Course: Software Requirements Engineering

Week 9: Specification & SRS Structure

Lecturer: Yimer Amedie (MSc.)

Addis Ababa Science and Technology University

May, 2026

Contents

- Introduction
- Requirements Specifications and SRS
- Principles of Specification
- Structure and Components of SRS
- Standards for Requirements Documentation

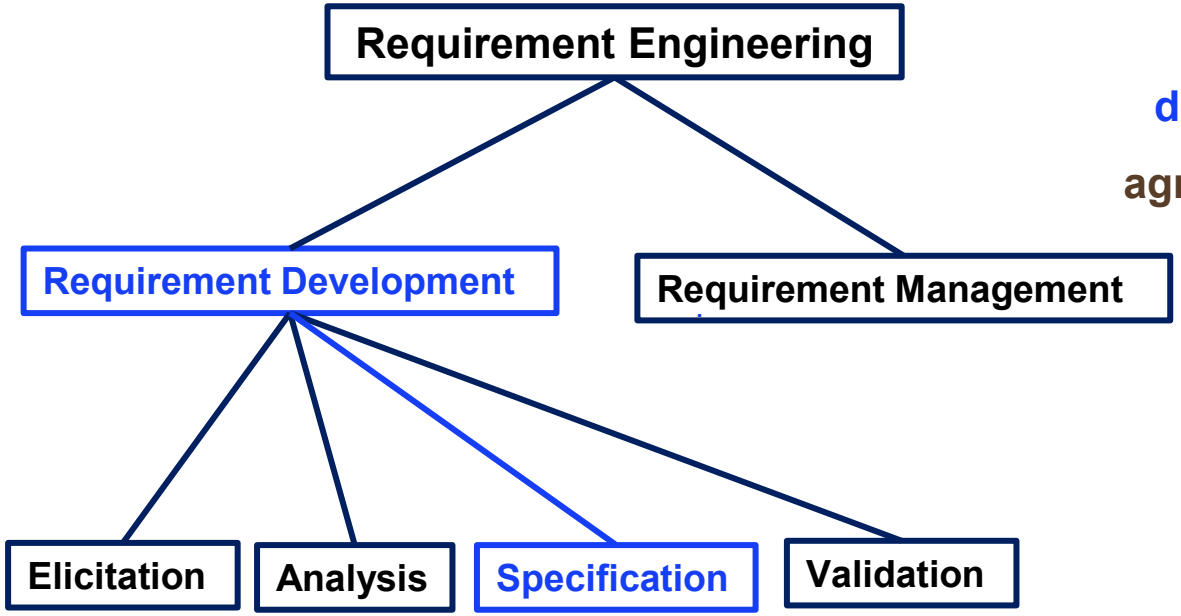
Figure 1. *The software requirements*
Note. Image generated using Sora by OpenAI (2026).

Learning Outcomes

After completing this lesson, you will be able to:

- Explain the purpose and importance SRS
- Identify the standard structure and components of an SRS document
- Apply IEEE/ISO standards in requirements specification & documentation
- Organize and document requirements using proper structure

Introduction



The result of requirements development is a documented agreement among stakeholders about the product to be built (Beatty, 2013)

Documented agreement ➔ SRS

What is Specification?

A precise, structured description of system requirements that defines what the system must achieve, without prescribing how it is implemented.

Not:

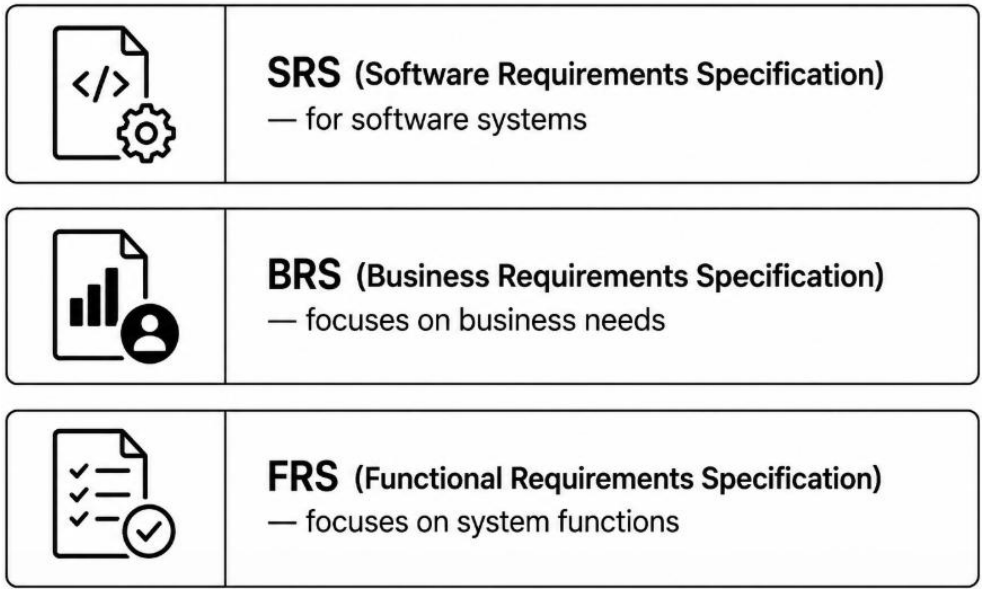
- design decisions (how to implement)
- coding details
- technology choices

Example

Design (not specification):	Specification:
"The system shall use React for the UI"	"The system shall provide a user interface for submitting and tracking requests"

Requirement Specification (RS)

- ✓ **RS** is a detailed document that clearly describes what a system, software, or product should do and the constraints under which it must operate.



