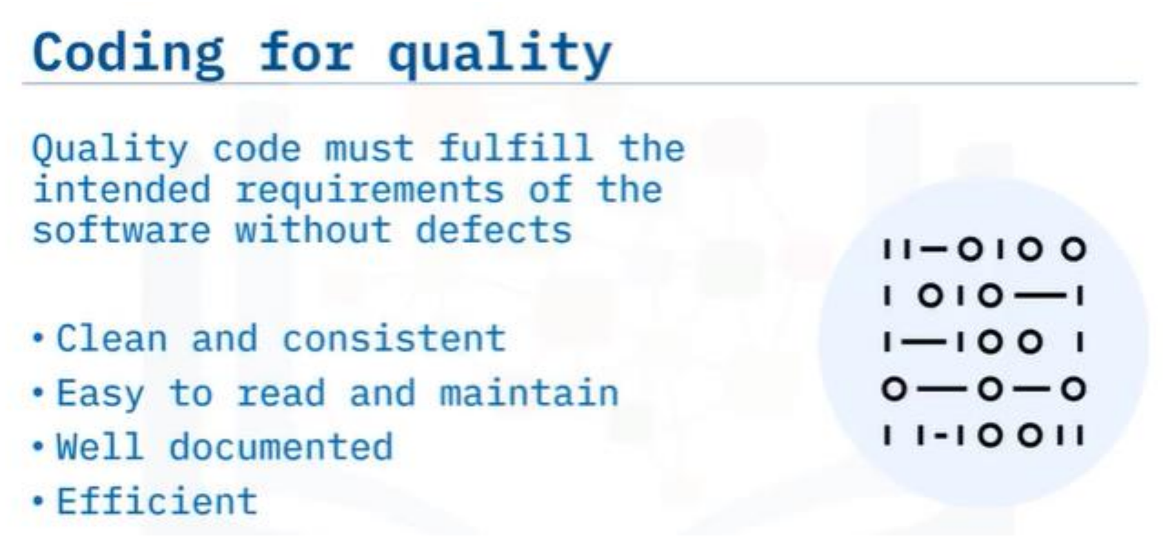
Design

- Transforming requirements into code
- Breaking down requirements into sets of related components
- Communicating business rules and application logic

Coding for quality

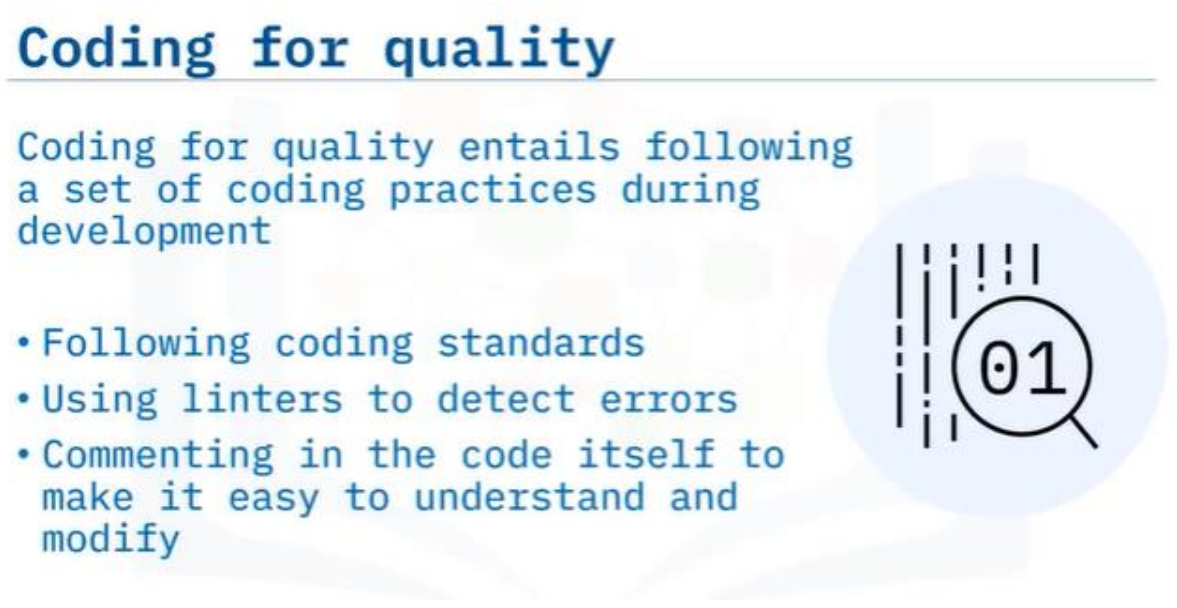
✓ Maintainability
✓ Readability
✓ Testability
✓ Security



