Course: Software Requirements Engineering

Week 7: Requirements Analysis

Lecturer: Yimer Amedie (MSc.)

Addis Ababa Science and Technology University

May, 2026

Contents

- Introduction
- Overview of Requirement Analysis
- Goal of Requirement Analysis
- Conflict Detection and Consistency Checking
- Requirement Classification Methods
- Completeness and Quality of Requirements
- Requirements Interaction Analysis

Figure 1. *The software requirements*
Note. Image generated using Sora by OpenAI (2026).

Learning Outcomes

After completing this lesson, you will be able to:

- Explain the process and purpose of requirement analysis
- Identify and resolve ambiguity in software requirements
- Detect conflicts and ensure consistency among requirements
- Classify requirements into appropriate categories
- Evaluate requirements for completeness and quality

Introduction

From Requirements Elicitation to Analysis

- ✓ Requirements engineering lifecycle (Beatty, 2013)
  - ✓ **Steps: Elicitation ➔ Analysis ➔ Specify ➔ Validation ➔ Mgmt.**
  - ✓ Raw requirements may contain issues
    - ✓ Need for refinement and deeper understanding
      - ✓ Leads to requirement analysis phase

What is Requirement Analysis?

- ✓ Process of **examining**, **refining**, and **structuring** the requirements gathered during elicitation.

Why Requirement Analysis Matters?

Incomplete requirements

Important details may be missing or unclear.

Conflicting stakeholder needs

Different priorities and goals lead to conflicts.

Unrealistic expectations

Stakeholders may expect more than what is feasible in time, cost or technology.

Hidden assumptions

Unspoken beliefs or assumptions can lead to misunderstandings later.