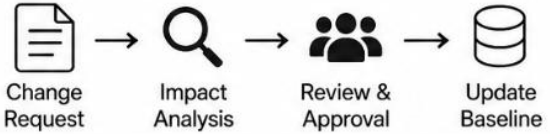
Overview of SRS

Formal document describing functional and non-functional requirements

Single source of truth for all project participants

Used for validation, verification, and maintenance planning

Baseline for change control management

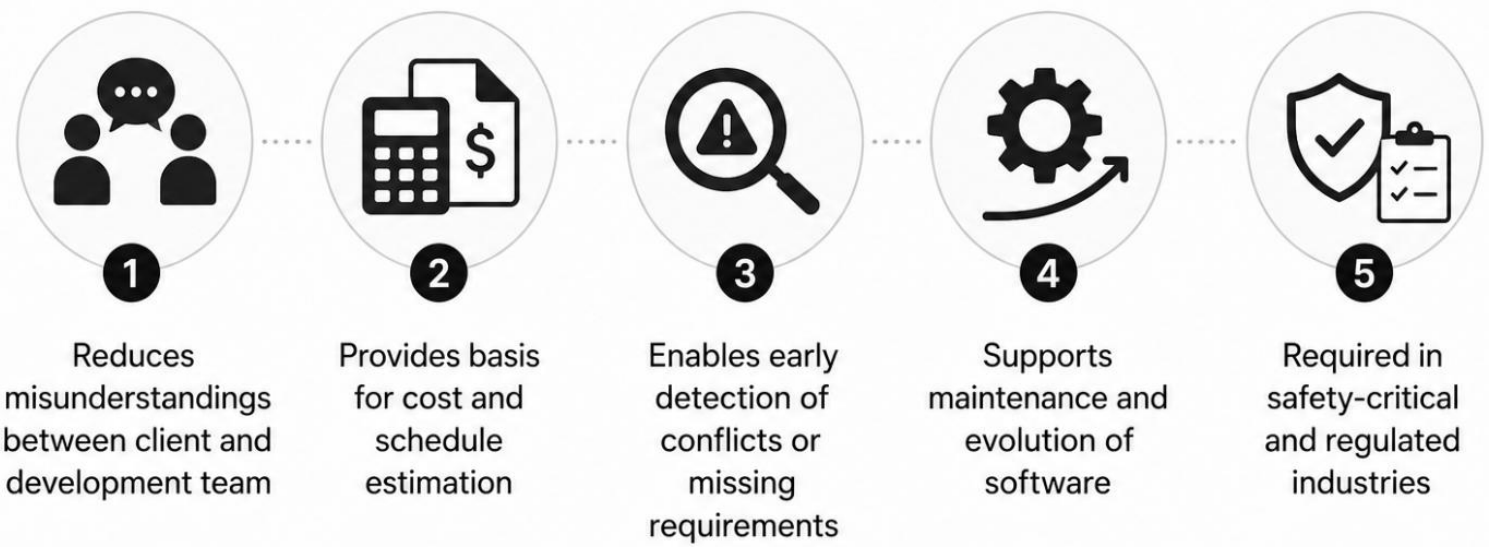


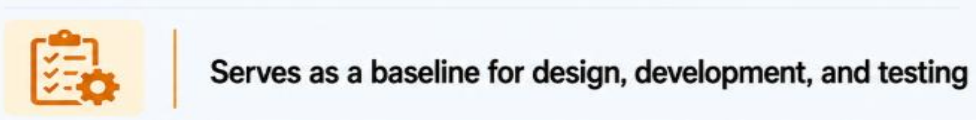
Distinction from business requirements and design specs

Serve as a contract between stakeholders and developers.

Provides a common understanding and agreement on what is to be built.

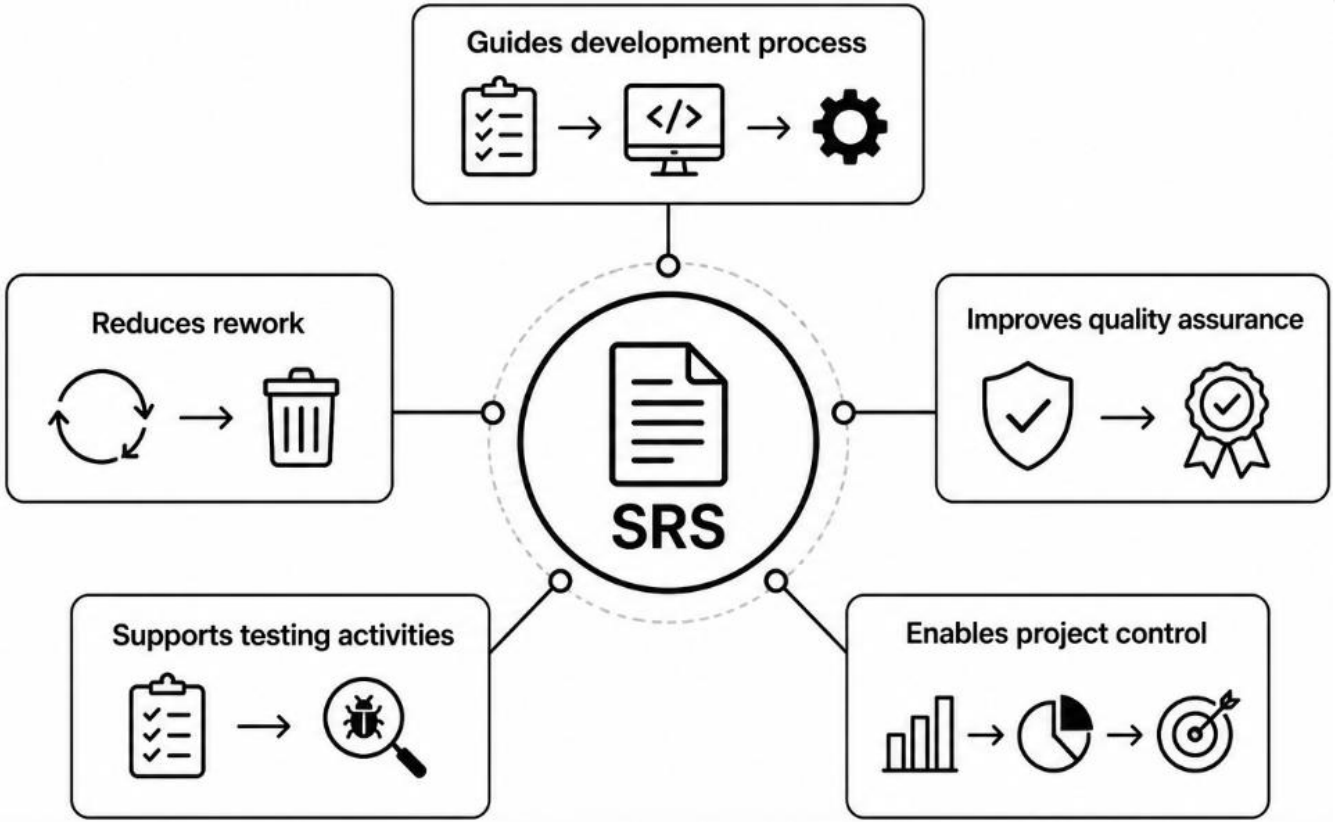
Why is it Important?





