WOLKITE UNIVERSITY

College of Computing and Informatics

Department of Software Engineering

Course: Software Engineering Tools and Practices

Chapter-Six: Software requirements analysis: reading use case scenarios and use case textual Analysis

By: Mekuria Gemechu

CONTENTS

Introduction to Software Requirements Analysis

- ❖Software Requirements Analysis (SRA) is a crucial phase in the software development lifecycle (SDLC) where the requirements for a software system are identified, documented, validated, and managed.
- ❖It involves a systematic approach to understanding what stakeholders expect from the software and translating these expectations into clear, concise, and unambiguous requirements that serve as a blueprint for the development process.
- ❖Following are the breakdown of the key aspects involved Software Requirements Analysis:

Introduction to Software Requirements Analysis …

- Identification:
    - ✓ This involves gathering information from various stakeholders to understand the needs and expectations for the software system.
    - ✓ Stakeholders may include end-users, customers, managers, domain experts, and other relevant parties.
- Documentation:
    - ✓ Once requirements are identified, they need to be documented systematically.
    - ✓ This documentation typically includes functional requirements (what the system should do) and non-functional requirements (qualities the system should have, such as performance, security, usability, etc.).
- Analysis:
    - ✓ In this phase, requirements are analyzed to ensure they are clear, complete, consistent, and feasible.

Introduction to Software Requirements Analysis …

  - ✓ Ambiguities or contradictions are identified and resolved.
  - ✓ Techniques such as requirement prioritization, requirement traceability, and feasibility analysis may be employed.
- Validation:
  - ✓ Requirements validation ensures that the documented requirements accurately represent the needs of the stakeholders and the intended functionality of the software system.
  - ✓ This may involve techniques such as reviews, walkthroughs, prototyping, simulations, and validation meetings with stakeholders.
- Requirements Management:
  - ✓ Throughout the software development process, requirements may evolve due to changes in stakeholder needs, market conditions, or technological advancements.
  - ✓ Requirements management involves tracking changes to requirements, assessing their impact, and ensuring that the development team works with the most up-to-date set of requirements.

Importance of Software Requirements Analysis

- Alignment with Stakeholder Needs:
  - Requirements analysis ensures that the software system is designed and developed to meet the needs, expectations, and goals of its intended users and stakeholders.
  - By understanding these requirements upfront, developers can create a solution that addresses the specific needs of the end users.
- Reduced Costs and Risks:
  - Identifying and addressing requirements early in the development process can help prevent costly rework and changes later on.
  - By investing time in thorough requirements analysis, potential risks and issues can be identified and mitigated before they escalate, saving time and resources in the long run.

Importance of Software Requirements Analysis …

❖ Improved Communication:

- Requirements analysis facilitates clear communication between stakeholders, including clients, users, developers, and testers.
- It ensures that everyone has a shared understanding of the software's objectives, features, and functionalities, reducing misunderstandings and conflicts during the development process.

❖ Foundation for Design and Development:

- Software requirements serve as the foundation for the design, development, and testing phases of the project.
- They provide guidance for architects and developers to create a system that fulfills the specified needs and functions correctly.

Importance of Software Requirements Analysis …

- ❖ Enhanced Quality Assurance:
  - ▪ Clear and well-defined requirements enable testers to develop comprehensive test cases and scenarios that thoroughly validate the software's functionality.
  - ▪ This ensures that the final product meets the specified requirements and functions as intended.
- ❖ Scope Management:
  - ▪ Requirements analysis helps in defining the scope of the project by determining what features and functionalities are included in the software system and what are not.
  - ▪ This prevents scope creep, where additional features are added without proper evaluation, leading to project delays and budget overruns.

Importance of Software Requirements Analysis …

- Customer Satisfaction:
  - By accurately capturing and addressing the needs of stakeholders, requirements analysis contributes to the development of a software system that aligns with customer expectations.
  - This leads to higher levels of customer satisfaction and increases the likelihood of successful adoption and usage of the software.

Role of Software Requirements Analysis in Software Development Lifecycle

- ❖ Requirements analysis plays a crucial role throughout the software development lifecycle.
- ❖ It serves as the foundation for all subsequent phases, including design, implementation, testing, and maintenance.
- ❖ Initiation:
  - In the initiation phase, requirements analysis helps in defining the scope and objectives of the project, setting the foundation for subsequent stages.
- ❖ Planning:
  - During planning, requirements analysis informs resource allocation, budgeting, and scheduling based on the identified project requirements.