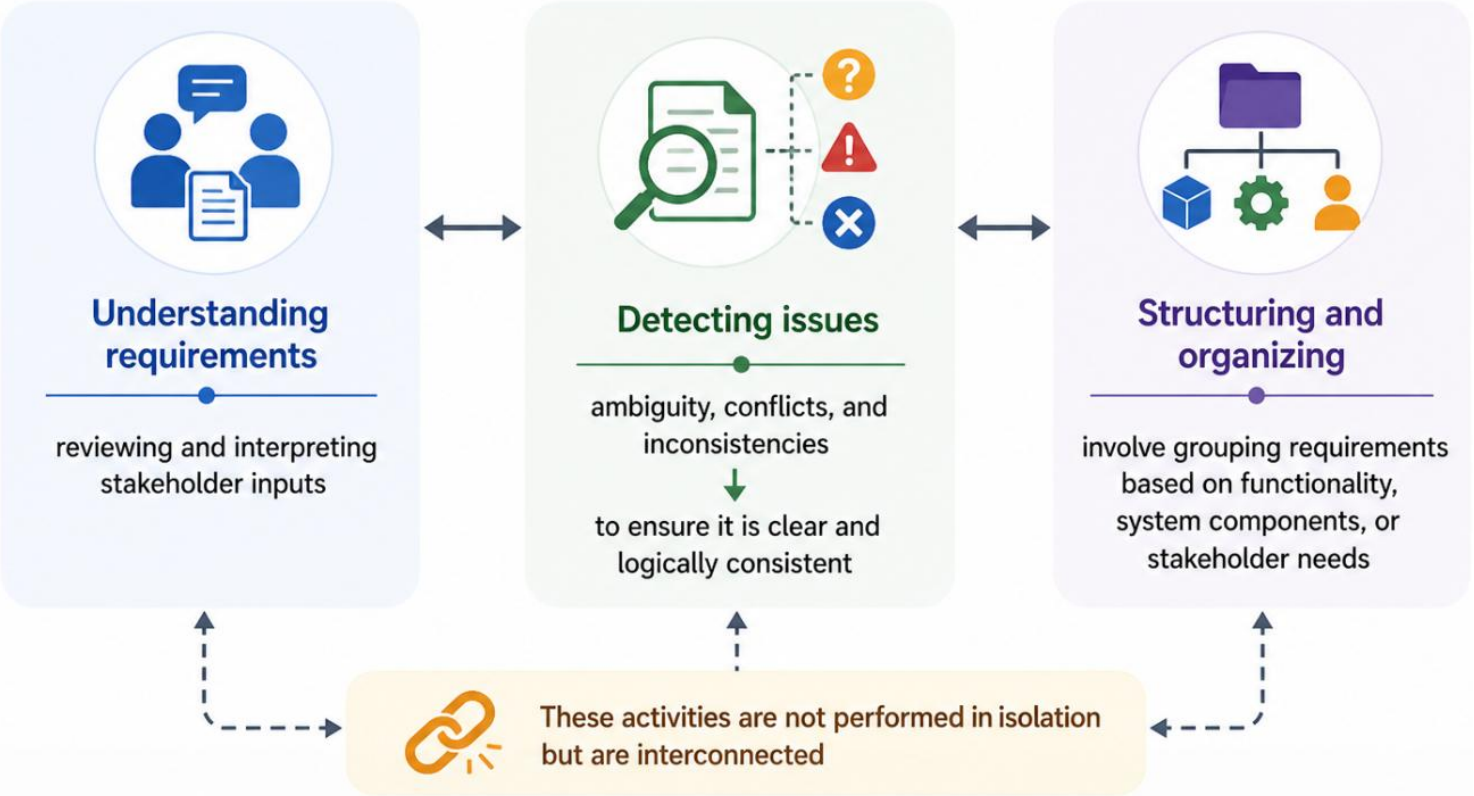
Importance of Requirement Analysis

Reduces errors and rework

Improves requirement clarity

Supports better system design

Key Activities in Requirement Analysis



Goals of Requirements Analysis

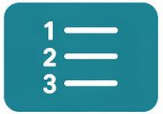
Remove ambiguity

Ensure completeness

Check consistency

Assess feasibility

Validate with stakeholders



Prioritize requirements

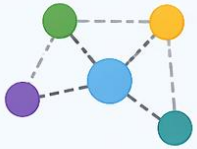
➡ **Output:** Well-structured and validated requirements

More requirements ≠ better system

☑ Quality > Quantity

Principles of Requirement Analysis

No requirement exists in isolation



Conflicts are normal, not exceptions

Negotiation is continuous

Balance is more
important than perfection

Concept of Ambiguity

Multiple interpretations

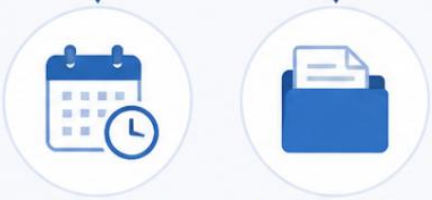
Lack of clarity

Common in natural language

Types of Ambiguity

1 Lexical ambiguity

The student can submit the **assignment** until Friday.



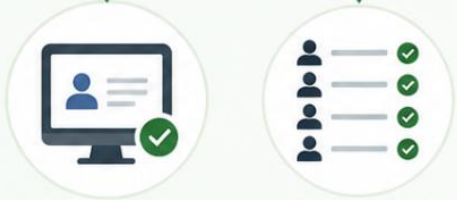
An assignment (one task)

All assignments (multiple tasks)

The word used has more than one meaning.

2 Syntactic ambiguity

Students can view the grades of submitted assignments.



View grades (for their own submitted assignments)

View grades (of all students' submitted assignments)

The sentence structure allows more than one interpretation.

3 Semantic ambiguity

The system allows students to retake the quiz.

Retake the same quiz (again)

Retake another quiz (different quiz)

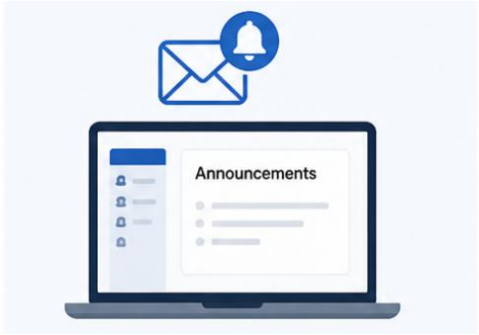
The meaning changes based on the context or intent.

Identifying and resolving ambiguity in requirements ensures clarity and prevents misunderstandings in an online learning system.

Resolving Ambiguity

Use clear language

Before (Ambiguous)	After (Clear)
The system should notify students.	The system shall send an email notification to students when a new announcement is posted.



Define terms

Before (Ambiguous)	After (Defined)
The system should provide timely feedback.	Timely feedback: Feedback provided to the student within 48 hours after submission.

Definitions

Timely feedback	Feedback provided to the student within 48 hours after submission.
Active user	A user who has logged in at least once in the last 30 days.
Course	A collection of learning materials, activities, and assessments.

