**Key Topics:**

1. **Requirements Specification Document**
2. **Managing Requirements**
3. **Attributes of a Good Requirements Specification**
4. **Requirements Analysis vs. Validation**
5. **UML Class Diagram**

**1. Requirements Specification Document**

**IEEE Standard Template:**

1. **Introduction:**
   * **Purpose of the system:** Describe why the system is being developed.
   * **Scope of the system:** Define the system boundaries and major functions.
   * **Objectives and success criteria:** Outline the goals and how success will be measured.
   * **Definitions, acronyms, and abbreviations:** Provide a glossary.
   * **References:** List related documents.
   * **Overview:** Summarize the document structure.
2. **Current System:** Describe the existing system (if any).
3. **Proposed System:**
   * **Overview:** Briefly introduce the new system.
   * **Functional Requirements:** Detailed list of functionalities.
   * **Non-functional Requirements:** Metrics for usability, reliability, performance, supportability, implementation, interface, packaging, and legal considerations.
   * **System Models:** Include scenarios, UML use case models, object models, dynamic models, and user interface mock-ups.
4. **Glossary:** Define all technical terms and abbreviations used in the document.

Define requirement focus on PEGS (project environment goals system)

**1. Project Book (P)**

The Project Book focuses on the management aspects of the project. It includes:

* **P.1 Roles:** Defines the roles and responsibilities of all team members and stakeholders involved in the project.
* **P.2 Personnel characteristics and constraints:** Describes the skills, expertise, and limitations of the personnel working on the project.
* **P.3 Imposed technical choices:** Lists any mandatory technical decisions, such as programming languages, frameworks, or tools that must be used.
* **P.4 Schedule and milestones:** Provides a timeline for the project, including key milestones and deadlines.
* **P.5 Tasks and deliverables:** Details the specific tasks to be completed and the deliverables to be produced throughout the project.
* **P.6 Risks and mitigation analysis:** Identifies potential risks to the project and outlines strategies to mitigate them.
* **P.7 Requirements process and report:** Describes the process for gathering and managing requirements and the format for reporting them.

**2. Environment Book (E)**

The Environment Book addresses the context in which the system operates. It includes:

* **E.1 Glossary:** Defines key terms and concepts used in the project to ensure common understanding.
* **E.2 Components:** Lists the external systems, hardware, software, and other components that interact with the system.
* **E.3 Constraints:** Identifies any environmental or external constraints that impact the system, such as regulations, standards, or physical limitations.
* **E.4 Assumptions:** States any assumptions made during the project planning and design.
* **E.5 Effects:** Describes the potential effects or impacts of the system on its environment.
* **E.5 Invariants:** Lists any invariants or conditions that must always hold true for the system.

**3. Goals Book (G)**

The Goals Book outlines the overarching objectives and context of the project. It includes:

* **G.1 Overall context & goals:** Describes the broader context and primary goals of the project.
* **G.2 Current situation:** Provides an analysis of the current state of the system or problem domain.
* **G.3 Expected benefits:** Details the anticipated benefits and improvements resulting from the project.
* **G.4 System overview:** Offers a high-level overview of the system, including its purpose and scope.
* **G.5 Limitations and exclusions:** Identifies any limitations of the system and areas that are explicitly excluded from the project.
* **G.6 Stakeholders:** Lists all stakeholders involved in the project and their interests.
* **G.7 Requirements sources:** Describes the sources from which the requirements are derived, such as user interviews, surveys, or existing documentation.

**4. System Book (S)**

The System Book focuses on the technical details of the system itself. It includes:

* **S.1 Components:** Describes the internal components of the system and their interactions.
* **S.2 Functionality:** Lists the specific functions and features that the system must provide.
* **S.3 Interfaces:** Defines the interfaces between the system and its users, as well as between internal components.
* **S.4 Scenarios (use cases, user stories):** Provides detailed scenarios, use cases, or user stories that illustrate how the system will be used.
* **S.5 Prioritization:** Prioritizes the system requirements based on their importance and impact.
* **S.6 Verification and acceptance criteria:** Describes the criteria and methods for verifying that the system meets its requirements and for accepting the final product.

**2. Managing Requirements**

**Project vs. Goals:**

* **Project Requirements:** Focus on how the software development process will be carried out.
  + *Example*: "The system will be implemented using React and Express."
  + *Example*: "The system will be delivered in 3 sprints, with the final deliverable on 31 May 2021."
* **Goals Requirements:** Capture the business objectives.
  + *Example*: "The system will enable admin staff to process 4 student applications per hour instead of 2 hours per application now."

**Requirements Sources:**

* Requirements can come from various stakeholders, documents, and user feedback.