Role of Software Requirements Analysis in Software Development Lifecycle …

- Design:
  - Requirements analysis guides the design process by specifying what the software should do and how it should behave to meet user needs.
- Implementation:
  - Developers use the requirements as a blueprint to code the software, ensuring that it fulfills the specified functionalities.
- Testing:
  - Testers verify that the software meets the documented requirements through various testing methods, including functional, usability, and performance testing.
- Deployment:
  - Prioritized requirements help in determining the rollout strategy and ensure that the deployed software meets user expectations.

Use Case Scenarios and Textual Analysis

- ❖Use case scenarios are a fundamental aspect of software development, system design, and business analysis.
- ❖They are descriptions or narratives that outline how a system, product, or service is utilized in various situations to achieve specific goals or tasks.
- ❖They serve as a tool for defining and understanding the interactions between users (actors) and a system to achieve specific goals.
- ❖Essentially, a use case scenario describes a sequence of events that demonstrates how a system interacts with its users to accomplish a particular task or function.

Components of a Use Case Scenario

❖ Actor:

- An actor represents any entity (user, system, or external element) that interacts with the system to achieve a goal.
- Actors are typically depicted as stick figures and can be primary actors (directly involved in the use case) or secondary actors (involved indirectly or occasionally).

❖ Use Case:

- A use case represents a specific functionality or feature provided by the system to its users.
- It describes a set of interactions between the actors and the system to achieve a particular goal.
- Use cases are often presented as ovals in diagrams, each labeled with a descriptive name.

Components of a Use Case Scenario …

❖ Steps or Sequence of Actions:

- This is the core of the use case scenario.
- It outlines the step-by-step actions performed by the actors and the system to accomplish the desired goal.
- These steps are described in a logical and sequential manner, often in plain language.

❖ Preconditions:

- Preconditions are the conditions that must be true before the use case scenario can be initiated.
- They define the starting state of the system or environment necessary for the successful execution of the scenario.

Components of a Use Case Scenario …

- Postconditions:
  - Postconditions define the expected state of the system or environment after the successful completion of the use case scenario.
  - They describe the outcome or result achieved by executing the scenario.

Approaches to Understanding Use Case Scenarios

- Top-down Approach:
  - In the top-down approach, you start by examining the overarching goals or objectives of the system or application.
  - You then break these down into smaller, more detailed components.
  - This method helps in understanding the big picture before delving into specific details.
  - It's useful for identifying high-level functionalities and their interactions.
- Bottom-up Approach:
  - Conversely, the bottom-up approach involves starting with the specific details or individual components and gradually building up to understand how they contribute to the overall system.
  - This method is beneficial for gaining insights into the finer details of the system's functionalities and how they integrate into the larger structure.

Steps of Analyzing Use Case Scenarios for Requirements

- Identify Actors:
  - Determine the actors involved in the system, including users, external systems, or any other entities interacting with the system.
- Define Use Cases:
  - Identify and define the various use cases, which represent specific interactions between actors and the system to achieve a particular goal.
- Specify Preconditions and Postconditions:
  - Clearly outline the conditions that must be true before a use case can be initiated (preconditions) and the expected outcomes once the use case is completed (postconditions).

Steps of Analyzing Use Case Scenarios for Requirements …

- ❖ Detail Steps and Alternatives:
  - ▪ Describe the steps involved in each use case, including any alternative paths or exceptions that may occur during execution.
- ❖ Identify System Responses:
  - ▪ Determine how the system should respond to different actions or inputs from the actors, including error handling and validation procedures.
- ❖ Validate and Verify Requirements:
  - ▪ Ensure that the use case scenarios accurately capture the functional and non-functional requirements of the system and that they align with stakeholders' expectations.

