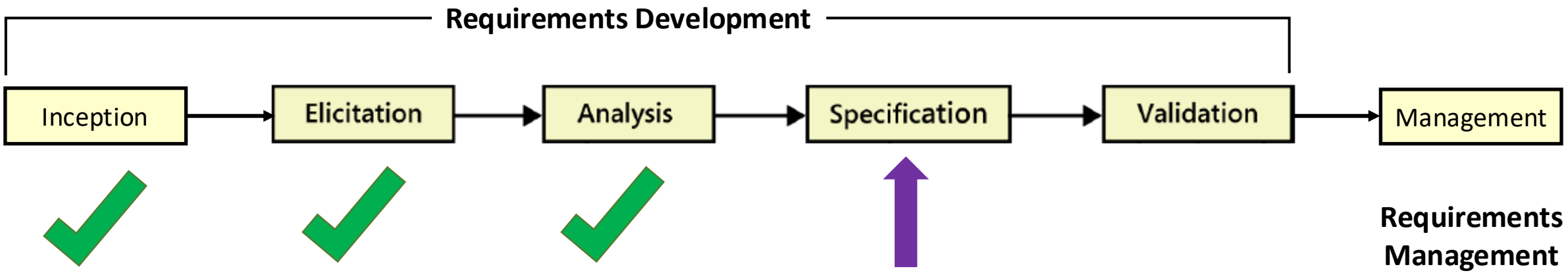
Requirements Specification

RE Process


Requirements Development
Inception
Elicitation
Analysis
Specification
Validation
Management
Requirements Management

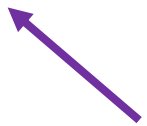
Requirements Specification

- The result of requirements development is a documented agreement among stakeholders about the product to be built.
- The vision & scope document contains the business requirements, and user requirements can be captured in the form of use cases
- Functional and nonfunctional requirements often are stored in a software requirements specification (SRS)
- SRS is delivered to those who must design, build, and verify the solution.

Requirements Specification

You can represent software requirements in several ways, including:

- Well-structured and carefully written natural language.
- Visual models
- Formal specifications that define requirements by using mathematically precise specification languages.



Provide the greatest rigor and precision, but few software developers—and even fewer customers—are familiar with them

SRS

- The SRS goes by many name: a *business requirements document* (BRD), *functional specification*, *product specification*, *system specification*, or simply *requirements document*.
- It states the functions and capabilities that a software system must provide, its characteristics, and the constraints that it must respect.
- It should describe as completely as necessary the system's behaviors under various conditions, as well as desired system qualities such as performance, security, and usability.
- It is the basis for subsequent project planning, design, and coding, as well as the foundation for system testing and user documentation. However, it should not contain design, construction, testing, or project management details other than known design and implementation constraints.

SRS Audience

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

Developers • Need to know what to build	**Project Managers** • Need it for schedule & resource estimations
Customers, marketing, & sales • Know what product they expect to be delivered	**Testers** • Develop requirements-based tests & test plans
Maintenance & support • Use it to understand what each part of the product is supposed to do	**Documentation writers & training personnel** • Use it to develop user manuals and educational material
Legal staff • Ensure that requirements comply with laws & regulations	**Subcontractors** • Base their work on—and can be legally held to—the specified requirements

SRS Audience

- A single requirements deliverable often cannot meet the needs of all audiences
  - Some people need to know just the business objectives, others want only a high-level big picture, still others want to see just the user's perspective, and yet others need all the details.
  - Not all user reps will go through the SRS, and developers need more than just the use cases
  - This is why we create the different models, and the SRS

Labeling Requirements

- Every requirement needs a unique and persistent identifier
  - Helps refer to specific requirements in change requests, modification history, or requirements traceability matrix
  - Helps reuse requirements in multiple projects
  - Facilitate collaboration between team members
- Simple numbered or bulleted lists aren’t adequate for these purposes

Requirements Labeling Methods – Sequence Number

- Simplest approach. Gives every requirement a unique sequence number, e.g. UC-9 or FR-26
- The prefeix indicate the requirement type
- A number is not reused if a requirement is deleted
- Used by commercial requirements management tools

Requirements Labeling Methods – Sequence Number

- Pros:
  - Make it easy to retain a unique identifier if you move requirements around in a document

- Cons:
  - Doesn’t provide any logical or hierarchical grouping of related requirements – number doesn’t imply any order
  - Numeric labels tell you nothing about the intent of a requirement

Requirements Labeling Methods – Hierarchical Numbering

- If for example your functional requirements appear in ”*3.2 Project Priorities*” of your SRS, they will all have labels that begin with “3.2”
- More digits indicate a more detailed, lower-level requirement
  - 3.2.4.3 is a child requirement of 3.2.4.