Add measurable criteria

Before (Ambiguous)	After (Measurable)
The system should load quickly.	The system shall load any page within 3 seconds for 95% of users under normal load conditions.

Conflict in Requirements

Conflict Detection Techniques

1 Requirement comparison

Compare requirements to find contradictions or inconsistencies.

Example (Online Learning System)

Requirement A (Instructor)	Requirement B (Student)
The system shall allow instructors to delete any student submission.	The system shall not allow any student submission to be deleted.

Conflict Detected
These requirements contradict each other.

Helps identify direct contradictions between requirements.

2 Stakeholder review

Review requirements with stakeholders to uncover conflicting needs or expectations.

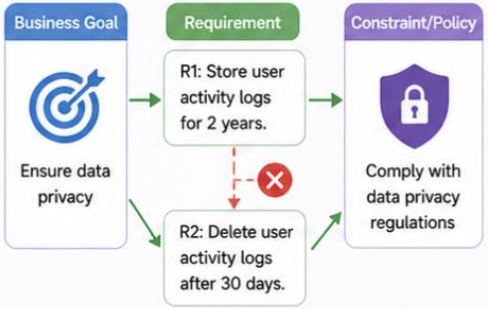
Example (Online Learning System)

Helps reveal conflicts arising from different stakeholder perspectives and priorities.

3 Traceability analysis

Analyze requirement relationships and dependencies to find conflicts.

Example (Online Learning System)



Conflict Detected
R1 and R2 cannot be satisfied together.

Helps find conflicts caused by dependencies or constraints.

Requirements Classification

1. Organizing requirements into categories

Group similar requirements based on functionality, user type, priority, or other relevant criteria.

2. Improves clarity and structure

Provides a clear view of what the system should do and for whom.

3. Supports better analysis

Easier to identify gaps, conflicts, dependencies, and missing requirements.

4. Enables easier management

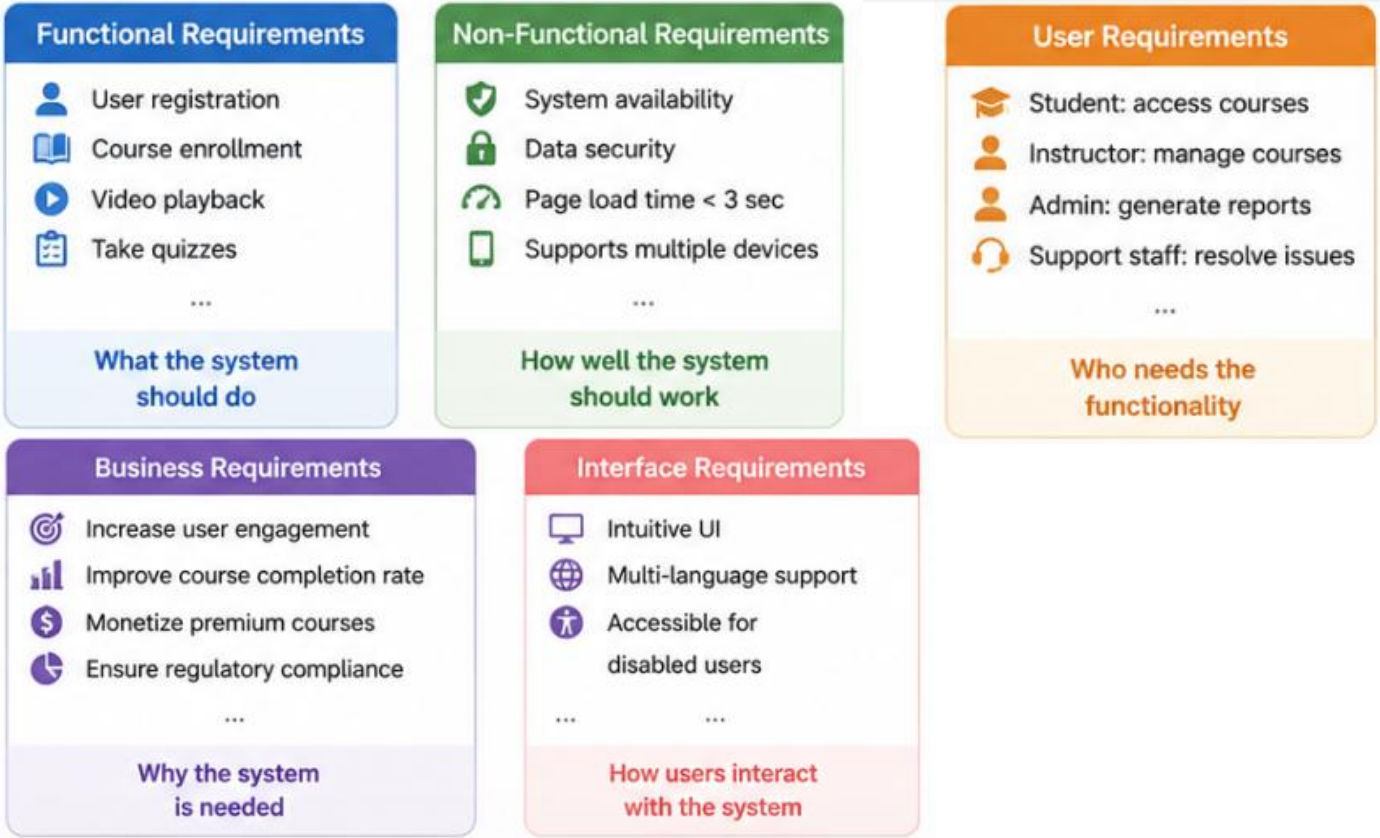
Helps in tracking, prioritizing, estimating, and assigning requirements.

