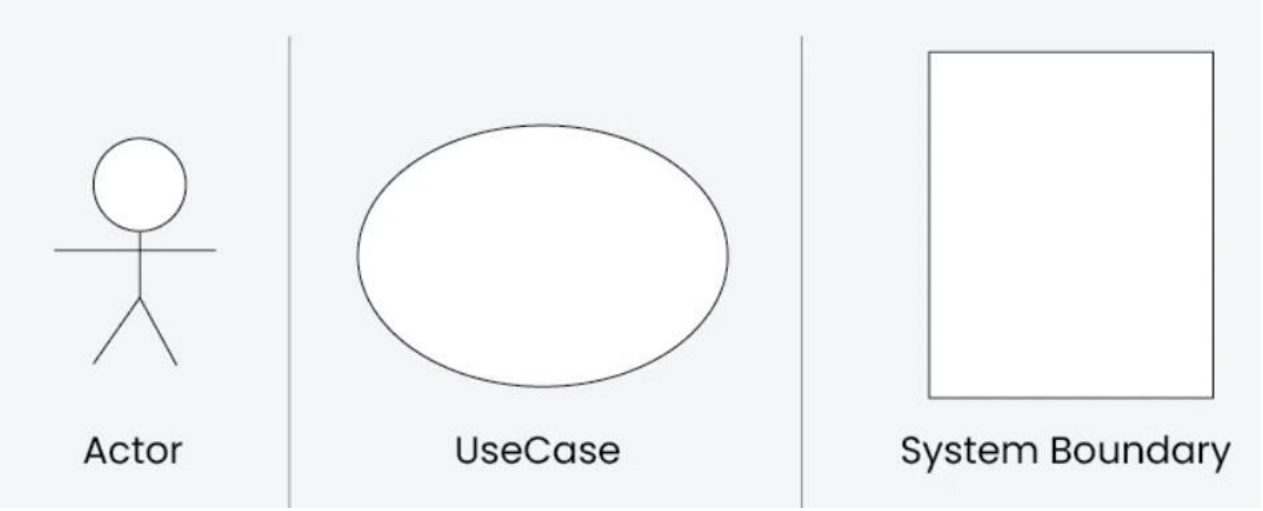
1. Identifying Stakeholders and Communicating Needs
2. Gathering Requirements
3. Analyzing and Prioritizing Requirements
4. Documenting Requirements
5. Validating Requirements
6. Creating Use Cases
7. Managing Changes

Requirements Analysis

Requirement Analysis Techniques

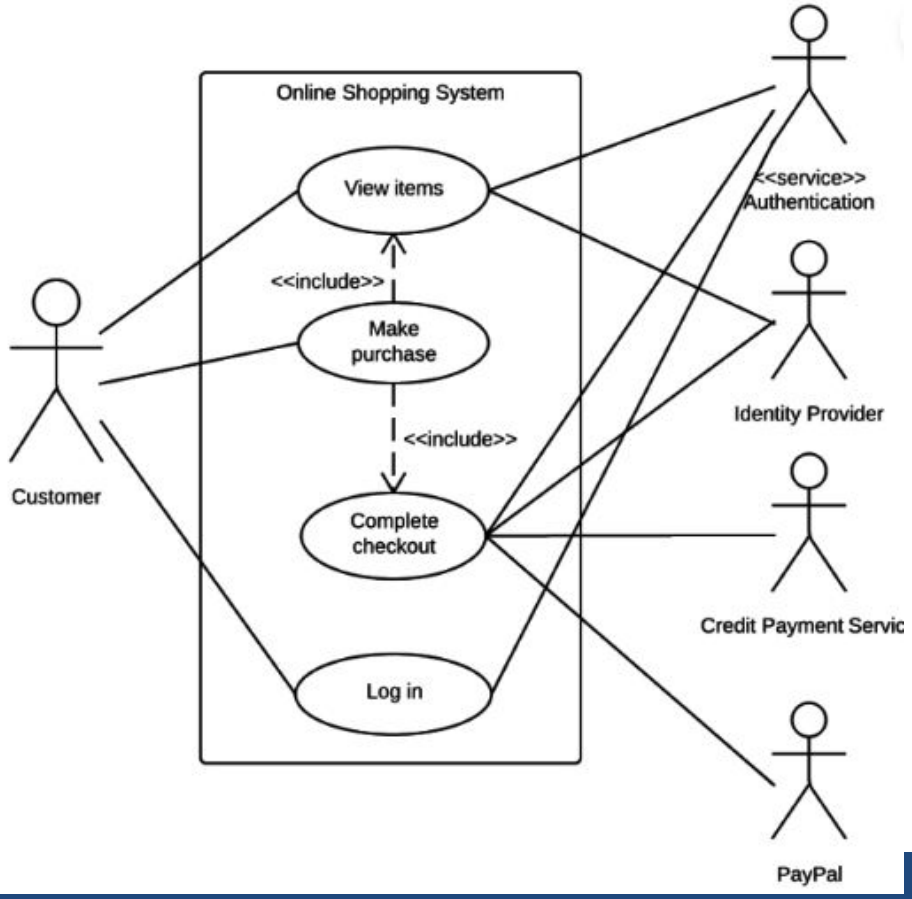
- Data Flow Diagrams (DFD)
- Entity-Relationship Diagrams (ERD)
- Unified Modeling Language (UML)
- State Diagrams
- Flowcharts
- Traceability Matrix

A use case diagram is used to represent the **interaction between actors (users or external systems)** and a system under consideration to accomplish specific goals.