Requirements Labeling Methods – Hierarchical Numbering

- Pros:
  - This method is simple, compact, and familiar
  - A word processor can assign the numbers automatically
  - Supported in requirements management tools

- Cons:
  - The labels can grow to many digits in even a medium-sized SRS
  - Numeric labels tell you nothing about the intent of a requirement

Requirements Labeling Methods – Hierarchical Numbering

TRAP

Word processors don't generate persistent labels – insert, move, merge, or delete and a lot of the labels will change

Don't let ineffective practices limit your ability to work effectively and sensibly

Mitigation

Number the major sections of the requirements hierarchically and then identify individual functional requirements in each section with a short text code followed by a sequence number, e.g., Section 3.5 Editor Functions has ED-1, ED-2… etc.

Requirements Labeling Methods – Hierarchical Textual Tags

- "*The system shall ask the user to confirm any request to print more than 10 copies*" might be tagged *Print.ConfirmCopies*
  - It is part of the print function and relates to the number of copies to print
- Hierarchical textual tags are structured, meaningful, and unaffected by adding, deleting, or moving other requirements.

Requirements Labeling Methods – Hierarchical Textual Tags

Unique ID = Parent label.Child label

Product.Cart

Product.Discount.Error

Parent
(looks like title, heading or feature name – not a discrete requirement)

Children
(deliver the capability described in parent)

Product: Ordering products from the website	
.Cart	The website shall use a shopping cart to contain products a Customer selects to purchase.
.Discount	The shopping cart shall provide one discount code field. Each discount code provides either a specific discount percentage or a fixed dollar discount amount on specific items in the cart.
.Error	If the Customer enters an invalid discount code, the website shall display an error message.
.Shipping	The shopping cart shall add a shipping charge if a Customer orders a physical product that must be mailed.

Requirements Labeling Methods – Hierarchical Textual Tags

Avoids maintenance problems with hierarchical numebrs

Product: Ordering products from the website

.Cart	The website shall use a shopping cart to contain products a Customer selects to purchase.
.Discount	The shopping cart shall provide one discount code field. Each discount code provides either a specific discount percentage or a fixed dollar discount amount on specific items in the cart.
.Error	If the Customer enters an invalid discount code, the website shall display an error message.
.Shipping	The shopping cart shall add a shipping charge if a Customer orders a physical product that must be mailed.

Tags are longer and more difficult to come up with and maintain uniqueness

Mitigation

Combine hierarchical naming with a sequence number for small sets of requirements, e.g. Product.Cart.01

Dealing with Incompleteness

TBDs won't resolve themselves

- Number them
- Record who's responsible for resolving them and by when
- Review their status at regular checkpoints
- Track them to closure

- Use the notation *TBD* to flag any missing information
- Plan to resolve all TBDs before implementing a set of requirements
- If you must proceed with TBDs, defer implementing the unresolved requirements or design those portions of the product to be easily modifiable
- Record TBDs and other requirements issues in an issues list

User Interfaces and the SRS

- Incorporating user interface designs in the SRS has both benefits and drawbacks.
- On the plus side:

  - Exploring possible user interfaces with paper prototypes, working mock-ups, wireframes, or simulation tools makes the requirements tangible to both users and developers.

User Interfaces and the SRS

Screen layouts don't replace functional requirements

- Don't expect developers to derive underlying functionality and data relationships from screen shots
- If a certain functionality must be implemented with specific UI controls, it's important to include that in the SRS

- On the negative side:
  - Screen images and user interface architectures make the SRS much larger and they might not truly be requirements – big documents frighten people
  - Delaying baselining of the SRS until the UI design is complete can slow down development and try the patience of people who are already concerned about spending too much time on requirements.
  - Including UI design in the requirements can result in the visual design driving the requirements, which often leads to functional gaps.

SRS – Readability Tips

- Use an appropriate template to organize all the necessary information.
- Label and style sections, subsections, and individual requirements consistently.
- Use visual emphasis (bold, underline, italics, color, and fonts) consistently. Remember that color highlighting might not be visible to people with color blindness or when printed in grayscale.
- Create a table of contents to help readers find the information they need.

SRS – Readability Tips

- Number all figures and tables, give them captions, and refer to them by number.
- If you are storing requirements in a document, define your word processor's cross-reference facility rather than hard-coded page or section numbers to refer to other locations within a document.
- Define hyperlinks to let the reader jump to related sections in the SRS or in other files.
- Include visual representations of information when possible to facilitate understanding.

Requirements Specification Best Practices

🏆 Adopt requirement document templates	🏆 Identify requirement origins
🏆 Uniquely label each requirement	🏆 Record business rules
🏆 Specify non-functional requirements	

Requirement Shell

or

Volere Shell