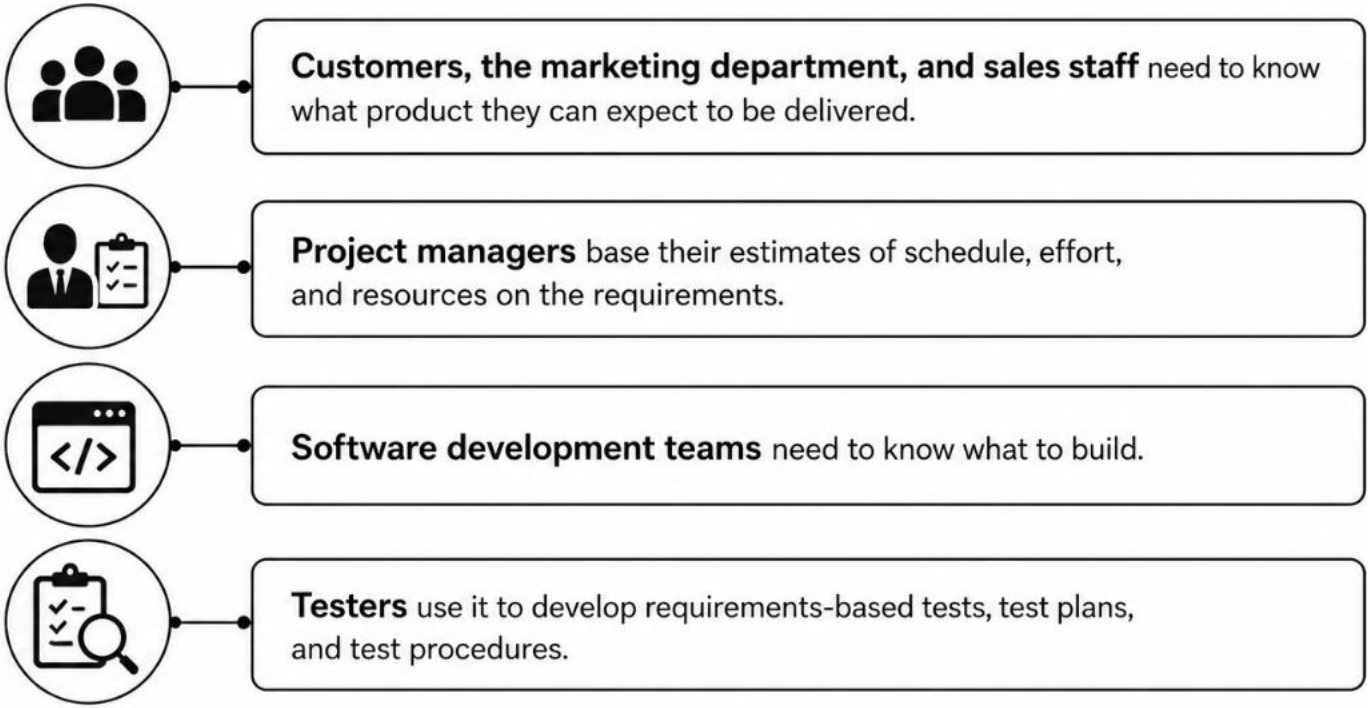
SRS focuses on **translating business needs into system requirements**

Importance of SRS in SDLC



Stakeholders of SRS

Numerous audiences rely on the SRS (Beatty, 2013).



Stakeholders of SRS … cont'd

Maintenance and support staff use it to understand what each part of the product is supposed to do.

Documentation writers base user manuals and help screens on the SRS and the user interface design.

Training personnel use the SRS and user documentation to develop educational materials.

Legal staff ensures that the requirements comply with applicable laws and regulations.

Subcontractors base their work on—and can be legally held to—the specified requirements.

Principles of SRS

Separate correct requirements from incorrect ones

Ensure clarity and remove ambiguity in requirements

Develop a complete description of system behavior

Maintain consistency throughout the specification

Make requirements verifiable

Rank requirements based on importance & stability

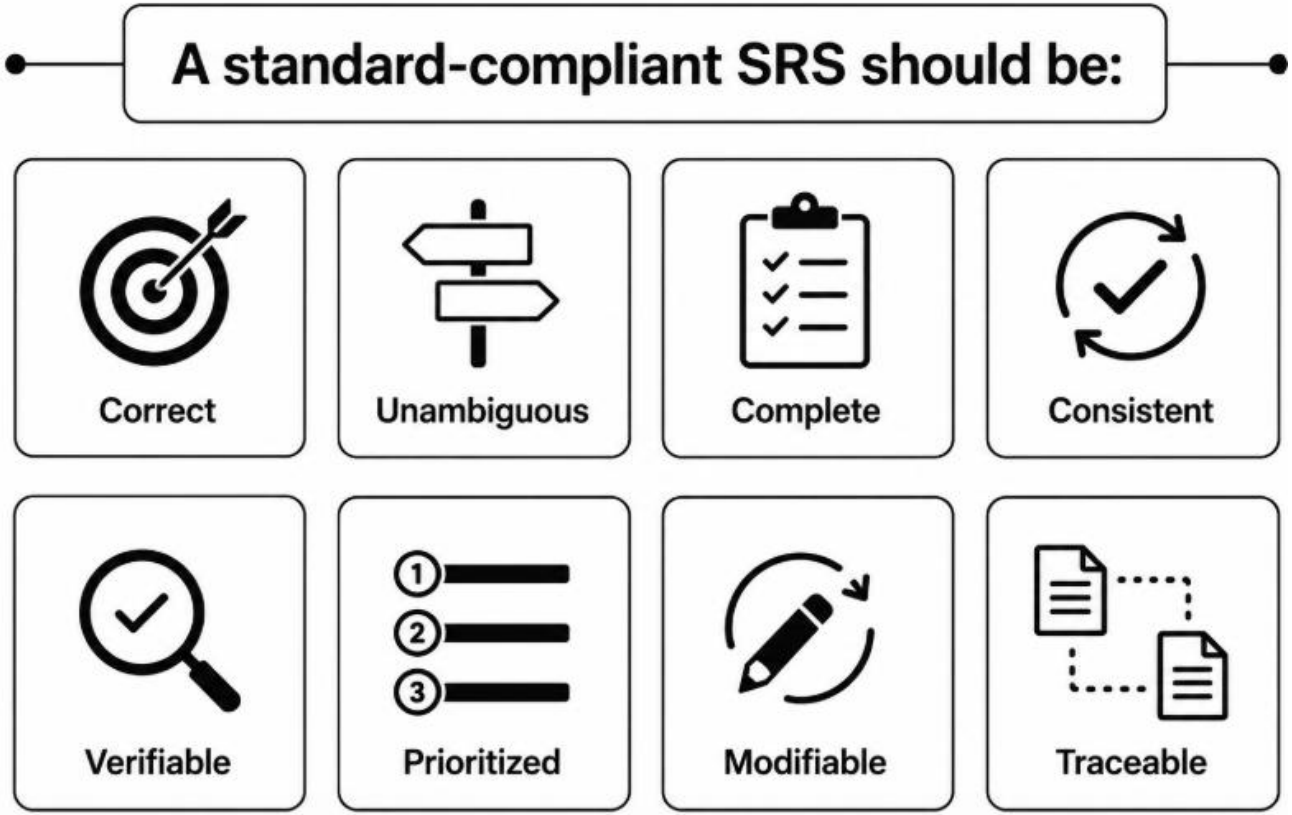
Structure the specification to be modifiable