Coding for quality

Quality code must fulfill the intended requirements of the software without defects

- Clean and consistent
- Easy to read and maintain
- Well documented
- Efficient



Coding for quality

Coding for quality entails following a set of coding practices during development

- Following coding standards
- Using linters to detect errors
- Commenting in the code itself to make it easy to understand and modify

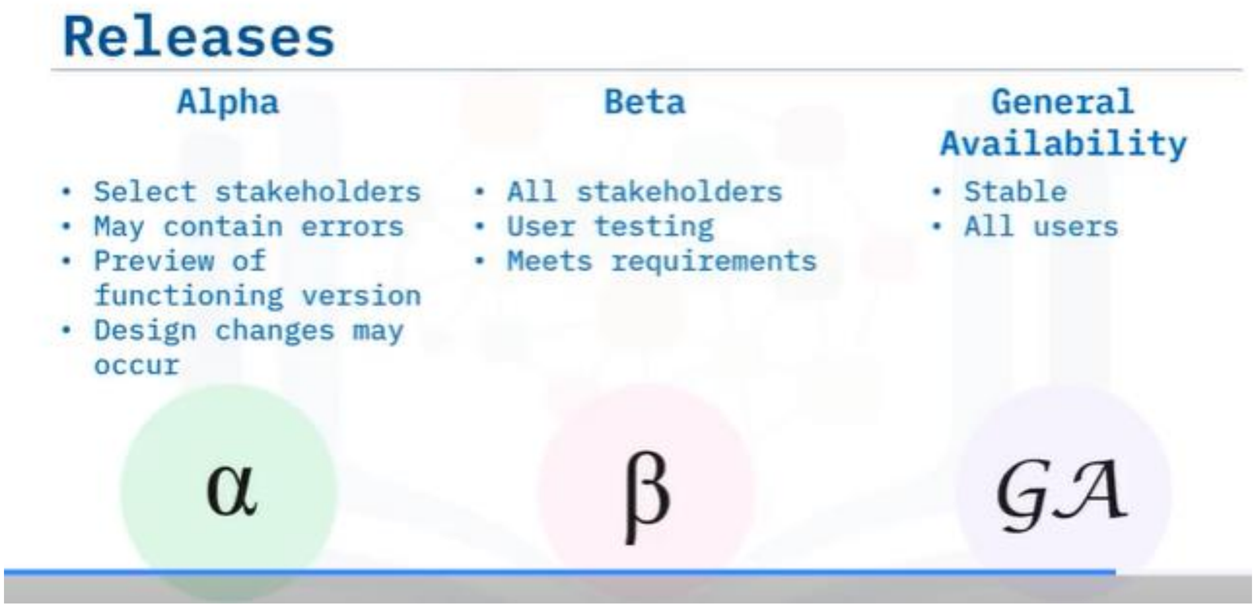
Testing

The process of verifying that the software matches established requirements and is free of bugs

- Identify errors, gaps, or missing requirements
- Ensures reliability, security, performance, and efficiency

Testing

- **Unit testing**
- **Integration testing**
- **System testing**
- **User acceptance testing (UAT) or Beta testing**