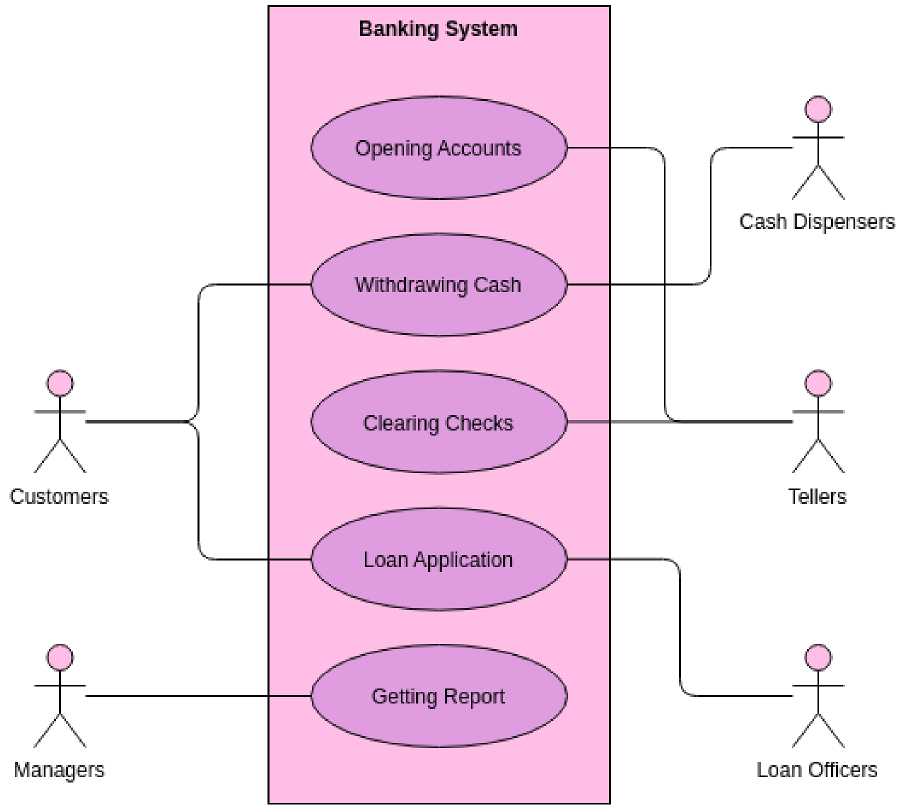
Common Pitfalls and How to Avoid Them

- Incomplete Scenarios:
  - Ensure that use case scenarios cover all possible interactions and edge cases to avoid overlooking essential functionalities.
- Ambiguous Requirements:
  - Clearly define use case scenarios to avoid ambiguity and misinterpretation by stakeholders or development teams.
- Overly Complex Scenarios:
  - Keep use case scenarios concise and focused to prevent confusion and make them easier to understand and implement.

Common Pitfalls and How to Avoid Them …

- **Neglecting Non-Functional Requirements**:
  - Include non-functional requirements such as performance, security, and scalability considerations in the use case scenarios to ensure a comprehensive understanding of system needs.
- **Lack of Stakeholder Involvement**:
  - Involve stakeholders throughout the use case scenario analysis process to gather feedback and ensure alignment with their expectations and needs.
- By applying these approaches and considerations, you can effectively understand, analyze, and avoid common pitfalls when working with use case scenarios for system development or requirements elicitation.

A simple use case scenario for a banking system



Use Case Model: Banking System

A simple use case scenario for a banking system …

- Use Case: Transfer Funds
- Actors: Customer, System
- Preconditions:
  - The customer must be authenticated and logged into their online banking account.
  - Sufficient funds must be available in the source account.
- Steps:
  1. The customer selects the "Transfer Funds" option from the main menu.
  2. The system presents the customer with options to select the source and destination accounts, as well as the amount to transfer.

A simple use case scenario for a banking system …

1. The customer enters the necessary details, including the source account, destination account, and transfer amount.
2. The system validates the transfer request, ensuring that the source account has sufficient funds.
3. If the validation is successful, the system debits the transfer amount from the source account and credits it to the destination account.
4. The system updates the transaction history for both accounts.
5. The system displays a confirmation message to the customer, indicating that the funds transfer was successful.
6. The use case ends.

A simple use case scenario for a banking system …

- ❖Postconditions:
  - The transfer amount is deducted from the source account balance.
  - The transfer amount is added to the destination account balance.
  - The transaction history for both accounts is updated.
  - A confirmation message is displayed to the customer.
- ❖In this example, the use case scenario outlines the steps involved in transferring funds between accounts within a banking system.
- ❖It defines the interactions between the customer and the system, along with the preconditions and postconditions for the successful execution of the scenario.

Textual Analysis

- ❖Textual analysis is a method of studying a text in order to understand the various meanings by identifying the who, what, when, where, why, and how of a text.
- ❖It is a method used to interpret and understand written, spoken, or visual communication.
- ❖It involves dissecting the content, structure, and language used in a text to uncover deeper meanings, themes, and messages.
- ❖This analytical approach is commonly employed in various fields such as literature, linguistics, media studies, communication studies, and cultural studies.

