STUDY JOURNAL - Requirements Gathering and Analysis

Author: Abner Soberon

Date: April 16, 2025

Context: Personal deep-dive study journal for professional growth as a technical AI Product Manager.

Purpose: To reinforce and expand understanding of the requirement gathering and analysis process, with practical insights and structured breakdown.

1. INTRODUCTION

The first step in building any software system is understanding what the client needs. Before writing a single line of code, a business analyst or project team must gather and analyze the requirements that define the product.

This process happens during the Analysis Phase of the Software Development Life Cycle (SDLC). This phase transforms abstract client expectations into actionable, structured, and technically feasible requirements that guide the development process.

2. REQUIREMENTS GATHERING

Requirements gathering refers to the systematic collection of needs, goals, and expectations from the client. These needs must be captured in a clear, specific, and unambiguous way to ensure the development team builds exactly what the client expects.

2.1 Methods of Gathering Requirements

- Interviews: One-on-one or group sessions where analysts ask clients direct questions.
- Questionnaires: Written forms with predefined questions. Useful when clients are remote or numerous.
- Document Searches: Analysts review existing business documents, reports, or manuals.
- Observations: Analysts shadow users and observe how work is done.
- Meetings: Structured sessions for discussing and validating requirements.

2.2 Types of Requirements

- Business: High-level goals from a business perspective.
- Functional: Specific features or functions the software must perform.
- Non-Functional: Qualitative aspects like performance, security, or usability.

3. FEASIBILITY STUDY

A feasibility study evaluates whether a proposed project should move forward. It examines technical, operational, economic, and schedule aspects to identify risks and determine viability.

3.1 Types of Feasibility

- Technical: Are resources and technology available?
- Operational: Will the product work for the client?
- Economic: Will it be cost-effective?
- Schedule: Can it be delivered on time?

4. SOFTWARE REQUIREMENTS SPECIFICATION (SRS)

An SRS document formalizes all gathered and refined requirements. It acts as a contract between the client and development team, ensuring all stakeholders agree on what will be built.

4.1 Benefits of an SRS

- Establishes shared expectations.
- Reduces development effort by avoiding misunderstandings.
- Serves as a reference for design, testing, and future enhancements.
- Supports cost and time estimations.

4.2 Characteristics of a Good SRS

- Correct
- Precise
- Complete
- Traceable

4.3 Structure of an SRS

Introduction: Purpose, Scope, Definitions, References, Overview.

General Description: Product perspective, functions, users, constraints, assumptions.

Specific Requirements:

- External Interface Requirements
- Functional and Non-Functional Requirements
- Use Cases

- Design Constraints
- Logical Database Requirements
- Other Requirements

5. ANALYZING REQUIREMENTS

Purpose:

- Clarify vague or conflicting requirements.
- Prepare the framework for design.

Two Levels of Analysis

System-Level: Broad organizational and system goals.

- Detect inconsistencies, gaps, ambiguities.
- Hold meetings.
- Document and resolve open issues.
- Create supporting prototypes if necessary.

Software-Level: Specific product design.

- Define architecture, performance, data models, and behavior.
- Drives system design.

6. INSIGHT: LOGICAL PROBLEM-SOLVING

Sometimes, a problem contains the seeds of its own solution.

In one exercise, by analyzing the phrasing of each statement and identifying keywords (e.g., financial -> economic, resources -> technical, time -> schedule), I was able to deduce the correct matches without relying solely on memory.

This technique can be generalized to many real-world problems: if we read carefully, analyze logically, and break the elements down, the solution often becomes self-evident.

7. FINAL REFLECTION

This lesson not only reinforced key concepts in requirements engineering but also strengthened my analytical thinking and problem-solving discipline. I learned to approach software planning from both a human and

technical perspective, using tools, logic, and precision.

As a future AI Product Manager, mastering requirement gathering and analysis equips me to lead teams, manage stakeholders, and build meaningful, high-impact systems.