5. Reduces redundancy

Avoids duplicate requirements and overlapping functionality.

Requirement classification means organizing requirements into meaningful categories based on their characteristics, purpose, or function in the online learning system.

Requirements Classification – Example



Requirement Structuring

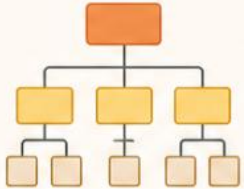
1. Organizing requirements logically

Structure requirements in a clear and consistent manner.

2. Grouping related requirements

Cluster similar requirements into meaningful groups.

3. Using hierarchies and models

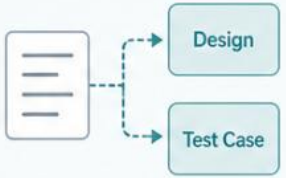


Use hierarchies, use case or domain models to represent relationships.

4. Improving readability

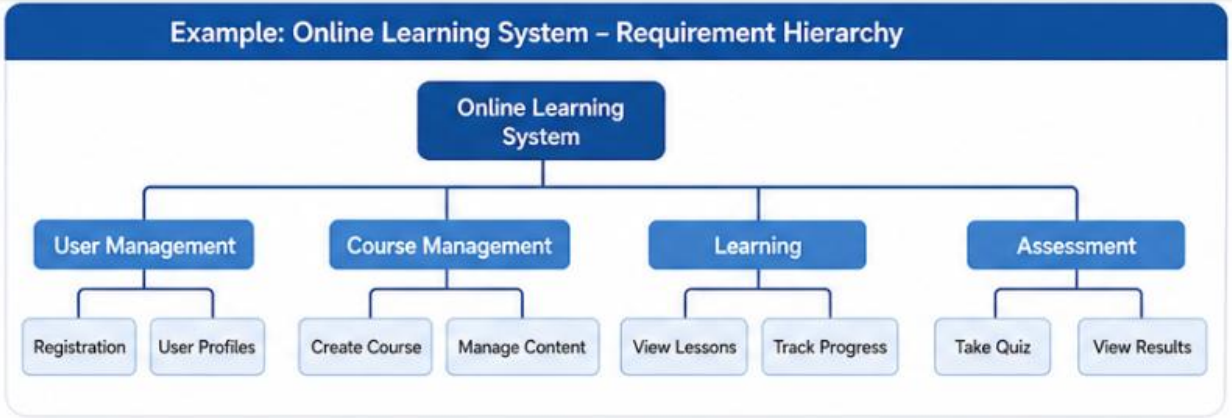
Make requirements easy to understand and review.