Ensure traceability of requirements

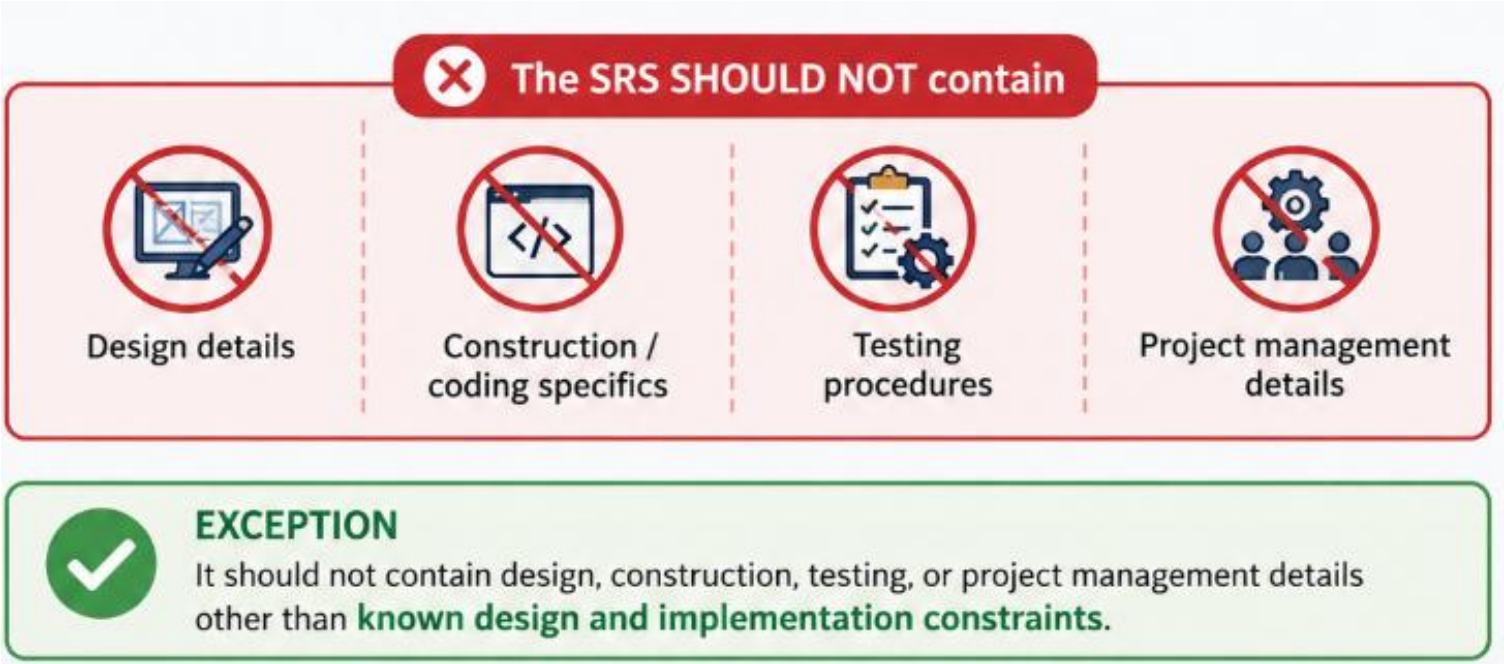
Keep the specification independent of design decisions