Releases

Alpha

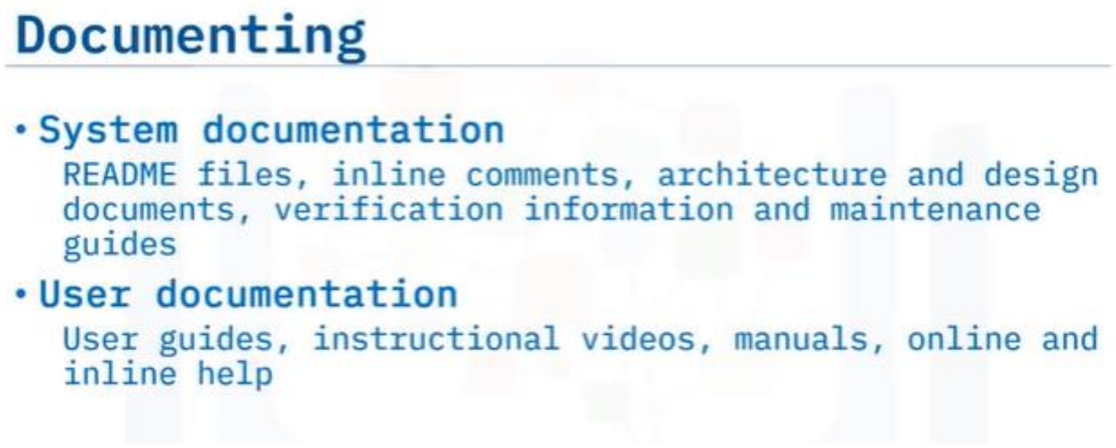
- Select stakeholders
- May contain errors
- Preview of functioning version
- Design changes may occur

Beta

- All stakeholders
- User testing
- Meets requirements

General Availability

- Stable
- All users



Documenting

- **System documentation**
  README files, inline comments, architecture and design documents, verification information and maintenance guides
- **User documentation**
  User guides, instructional videos, manuals, online and inline help

REQUIREMENTS PROCESS

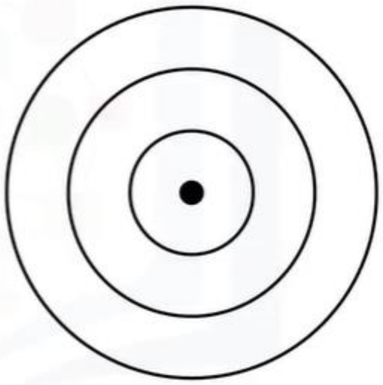
Identifying stakeholders

Key personnel:

- Decision-makers
- End-users
- System administrators
- Engineering
- Marketing
- Sales
- Customer support

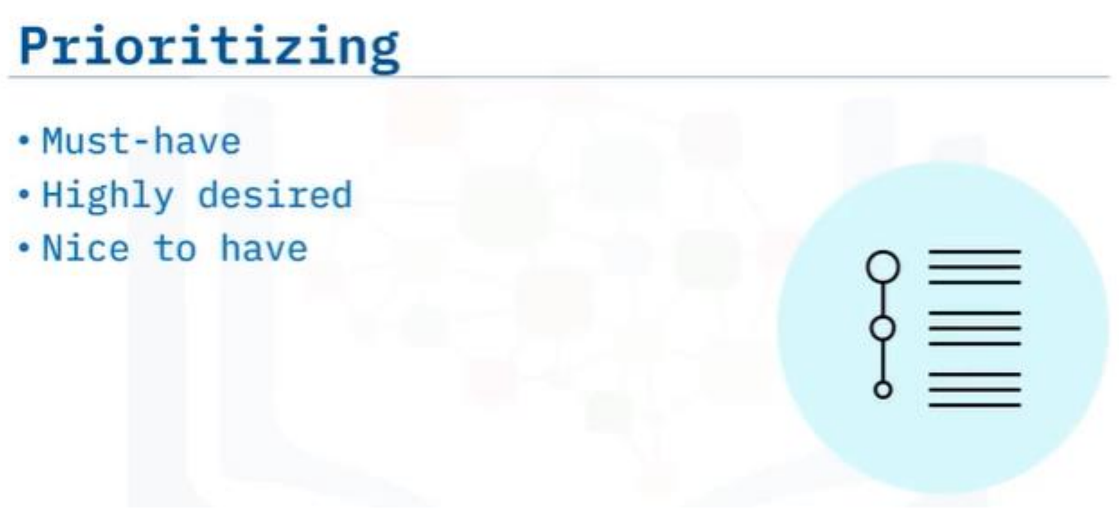
Establishing goals and objectives

- **Goals:** broad, long-term achievable outcomes
- **Objectives:** actionable, measurable actions that achieve the goal