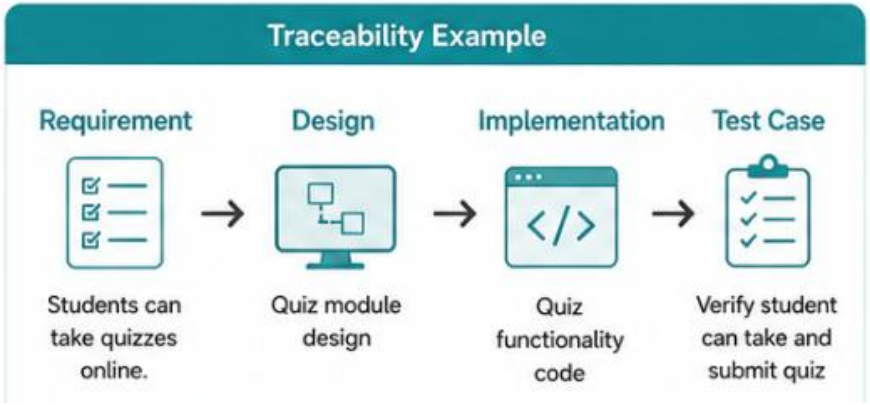
5. Supporting traceability



Enable tracking from requirements to design, implementation and tests.

Requirement Structuring – Example





Well structured requirements leads to **better understanding, easier maintenance** and **successful project outcomes.**

Concept of Completeness

1. All required information included

Every requirement has the necessary details like what, who, when, where, how and constraints.

Example (Online Learning System)

"Students can submit assignments." includes details like file types, size limit, deadline, and submission method.

2. No missing requirements

All essential requirements are identified; nothing important is left out.

Example (Online Learning System)

Includes login, course enrollment, content access, progress tracking, notifications, payments, reports, etc.

3. Covers all scenarios

Requirements handle normal, alternative, and exception situations.

Example (Online Learning System)

Covers scenarios like:

- Successful login
- Incorrect password
- Forgot password
- Account locked

4. Addresses all stakeholders

Considers needs and expectations of every relevant stakeholder.

Example (Online Learning System)

- Students
- Instructors
- Administrators
- Support Staff
- Business/Management

5. Supports system goals

All requirements contribute to achieving the overall objectives of the system.

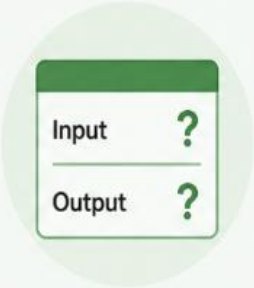
✓ Completeness ensures that requirements are whole, fully defined and nothing important is missing.

Concept of Completeness – Example – OLMS

✓ A complete set of requirements leads to a reliable system, fewer changes, and higher stakeholder satisfaction

Indicators of Incomplete Requirements

Missing scenarios



Undefined inputs or outputs

Lack of error handling

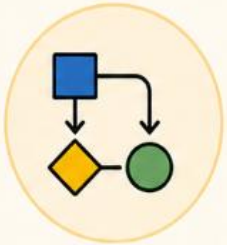
Unclear system boundaries

Gaps in stakeholder needs

Ensuring Completeness

Use checklists

Conduct stakeholder reviews



Analyze scenarios

Validate against objectives

Iterate and refine

Requirements Relationship & Interaction Analysis

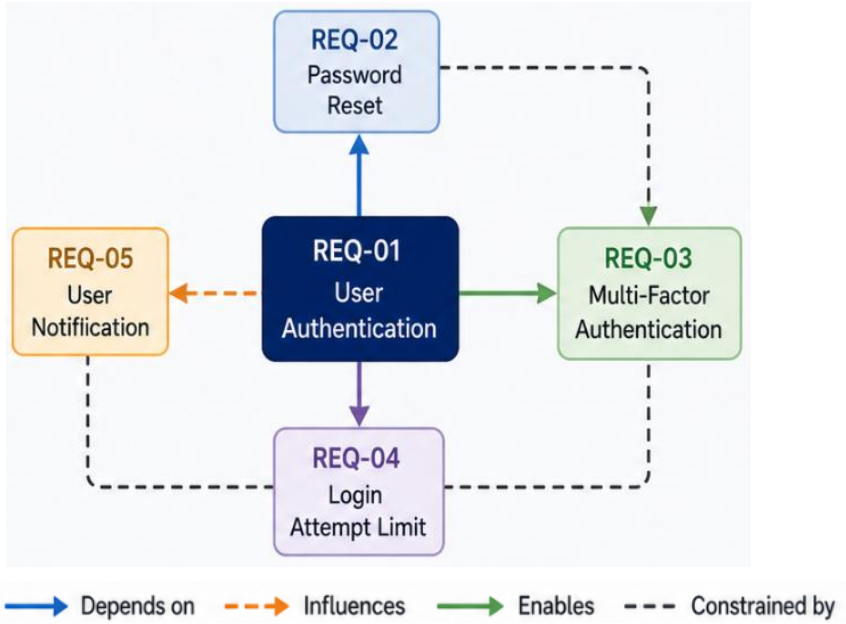
Requirements are interconnected and interdependent.