Make the specification adaptable to change and evolution

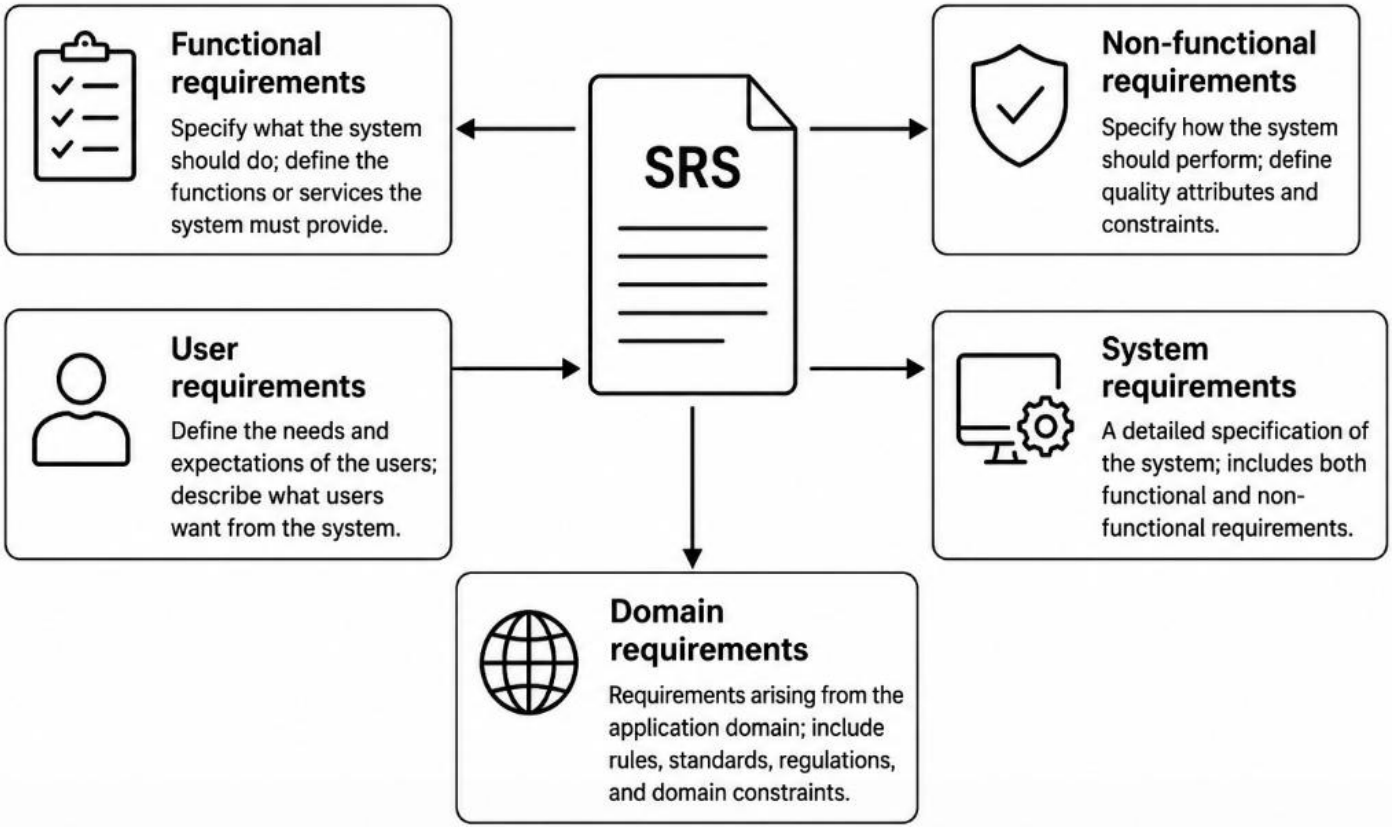
Principles of SRS – Final Remark



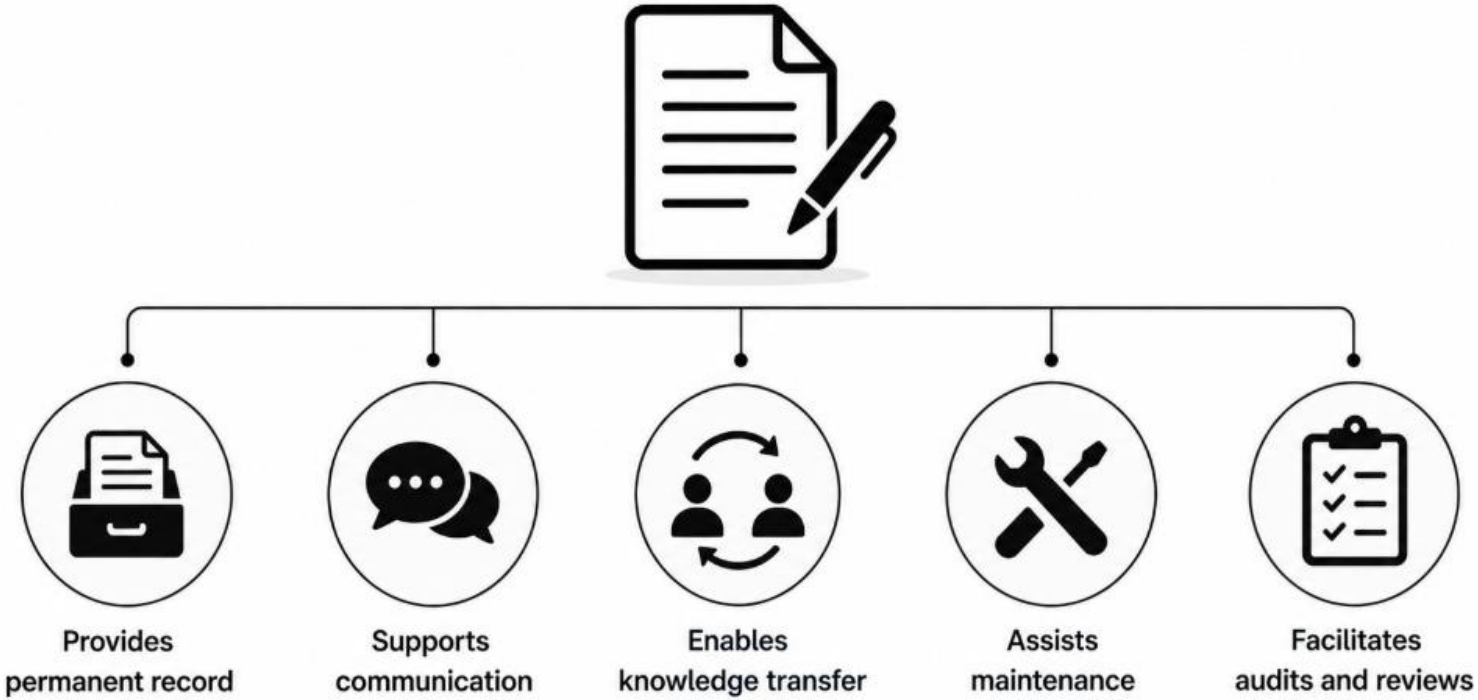
What is not Included in SRS



Types of Requirements in SRS



Role of Documentation



Maintaining SRS

Regular updates

The SRS is reviewed and updated at regular intervals to incorporate new information and changes.

Reflects changes

All changes in requirements, business rules, or stakeholder needs are accurately captured and reflected in the SRS.

Ensures accuracy

Updated SRS remains correct, consistent, and complete, ensuring reliable communication.

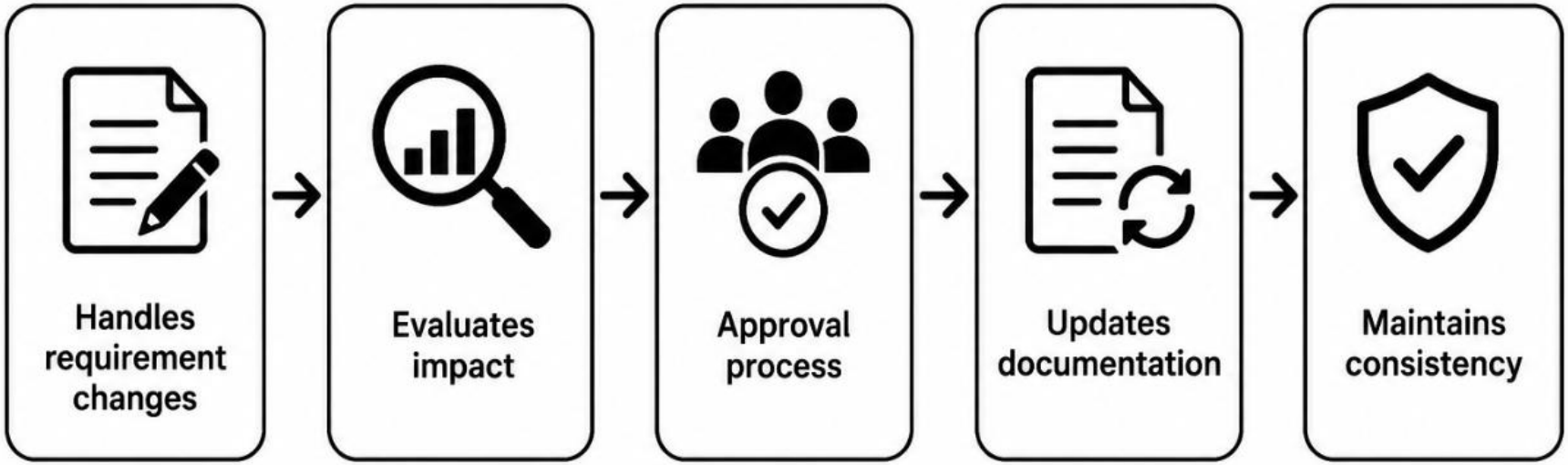
Supports evolution

The SRS evolves along with the system and business environment to support future enhancements.

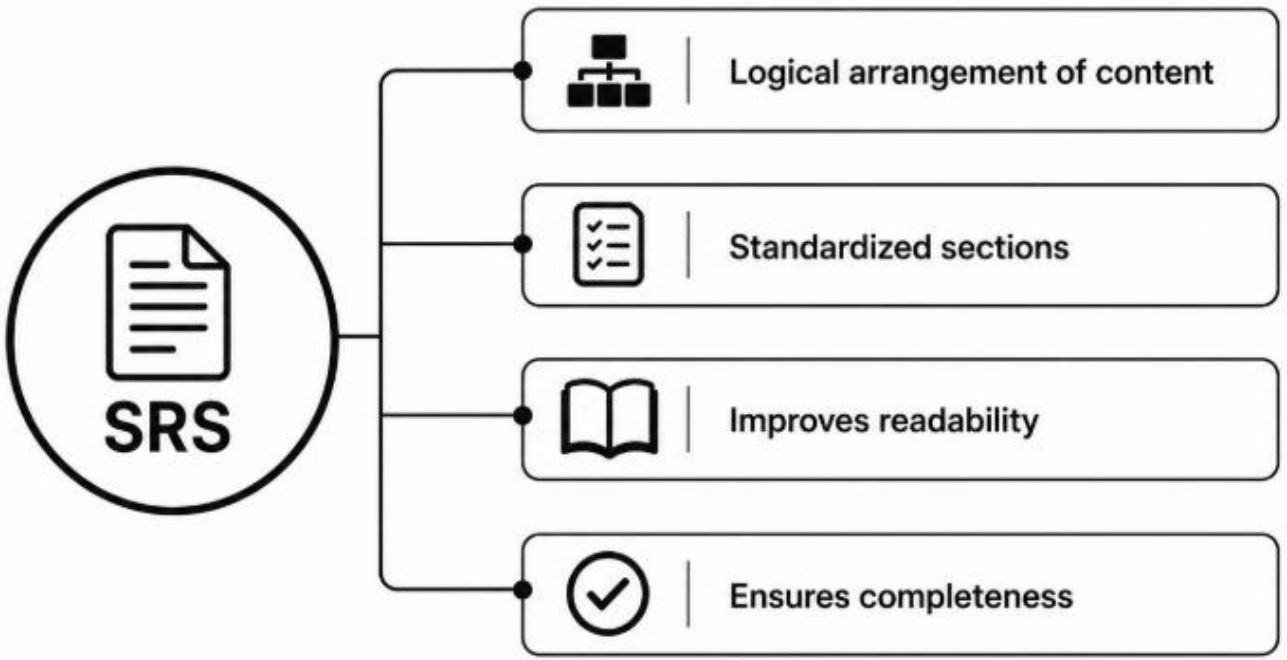
Continuous improvement

Version Control in SRS

Change Management in SRS



SRS Structure



Requirements Document Standard

Common Standards

1 IEEE 830 (Classic Standard)

- Widely used traditional SRS format
- Defines structure and qualities of good SRS

2 ISO/IEC/IEEE 29148 (Modern Standard)

- Updated and more comprehensive
- **Covers:**
  - Requirement engineering processes
  - Documentation standards
  - Lifecycle integration

Note: ISO/IEC/IEEE 29148 is the current industry-recommended standard as it supports the full requirements lifecycle.