Following are the purposes of a use case diagram given below:

- ✓ It gathers the **system's needs**.
- ✓ It shows **the external view of the system**.
- ✓ It **recognizes the internal as well as external factors** that influence the system.
- ✓ It represents the **interaction between the actors**.


Online Shopping System
View items
<<include>>
Make purchase
<<include>>
Complete checkout
Log in
Customer
<<service>> Authentication
Identity Provider
Credit Payment Servic
PayPal

Parul® University
Vadodara, Gujarat

DIGITAL LEARNING CONTENT

NAAC GRADE

Information and Communication Technology

Parul® University

www.paruluniversity.ac.in

Parul® University

DIGITAL LEARNING CONTENT

NAAC A++

Requirements Engineering

Chapter-3: Use case and Requirement Specifications

Prof. Arpita Vaidya
Assistant Professor
Department of Computer Science and Engineering

Parul® University

NAAC A++

Content

INDEX

Requirements validation

The **process of checking that the requirements accurately reflect the needs of the stakeholders**.

Ensures the software meets user expectations, prevents costly rework, and improves overall project success.

Objectives of Requirements Validation

1. Ensure requirements are **complete, consistent, unambiguous, and verifiable.**
2. **Identify and resolve errors, inconsistencies and omissions early in the development process**.
3. **Improve stakeholder confidence** in the project.

Requirements validation

Requirements Validation Techniques

1. Reviews and Inspections
2. Prototyping
3. Test Case Generation
4. Modeling and Simulation

Challenges in Requirements Validation

1. Difficulty in defining complete and accurate requirements.
2. Stakeholder involvement and collaboration.
3. Balancing cost and quality.
4. Evolving requirements

Requirements validation

Best Practices for Requirements Validation

1. Involve stakeholders early and continuously.
2. Use multiple validation techniques.
3. Document validation activities.
4. Iterative and incremental approach.
5. Prioritize requirements for validation

Requirements Analysis

Requirement analysis is a crucial stage in software development where the needs and expectations of stakeholders are identified and documented.

The requirements are typically categorized into **two types**:

1. functional requirements
2. non-functional requirements

Requirements Analysis

1. **Functional requirements-**

   Functional requirements **focus on** the core **features** and **operations** that the system needs to support.

   Example: For an online banking application- a functional requirement might be: "The system must allow users to transfer funds between accounts."

Requirements Analysis

2. **Non- functional requirements-**

Non-functional requirements address the **quality** and **performance of the system**. They include criteria such as speed, security, scalability, and user experience, and describe how the system should perform under various conditions.

Example: A non-functional requirement for the same banking system might be:

"The application should be able to handle 1,000 transactions per minute without performance issues."

Requirements Analysis

Importance of Requirement Analysis in Software Development

- Clear Understanding of Stakeholder Needs
- Avoiding Scope Creep
- Improved Communication and Collaboration
- Reduced Development Costs
- Minimized Risks and Errors
- Better Project Planning
- Improved Quality and User Satisfaction

Requirements Analysis

Steps involved in the Requirement Analysis Process

1. Identifying Stakeholders and Communicating Needs
2. Gathering Requirements
3. Analyzing and Prioritizing Requirements
4. Documenting Requirements
5. Validating Requirements
6. Creating Use Cases
7. Managing Changes