Relationships define how requirements influence each other.

Includes dependencies, overlaps, and constraints.

Interaction analysis examines these relationships deeply.

Supports consistency and system-wide understanding.



Types of Relationships & Interactions

Dependency
(one requirement relies on another)

- One requirement depends on the fulfillment or existence of another.

Example:

R1: User Login → R2: Reset Password

R2 depends on R1

Conflict
(requirements contradict each other)

- Two or more requirements have opposing or contradictory conditions.

Example:

R1: System must allow guest access ↔ R2: System must require login

R1 conflicts with R2

Overlap
(duplicate or similar requirements)

- Requirements have similar intent or cover the same functionality.

Example:

R1: System shall generate reports  R2: System shall create reports

R1 and R2 overlap in functionality

Types of Relationships & Interactions … cont'd

Complementary
(requirements support each other)

- Requirements work together to achieve a common goal or enhance functionality.

Example:

R2 complements R1

Constraint-based relationships

- Requirements are linked by rules, limitations, or external constraints.

Example:

R2 is constrained by R1

Understanding different types of relationships helps in analyzing interactions, resolving issues, and ensuring consistent and effective requirements.

Techniques for Relationships and Interactions

Requirement traceability matrix

Cross-referencing and comparison

Visual modeling (diagrams)

Stakeholder review sessions

Impact analysis of changes

Requirements Relationship – Final Remark!

Understanding relationship ensures consistent requirements, better change management and complete traceability.

1. Dependencies between requirements

Some requirements rely on others to be implemented first.

2. Parent-child relationships

High-level requirements are broken down into more detailed sub-requirements.

3. Functional and non-functional links

Functional requirements are supported or constrained by non-functional requirements.

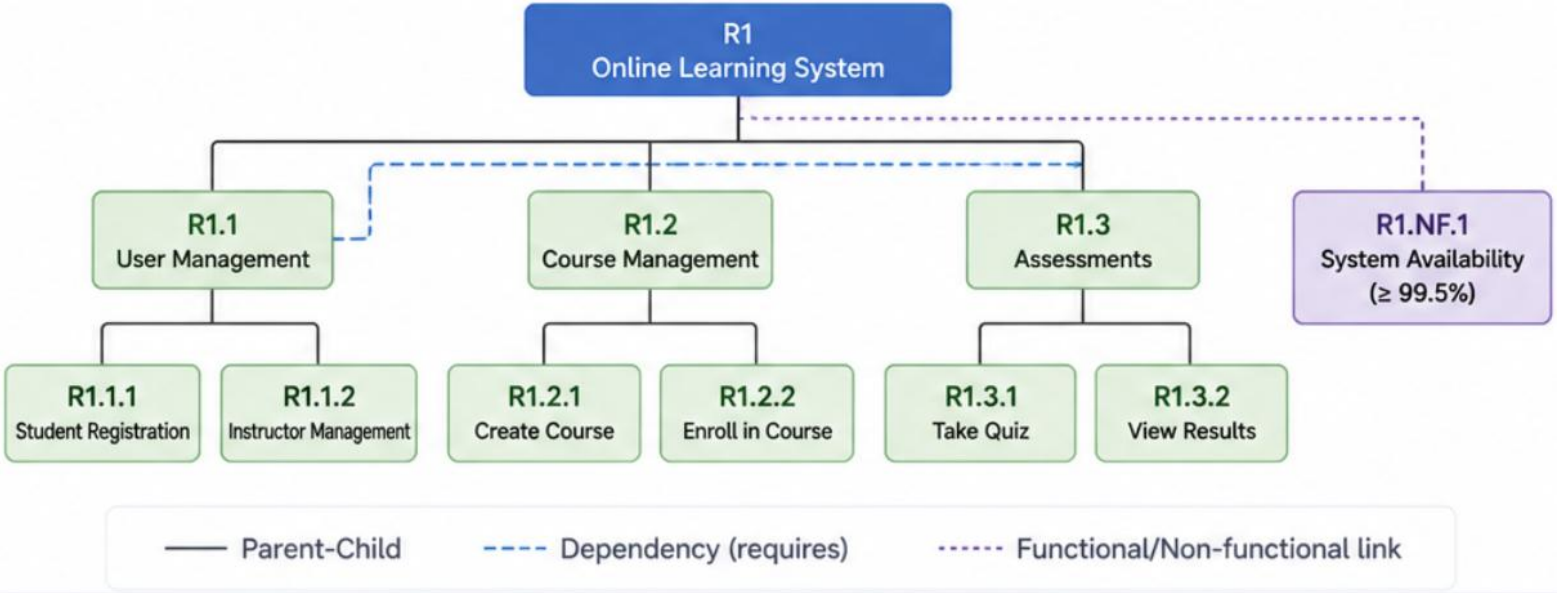
4. Impact of changes

A change in one requirement may affect others that are related.