IEEE Std 830 Vs. ISO/IEC/IEEE 29148

IEEE 830-1998 Historical Standard for SRS Structure	vs.	ISO/IEC/IEEE 29148:2011 (Revised 2018) Supersedes 830
IEEE 830-1998: historical standard for SRS structure		ISO/IEC/IEEE 29148:2011 (revised 2018) supersedes 830
Focused on software requirements specification (SRS) structure		29148 integrates system and software requirements engineering
Provides a structured outline for Software Requirements Specification		29148 adds annexes for writing techniques and quality indicators

Both standards emphasize →

- ✓ Completeness
- ✓ Consistency
- ✓ Unambiguity
- ✓ Verifiability

Generic SRS Structure (Clause-by-Clause)

Section 1:
Introduction

Purpose, Scope
Definitions, References

Section 2:
Overall description

User characteristics, Constraints
Assumptions & dependencies, Product perspectives
Summary of product features

Section 3:
Specific requirements

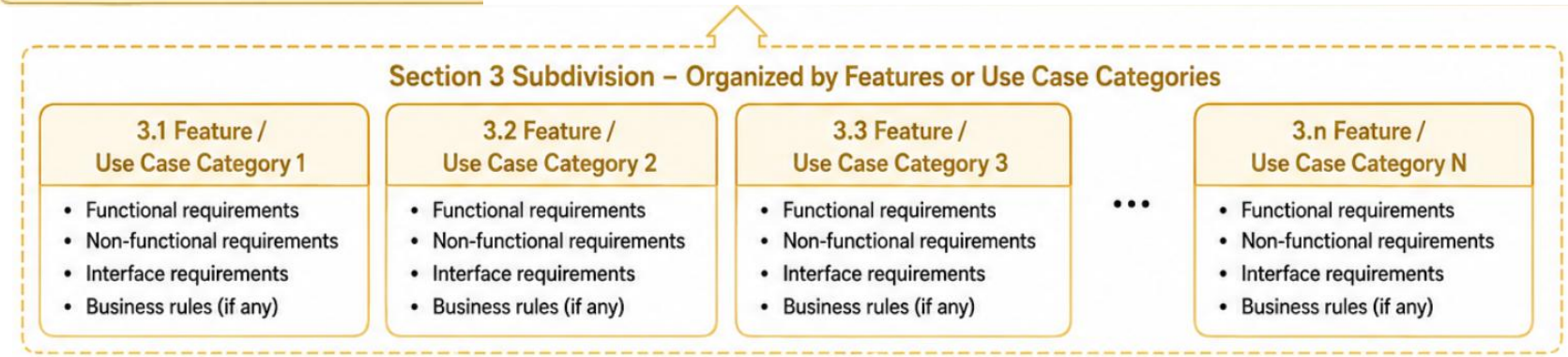
FRs, NFRs
Interface requirements

Section 4:
Appendices and index

Appendices, Index

Generic SRS Structure … cont'd

Functional requirements: what the system shall do
Non-functional requirements: quality attributes and constraints
Interface requirements: user, hardware, software, communication interfaces



Standard SRS Structure

A standard SRS document is typically structured as follows:

1. Introduction
- Purpose of the document
- Scope of the system
- Definitions, acronyms, abbreviations
- References

2. Overall Description
- Product perspective
- Product functions
- User classes and characteristics
- Operating environment
- Constraints
- Assumptions and dependencies

3. System Features (Functional Requirements)
- Detailed description of system functionalities
- Organized by features/modules

4. External Interface Requirements
- User interfaces (UI)
- Hardware interfaces
- Software interfaces
- Communication interfaces

5. Non-Functional Requirements
- Performance
- Security
- Usability
- Reliability
- Scalability

6. Other Requirements
- Legal requirements
- Regulatory compliance
- Business rules

Why Use Standards?