Importance of Textual Analysis

- ❖Textual analysis is crucial in understanding and interpreting use cases effectively.
- ❖It allows stakeholders to grasp the requirements, functionalities, and interactions within a system.
- ❖Through textual analysis, ambiguities and inconsistencies can be identified and resolved early in the development process, leading to more accurate and efficient system design and implementation.
- ❖Additionally, textual analysis helps in ensuring that the use cases align with the goals and objectives of the project, thereby improving communication and collaboration among stakeholders.

Process Involved in Textual Analysis

- ❖ Descriptive Stage:
  - This involves a thorough reading and summarization of the text.
  - Understanding the broader context, author's perspective, and intended audience form an integral part of this stage.
- ❖ Analytical Stage:
  - Drawing inferences and interpreting meanings comes under this stage.
  - The occurrences of specific terms, recurrent themes, and patterns are analyzed here.

Process Involved in Textual Analysis …

- Interpretive Stage:
  - Unraveling underlying meanings and implicit messages that the text represents, and understanding symbolism, metaphorical representations, and other linguistic nuances fall under this stage.
- Evaluative Stage:
  - The text is judged against certain pre-set standards or criteria.
  - Questions like how engaging or persuasive the text is, how well the argument is built, or how impactful the delivery was can be answered here.

Techniques for Analyzing Use Case Text

- **Identifying Actors and Actions**:
  - Identify the various actors involved in the system, such as users, external systems, or hardware devices.
  - Determine the actions or tasks that each actor can perform within the system.
  - Classify actors based on their roles and responsibilities in relation to the system.
- **Extracting System Functions and Behaviors**:
  - Extract the functions or operations that the system needs to perform to satisfy the requirements outlined in the use case.
  - Define the behaviors or responses of the system to different stimuli or inputs from actors.
  - Analyze the flow of events within the use case to identify key functions and behaviors.

Techniques for Analyzing Use Case Text …

- ❖Clarifying Relationships and Dependencies:
  - Identify the relationships between actors and the system, such as associations, dependencies, or hierarchies.
  - Determine the dependencies between different use cases or system components.
  - Clarify the sequence of interactions and dependencies to ensure the coherent and consistent behavior of the system.

Tools and Methods for Textual Analysis

- Natural Language Processing (NLP) tools:
  - Utilize NLP techniques to parse and analyze textual descriptions of use cases automatically.
- Use Case Diagrams:
  - Create visual representations of use cases and their relationships using diagrams to aid in analysis and comprehension.
- Text Mining:
  - Apply text mining techniques to extract key concepts, relationships, and patterns from use case documents.

Tools and Methods for Textual Analysis …

- **Requirement Management Tools:**
  - Utilize tools specifically designed for managing requirements to organize, analyze, and track use cases and their associated details.
- **Stakeholder Collaboration Platforms:**
  - Use collaborative platforms to facilitate communication and collaboration among stakeholders during the analysis of use cases.
- **Domain-specific Analysis Techniques:**
  - Employ domain-specific analysis techniques and methodologies tailored to the characteristics and requirements of the system being developed.

Three key pillars of best practices for software requirements analysis

❖ Communication and Collaboration with Stakeholders:

- This involves engaging with all relevant stakeholders throughout the software development process.
- This ensures that their needs, expectations, and constraints are thoroughly understood and considered.
- Effective communication helps in gathering accurate requirements and avoiding misunderstandings.
- Techniques such as interviews, surveys, workshops, and regular meetings can facilitate communication and collaboration with stakeholders.

Three key pillars of best practices for software requirements analysis …

- Iterative Approach to Requirements Elicitation:
  - Rather than attempting to gather all requirements upfront in a single phase, adopting an iterative approach allows for refining and updating requirements over time.
  - This approach acknowledges that requirements may evolve as stakeholders gain a deeper understanding of the project and its implications.
  - It also allows for early validation of assumptions and identification of potential issues.
  - Techniques such as prototyping, user stories, and continuous feedback loops support this iterative approach.

Three key pillars of best practices for software requirements analysis …

- ❖Documenting and Validating Requirements:
  - ▪ Proper documentation of requirements is essential for ensuring clarity, completeness, and traceability throughout the software development lifecycle.
  - ▪ This documentation serves as a reference for all stakeholders and helps in managing changes effectively.
  - ▪ Additionally, validating requirements involves verifying that they accurately capture stakeholders' needs and align with the project's objectives.
  - ▪ Techniques such as reviews, walkthroughs, and prototypes can be used to validate requirements and ensure their quality.
- ❖By incorporating these best practices into software requirements analysis, teams can enhance collaboration, adaptability, and ultimately, the success of the software development project.

Thank you!