5. Supports traceability

Helps track relationships across the requirement lifecycle.

Requirement Relationship – Example 1



Impact of change

If **R1.1.1** student registration changes (e.g. **add email verification**), it impacts

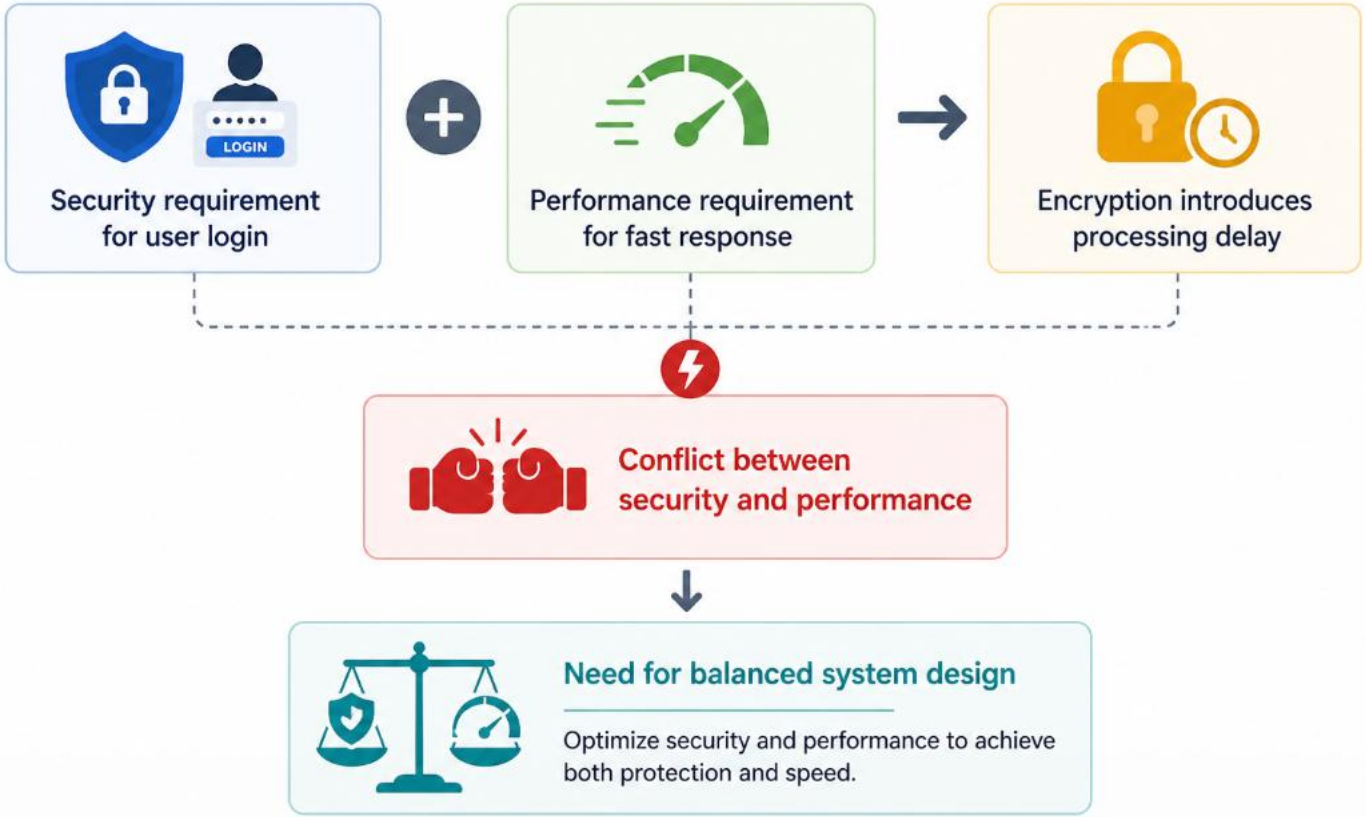
- ✓ **R1.1.2** Instructor mgmt.
- ✓ **R1.2.2** Enroll in Course