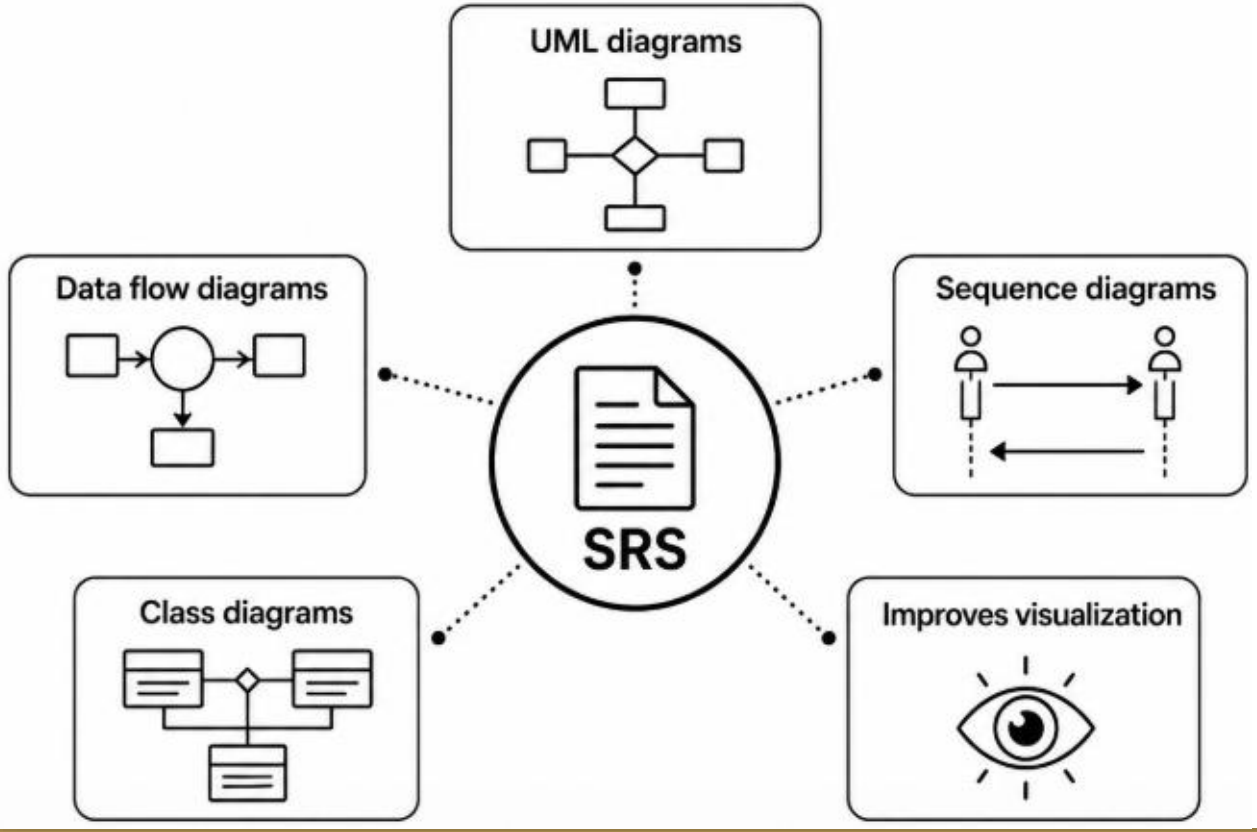
Ensures consistency

Improves communication

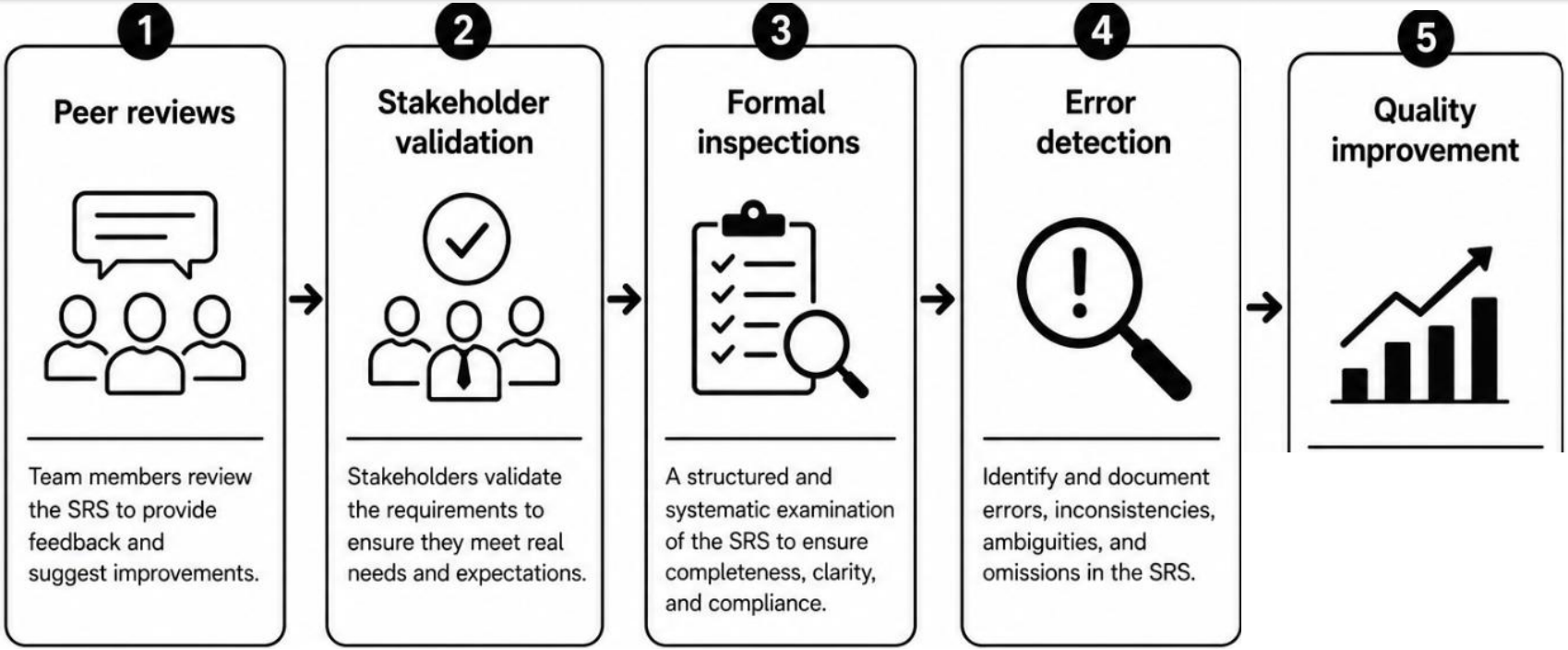
Facilitates review and validation

Supports compliance and auditing

Use of Diagrams in SRS



Review Process of SRS



Review typically done using **checklists**

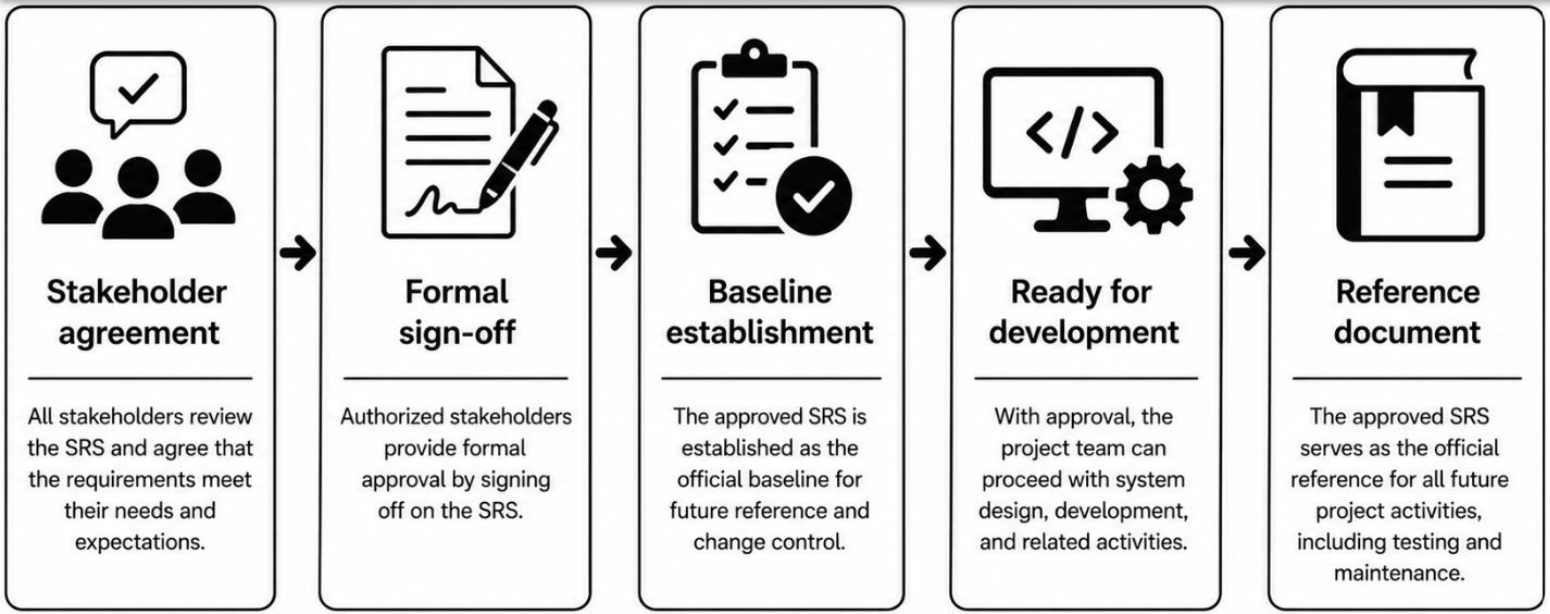
Sample SRS Checklists

Category	Checklist Questions
Correctness	Do requirements reflect actual stakeholder needs?
	Are all requirements necessary and valid?
Clarity	Are requirements clear and unambiguous?
	Are vague terms avoided or properly defined?
Completeness	Are all functional requirements included?
	Are non-functional requirements specified?
	Are inputs, outputs, and error conditions defined?
Consistency	Are there any conflicting requirements?
	Is terminology used consistently?
Verifiability	Can each requirement be tested or measured?
	Are acceptance criteria defined?
Feasibility	Are requirements realistic within constraints (time, budget, technology)?