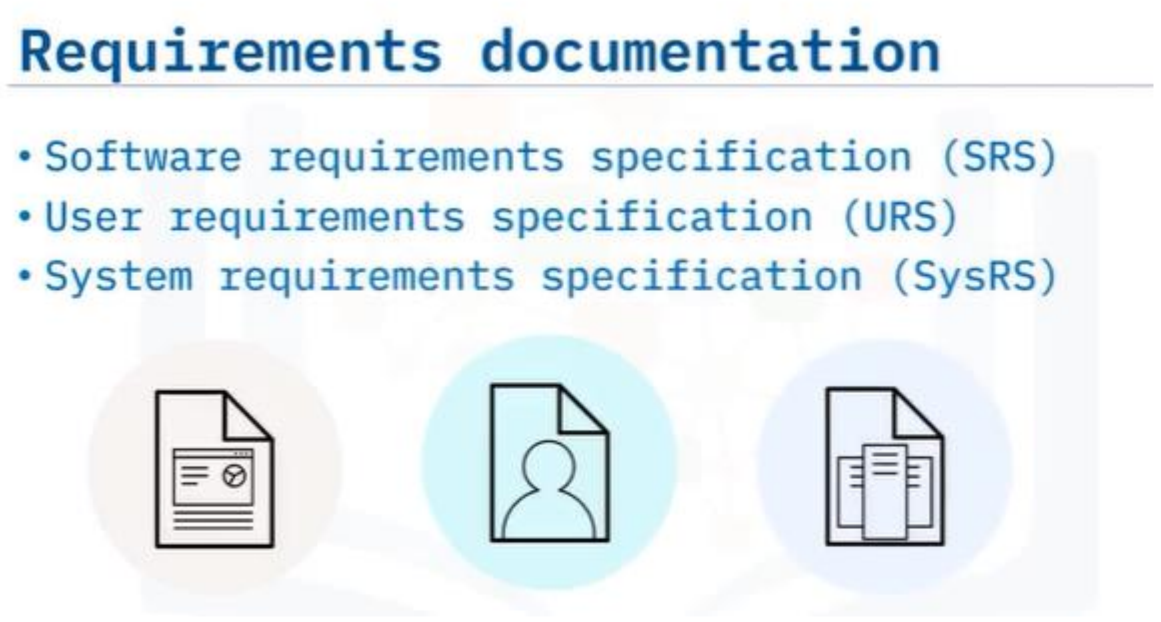
Eliciting, documenting, confirming

- **Elicit**
  - Surveys
  - Questionnaires
  - Interviews
- **Document**
  - Align with goals and objectives
  - Easily understood
- **Confirm**
  - Consistency
  - Clarity
  - Completeness