Traceability

R1.2.2 Enroll in Course is linked to:

- ✓ **Use Case:** Enroll Course
- ✓ **Test Case:** Verify course enrollment
- ✓ **Goal:** Improve learning access

Requirement Relationship – Example 2



Importance of Relationship & Interaction Analysis

Ensures requirement consistency

Reveals hidden dependencies

Prevents conflicts and redundancy

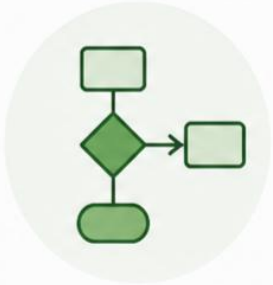
Improves system quality

Supports informed design decisions

Requirement Modeling

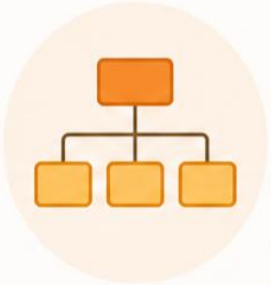
1. Visual representation of requirements

Presents requirements in a clear and visual manner.



2. Use of diagrams

Uses models like use case, activity, sequence, class diagrams, etc.



3. Simplifies complex systems

Breaks down complex requirements into manageable parts.

4. Enhances understanding

Helps stakeholders better understand requirements.

5. Supports communication

Consistency in Requirement

No contradictions

Requirements do not conflict with each other.

Uniform terminology

Same terms used consistently throughout.

Aligned objectives

All requirements support the same goals.

Compatible constraints

Constraints do not conflict and are feasible together.

Clear relationships

Dependencies and links between requirements are well-defined.

Consistent requirements lead to reliable systems, efficient development, and successful delivery.

Common Mistakes in Requirement Analysis

Ignoring ambiguity

Unclear requirements lead to different interpretations and wrong solutions.

Overlooking conflicts

Conflicting requirements cause confusion, rework, and project delays.

Missing key requirements

Important requirements left out result in incomplete and unsatisfactory systems.

Poor classification

Improper categorization makes requirements hard to manage and track.

Lack of stakeholder involvement

Not involving stakeholders leads to misunderstood needs and low user satisfaction.

Avoiding these common mistakes improves requirement quality and increases the chances of project success.

Role of Analyst in Requirement Analysis

Interpret requirements

Understand stakeholder needs and translate them into clear requirements.

Identify issues

Detect gaps, conflicts, ambiguities, and inconsistencies early.

Ensure clarity

Refine requirements to be clear, complete, and unambiguous.

Communicate with stakeholders

Collaborate with stakeholders to validate and align requirements.

Organize requirements

Structure and prioritize requirements for effective analysis and traceability.

The analyst bridges the gap between stakeholder needs and solutions through careful analysis, clear communication, and structured documentation.

Checklist for Requirement Analysis

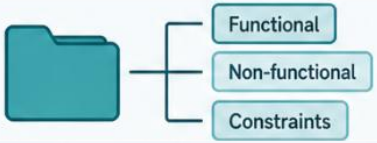
Are requirements clear?

Are there ambiguities?

Are there conflicts?

Are requirements complete?

Are they properly classified?



Use this checklist to ensure high-quality, reliable, and well-structured requirements.

Benefits of Effective Requirement Analysis

Reduces development errors

Clear requirements minimize mistakes and rework.

Improves system quality

Well-defined requirements lead to reliable, efficient, and user-friendly systems.

Enhances communication

Promotes better understanding and alignment among all stakeholders.

Saves time and cost

Early clarity prevents delays, rework, and unnecessary expenses.

Supports successful delivery

Well-analyzed requirements increase the chances of delivering the right solution on time.

Effective requirement analysis lays the foundation for the right solution, leading to satisfied users and business success.

Summary

Requirements analysis is the process of examining, refining, and structuring requirements so they are clear, complete, consistent, and testable.

- ✓ Analysis refines requirements
- ✓ Ambiguity must be eliminated
- ✓ Conflicts must be identified
- ✓ Classification improves organization
- ✓ Completeness ensures coverage

References

1. Beatty, K. W. (2013). Software Requirements (3rd ed.). Washington: Microsoft Press.

Thank You!