Sample SRS Checklists … cont'd

Category	Checklist Questions
Prioritization	Are requirements ranked by importance and stability?
Modifiability	Is the document well-structured and easy to modify?
	Is redundancy minimized?
Traceability	Does each requirement have a unique ID?
	Can requirements be traced backward and forward?
Design Independence	Does the SRS avoid implementation details?
Interfaces	Are external interfaces (user, hardware, software) defined?
Environment	Is the operating environment clearly specified?
Overall Quality	Is the document easy to read and well-organized?
	Has the SRS been reviewed and approved?

Approval of SRS



SRS as a Contract

Figure 2. *SRS as a contract*
Note. Image generated using ChatGPT by OpenAI (2026).

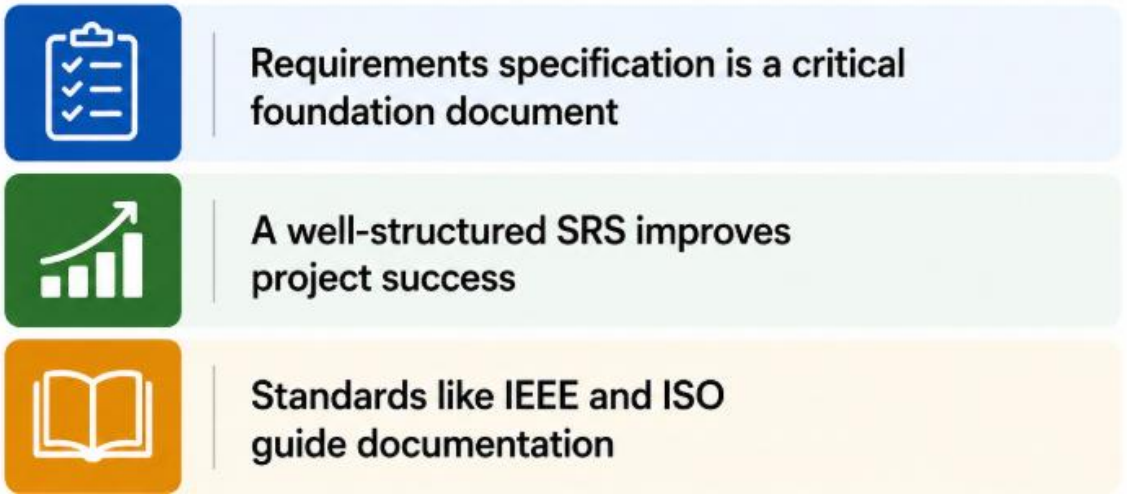
SRS as a Contract … cont'd

Common Mistakes in SRS

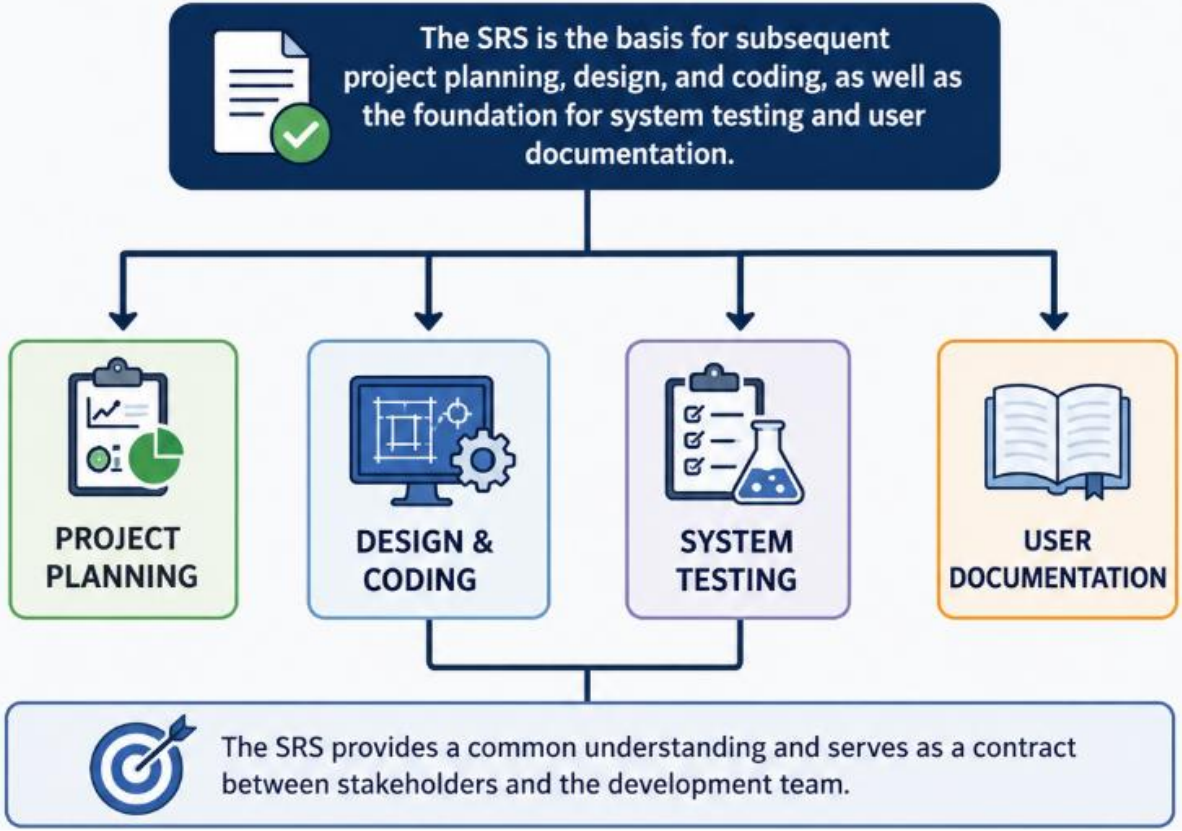
- ≠ Ambiguous and incomplete requirements
- ≠ Inconsistencies between requirements
- ≠ Design solutions included instead of actual requirements
- ≠ Lack of traceability between requirements, design, testing & implementation
- ≠ Poor organization and overly long paragraphs.
- ≠ Missing glossary
- ≠ FRs and NFRs mixed without clear labeling
- ≠ Lack of unique identifiers

Summary

- ✓ If a desired capability or quality doesn't appear somewhere in the requirements agreement, no one should expect it to appear in the product.



Summary


The SRS is the basis for subsequent project planning, design, and coding, as well as the foundation for system testing and user documentation.
PROJECT PLANNING
DESIGN & CODING
SYSTEM TESTING
USER DOCUMENTATION
The SRS provides a common understanding and serves as a contract between stakeholders and the development team.

References

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

Thank You!