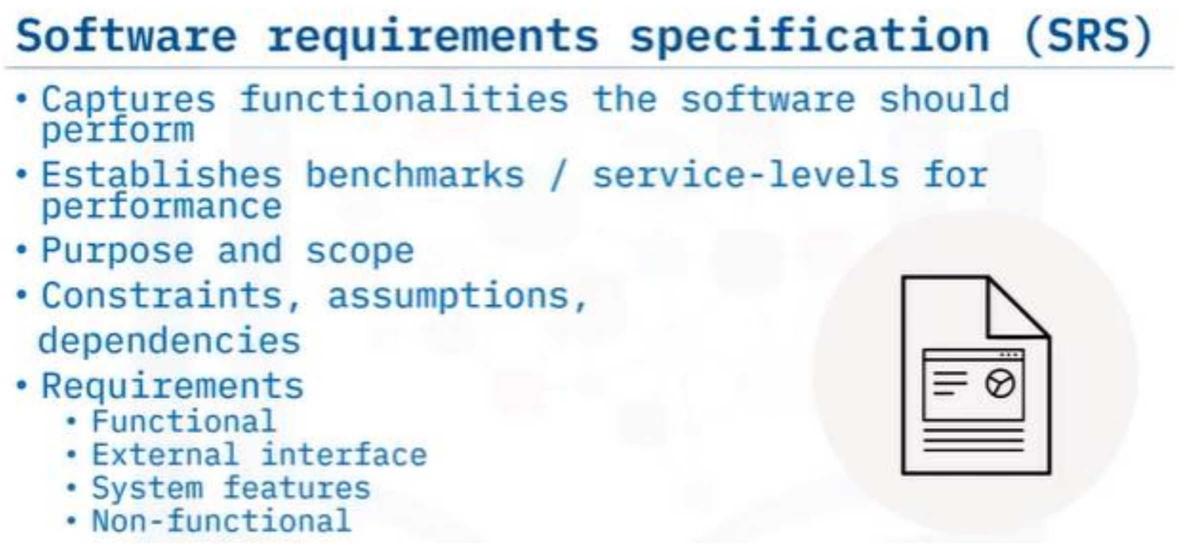
Prioritizing

- Must-have
- Highly desired
- Nice to have



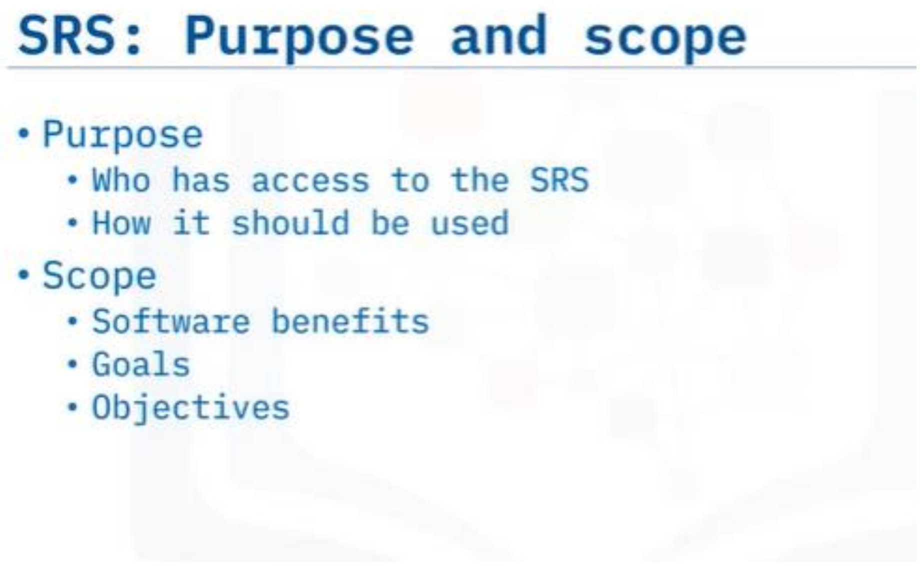
Requirements documentation

- Software requirements specification (SRS)
- User requirements specification (URS)
- System requirements specification (SysRS)



Software requirements specification (SRS)

- Captures functionalities the software should perform
- Establishes benchmarks / service-levels for performance
- Purpose and scope
- Constraints, assumptions, dependencies
- Requirements
  - Functional
  - External interface
  - System features
  - Non-functional



SRS: Purpose and scope

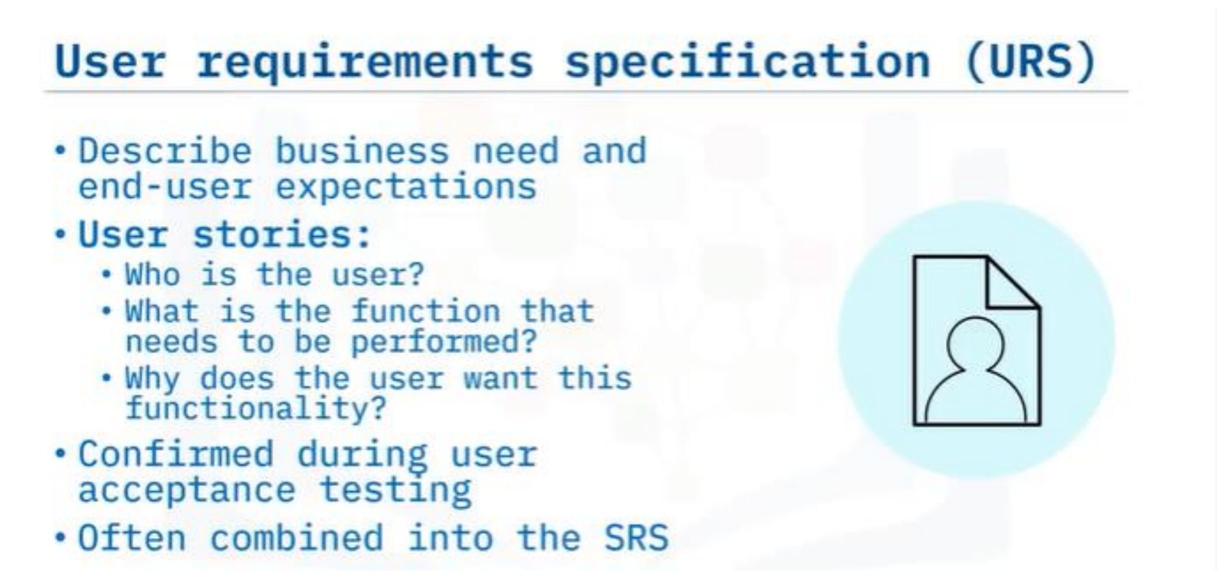
- Purpose
  - Who has access to the SRS
  - How it should be used
- Scope
  - Software benefits
  - Goals
  - Objectives