**Attributes of a Good Requirements Specification:**

* **Concise:** Avoid unnecessary details.
* **Complete:** Include all necessary requirements.
* **Unambiguous:** Use clear and precise language.
* **Testable:** Ensure each requirement can be validated.
* **Consistent:** Avoid conflicting requirements.
* **Feasible:** Ensure requirements are achievable.
* **Modifiable:** Allow for easy updates.
* **Traceable:** Link requirements to their origins.
* **Specifies External Behaviour:** Focus on what the system should do, not how.
* **Specifies Constraints:** Detail any limitations on the implementation.
* **Easy to Change:** Simplify modifications.
* **Reference Tool:** Serve as a reference throughout the project lifecycle.
* **Life Cycle Considerations:** Plan for long-term maintenance and updates.
* **Handles Undesired Events:** Outline acceptable responses to failures or errors.

**3. Requirements Analysis vs. Validation**

**Analysis:**

* Works with raw requirements gathered from stakeholders.
* Key Question: “Have we got the right requirements?”

**Validation:**

* Works with a final draft of the requirements document.
* Key Question: “Have we got the requirements right?”

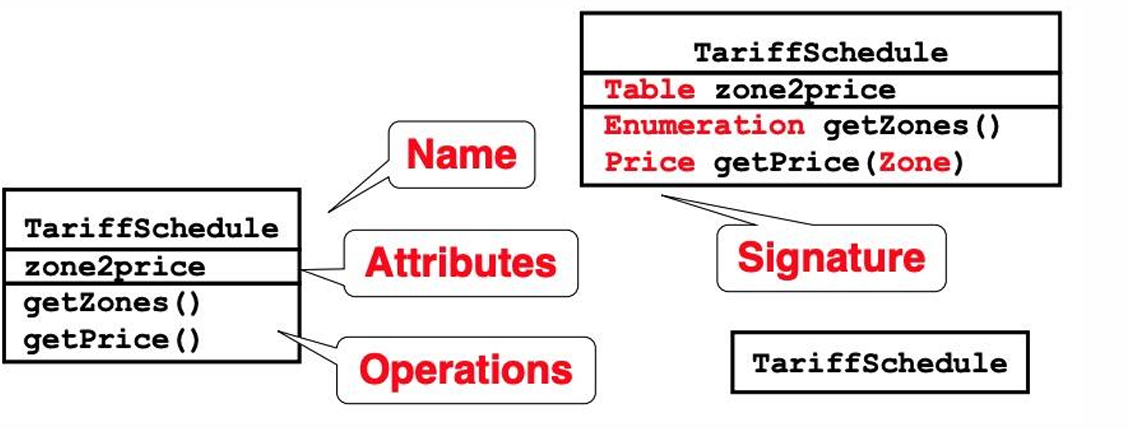
**4. UML Class Diagram**

**Definition:**

* Describes types of objects in the system and their static relationships.
* Shows properties and operations of a class and constraints on object connections.

**Components:**

* **Classes:** Templates for objects with attributes (state) and operations (behavior).
* **Objects:** Instances of classes.
* **Relationships:**
  + **Associations:** Interactions between objects.
  + **Dependency:** Temporary links between objects.
  + **Aggregation:** One object contains references to others.
  + **Composition:** "Whole-part" relationships where one object is composed of others.



**Multiplicity:**

* Indicates how many objects can be involved in a relationship.
  + *Example*: 1 (An order must have exactly one customer)
  + *Example*: 0..1 (A corporate customer may or may not have a sales rep)
  + *Example*: \* (A customer can have zero or more orders)

**Example:**

* **Class Diagram Interpretation:**
  + *Customer* has zero or more *Orders*, each *Order* has one *Customer*.
  + *Order* contains one or more *Order Lines*.
  + *Order Lines* refer to a *Product*.
  + *Corporate Customer* and *Personal Customer* are subtypes of *Customer*.
  + *Corporate Customer* may have one *Sales Rep*.

public class Customer {

private List<Order> orders;

}

public class Order {

private Customer customer;

private List<OrderLine> orderLines;

}

public class OrderLine {

private Product product;

}

public class CorporateCustomer extends Customer {

private SalesRep salesRep;

}

public class SalesRep {

// SalesRep details

}

**5. Prototyping**

**Why Prototype?**

* Validates assumptions and elicits new requirements.
* Provides a tangible representation of the system.

**Types of Prototypes:**

* **Software Prototypes:** Executable using rapid prototyping tools.
* **Mock-ups:** Screen designs or paper sketches.

**Benefits:**

* Clarifies requirements.
* Facilitates stakeholder feedback.

**Detailed Examples for Equations and Metrics**

**Example: Performance Requirement Test**

* *Requirement*: The system shall respond to user requests in < 1 second under normal load (<100 users).
* *Test Method*:
  1. Create a script to simulate 100 concurrent user requests.
  2. Measure response times for each request.
  3. Ensure all requests are processed within 100 seconds.

**Example: Memory Usage Test**

* *Requirement*: The app must use less than 200 MB of memory on Android.
* *Test Method*:
  1. Build the app for different Android versions.
  2. Install the app on various devices/emulators.
  3. Use a memory profiling tool to measure memory usage.
  4. Verify that memory usage is below 200 MB.