Detailed Analysis of Requirements in Software Engineering

Understanding different types of requirements is essential for creating effective and successful software. Requirements define what the system needs to do and how it should perform. Let's analyze each type of requirement in detail, their characteristics, and how they apply to real-world software projects.

---

1. Business Requirements

Definition:

Business requirements explain the *why* of the system — they define the high-level goals, objectives, and outcomes the business seeks to achieve through the system. These are broad statements, outlining the overall purpose of the system without going into specifics about how the system will operate.

When to Use:

- At the start of a project to capture the organization's or customer's key objectives.
- During the initial project scoping phase to set the direction and vision.

What They Include:

- High-level goals of the project.
- Benefits the system will provide to the business.
- Scope of the system (what the system will cover).

Example:

A **parcel delivery system** might have a business requirement to **"improve the efficiency of parcel tracking and customer satisfaction through real-time updates"**.

Strengths:

- Focuses on strategic goals, ensuring the software aligns with business needs.
- Helps stakeholders agree on project vision.

Limitations:

- Business requirements can be vague and may not address technical details, which can lead to misinterpretations.

Case Study Application: Hospital Management System

The business requirement for a hospital management system could be:

"Improve patient care and reduce administrative overhead by integrating patient records, billing, and scheduling into a single, unified system."

---

2. User Requirements

Definition:

User requirements describe what the system should do from the perspective of the end-user. They provide more detailed descriptions of the system's functions, focusing on what users expect from the system.

When to Use:

- After business requirements are established, user requirements define how users will interact with the system.
- During requirement gathering, user stories and feedback sessions.

What They Include:

- Descriptions of specific tasks that users need the system to perform.
- Functionalities users expect to use regularly.
- Usability and interface expectations.

Example:
For the same **parcel delivery system**, a user requirement might be:
"The system should allow users to track the status of their package in real-time through a web interface."
Strengths:

- Captures the needs of actual users, ensuring the system is user-friendly.
- Provides a detailed understanding of what users want, improving system design.

Limitations:

- May be incomplete or ambiguous if users are unsure of their needs.
- Can lead to scope creep if too many features are requested.

Case Study Application: Hospital Management System
A user requirement for a hospital system could be:
"Doctors should be able to view a patient's complete medical history, including previous treatments and medications, at the click of a button."

---

3. Functional Requirements

Definition:
Functional requirements define *what the system should do*—they describe specific functionalities or operations the system must be able to perform. These are detailed, technical requirements aimed at developers and designers.
When to Use:

- After defining user requirements, functional requirements specify how those user needs will be met.
- During system design and development.

What They Include:

- Specific inputs, outputs, and behaviors of the system.
- Processes the system must carry out in response to various inputs.
- How the system interacts with users and other systems.

Example:
In the **parcel delivery system**, a functional requirement could be:
"The system must allow users to input a tracking number, search the database for matching records, and display the current status of the package."
Strengths:

- Provides clear and specific instructions for developers, leading to better system implementation.
- Ensures that the system functions as expected.

Limitations:

- Overly detailed functional requirements can restrict flexibility during development.
- If not aligned with user needs, they can result in a system that technically works but is not user-friendly.

Case Study Application: Hospital Management System
A functional requirement might be:
"The system shall automatically generate a list of patient appointments for each doctor every morning."

---

4. Non-Functional Requirements

Definition:

Non-functional requirements define *how the system should perform* rather than what it should do. These include performance, security, usability, reliability, and other attributes that describe the system's quality and behavior.

When to Use:

- During the design and architectural phases to define constraints on system operations.
- In system testing to ensure the software performs as required under different conditions.

What They Include:

- Performance metrics (e.g., speed, response time).
- Usability and accessibility.
- Security protocols and privacy policies.
- Reliability (e.g., uptime and failure recovery).

Example:

For the **parcel delivery system**, a non-functional requirement might be:

"The system should process package tracking requests within 2 seconds for 95% of all users."

Strengths:

- Ensures the system meets quality standards and user expectations beyond basic functionality.
- Helps define system performance benchmarks.

Limitations:

- Non-functional requirements can be difficult to quantify and test.
- If poorly defined, they can lead to performance issues that are not detected until after deployment.

Case Study Application: Hospital Management System

A non-functional requirement might be:

"The system shall ensure that patient data is encrypted in transit and at rest to comply with privacy regulations."

---

5. Levels of Requirements

Definition:

Requirements can be classified into different levels, typically **Business**, **User**, **System**, and **Functional/Non-Functional**. Each level represents a different granularity of detail, helping to target different audiences.

When to Use:

- Throughout the software development life cycle to communicate requirements at various stages.
- During project planning to ensure that stakeholders at all levels understand the project's scope and goals.

What They Include:

- **Business Requirements:** High-level goals and objectives.
- **User Requirements:** Functional needs from a user's perspective.
- **System Requirements:** Detailed descriptions of system operations, interactions, and constraints.
- **Functional and Non-Functional Requirements:** Detailed behaviors and performance measures.

Example:

For the **parcel delivery system**, the levels could include:

- **Business Requirement:** Improve customer experience.
- **User Requirement:** Provide real-time tracking.
- **Functional Requirement:** Implement a search function for tracking numbers.
- **Non-Functional Requirement:** Ensure 99.9% uptime.

Strengths:

- Helps manage complexity by breaking down requirements for different audiences.
- Ensures that every aspect of the system, from goals to technical details, is covered.

Limitations:

- Miscommunication between different levels of stakeholders can lead to incomplete or conflicting requirements.
- Overly granular levels can make the project cumbersome to manage.

Case Study Application: Hospital Management System

Levels of Requirements:

- **Business Requirement:** Reduce hospital administration time by 20%.
- **User Requirement:** Doctors can access medical histories instantly.
- **Functional Requirement:** Create a daily report of patient appointments.
- **Non-Functional Requirement:** System should have 99.5% uptime with a response time under 3 seconds.

Case Study: Hospital Management System

Application of Business, User, Functional, and Non-Functional Requirements

- **Business Requirement:**
  The hospital's management wants a system that reduces patient record retrieval time and improves overall patient care efficiency.
- **User Requirements:**
  - Doctors should be able to access patient records in real-time.
  - Nurses should manage appointment schedules.
  - Administrators should be able to generate reports quickly.
- **Functional Requirements:**
  - The system shall store patient records in a central database.
  - The system shall allow doctors to filter records by date, treatment, and medication.
  - The system shall automatically notify nurses of upcoming appointments.
- **Non-Functional Requirements:**
  - The system must be available 24/7 with an uptime of 99.9%.
  - All patient data must be encrypted.
  - The system should load patient records within 2 seconds.

Conclusion

In software development, understanding different types of requirements is crucial for building a successful product. Business requirements ensure the system aligns with organizational goals, while user, functional, and non-functional requirements provide the details needed to design, implement, and test the system effectively. Managing these different levels of requirements helps ensure that the system is not only functional but also meets performance and user expectations.