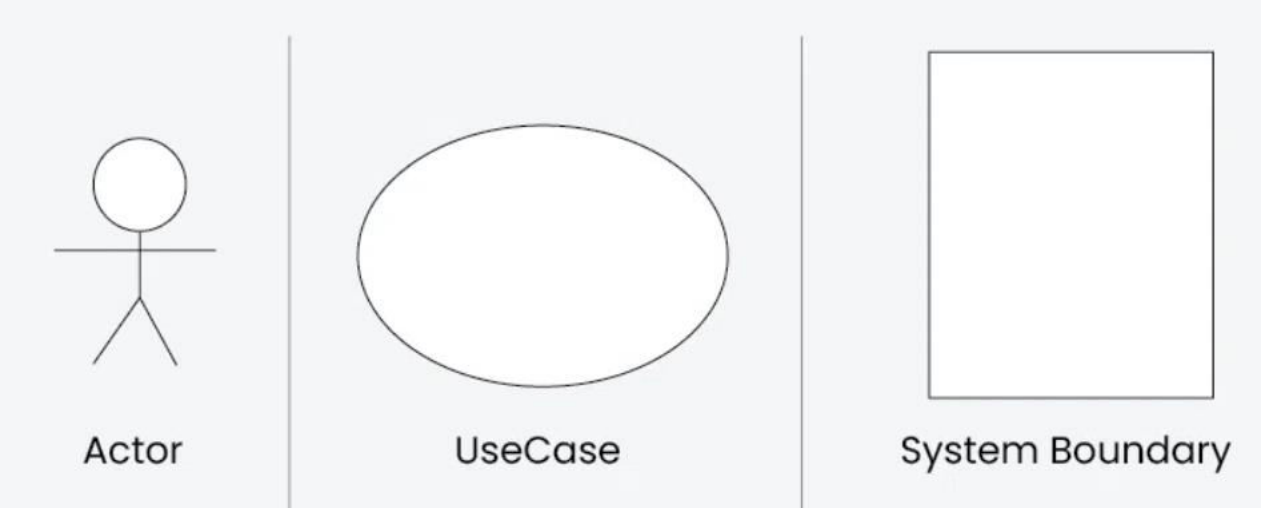
Requirements Analysis

Requirement Analysis Techniques

- Data Flow Diagrams (DFD)
- Entity-Relationship Diagrams (ERD)
- Unified Modeling Language (UML)
- State Diagrams
- Flowcharts
- Traceability Matrix

Use case

A use case diagram is used to represent the **interaction between actors (users or external systems)** and a system under consideration to accomplish specific goals.

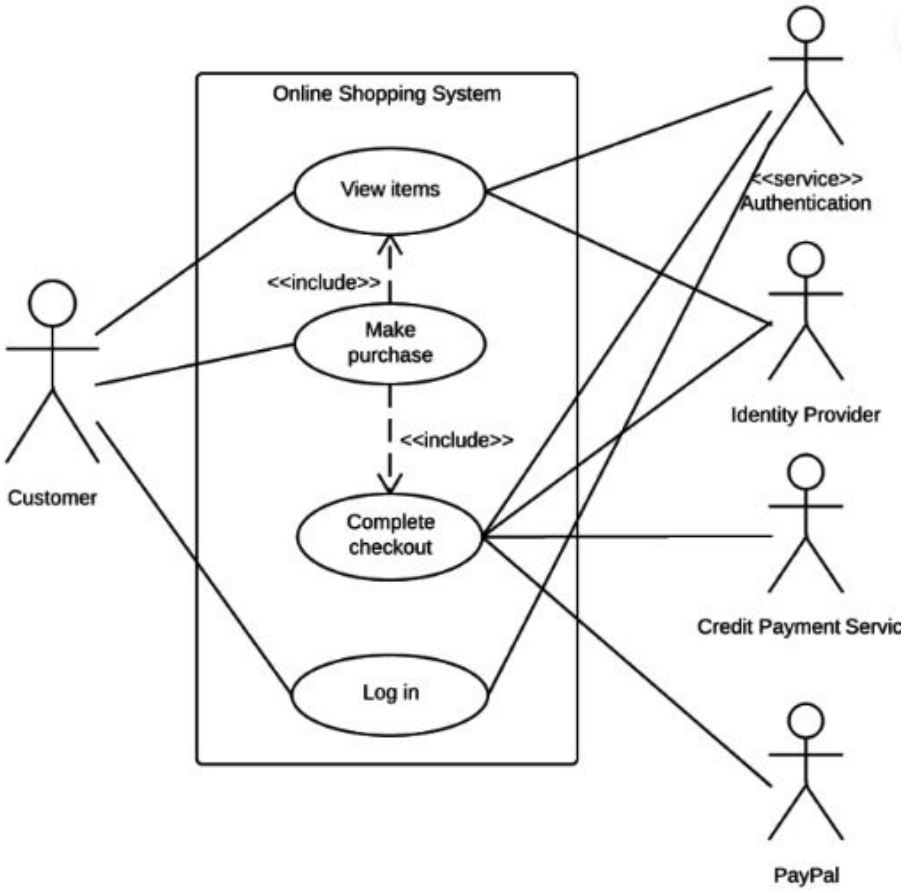


Use case

Following are the purposes of a use case diagram given below:

- ✓ It gathers the **system's needs**.
- ✓ It shows **the external view of the system**.
- ✓ It **recognizes the internal as well as external factors** that influence the system.
- ✓ It represents the **interaction between the actors**.

Use case


Online Shopping System
View items
<<include>>
Make purchase
<<include>>
Complete checkout
Log in
Customer
<<service>> Authentication
Identity Provider
Credit Payment Servic
PayPal

PPT Content Resources Reference :

1. **Book Reference**

   Software Engineering: A Practitioner's Approach, by R.S.Pressman published by TMH.

2. **Reference Books:**

- Software Engineering, 8th Edition by Sommerville, Pearson.
- Software Engineering 3rd Edition by Rajiv Mall, PHI.
- An Integrated Approach to Software Engineering by Pankaj Jalote Wiley India, 2009.

3. **Website Resource :**

- NPTEL - Software Engineering
- MIT OpenCourseWare
- Coursera – Software Engineering Specializations
- edX – Software Development Courses

4. **Sources :**

- geeksforgeeks.org
- tutorialspoint.com
- javatpoint.com

5. **Article : ieeexplore.ieee.org**

Parul® University | NAAC GRADE A++

https://paruluniversity.ac.in/