SRS: Constraints, assumptions, dependencies

- **Constraints:** how the software must operate under given conditions
- **Assumptions:** required OS or hardware
- **Dependencies:** on other software products

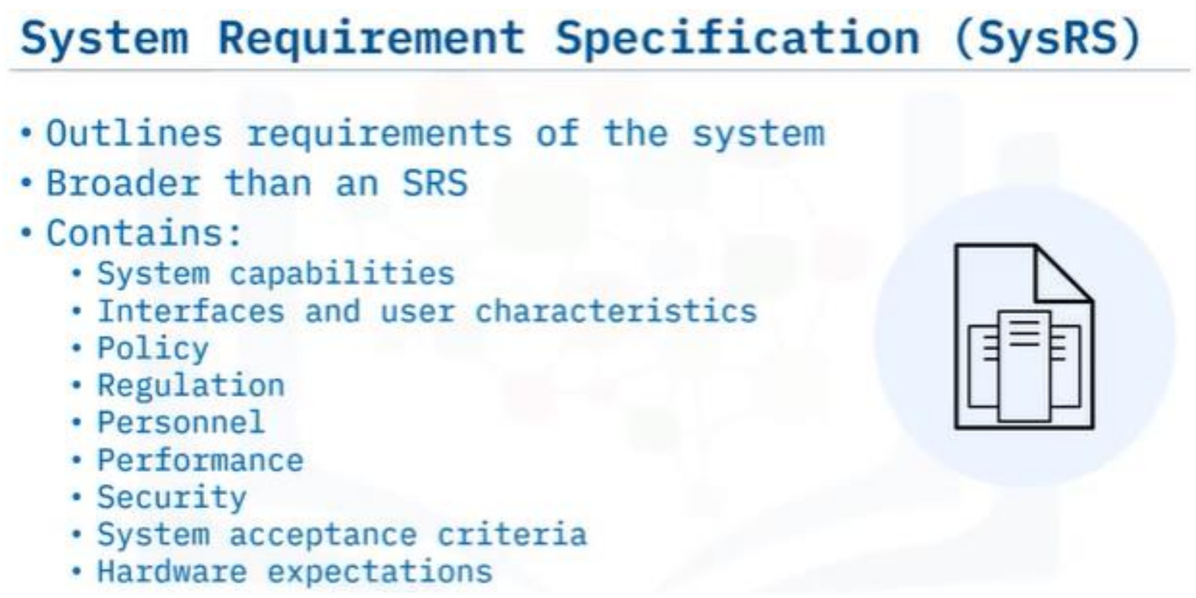
SRS: Requirements

- **Functional:** functions of the software
- **External:** users and interactions with other hardware or software
- **System features:** functions of the system
- **Non-functional:** performance, safety, security, quality



User requirements specification (URS)

- Describe business need and end-user expectations
- **User stories:**
  - Who is the user?
  - What is the function that needs to be performed?
  - Why does the user want this functionality?
- Confirmed during user acceptance testing
- Often combined into the SRS



System Requirement Specification (SysRS)

- Outlines requirements of the system
- Broader than an SRS
- Contains:
  - System capabilities
  - Interfaces and user characteristics
  - Policy
  - Regulation
  - Personnel
  - Performance
  - Security
  - System acceptance criteria
  - Hardware expectations