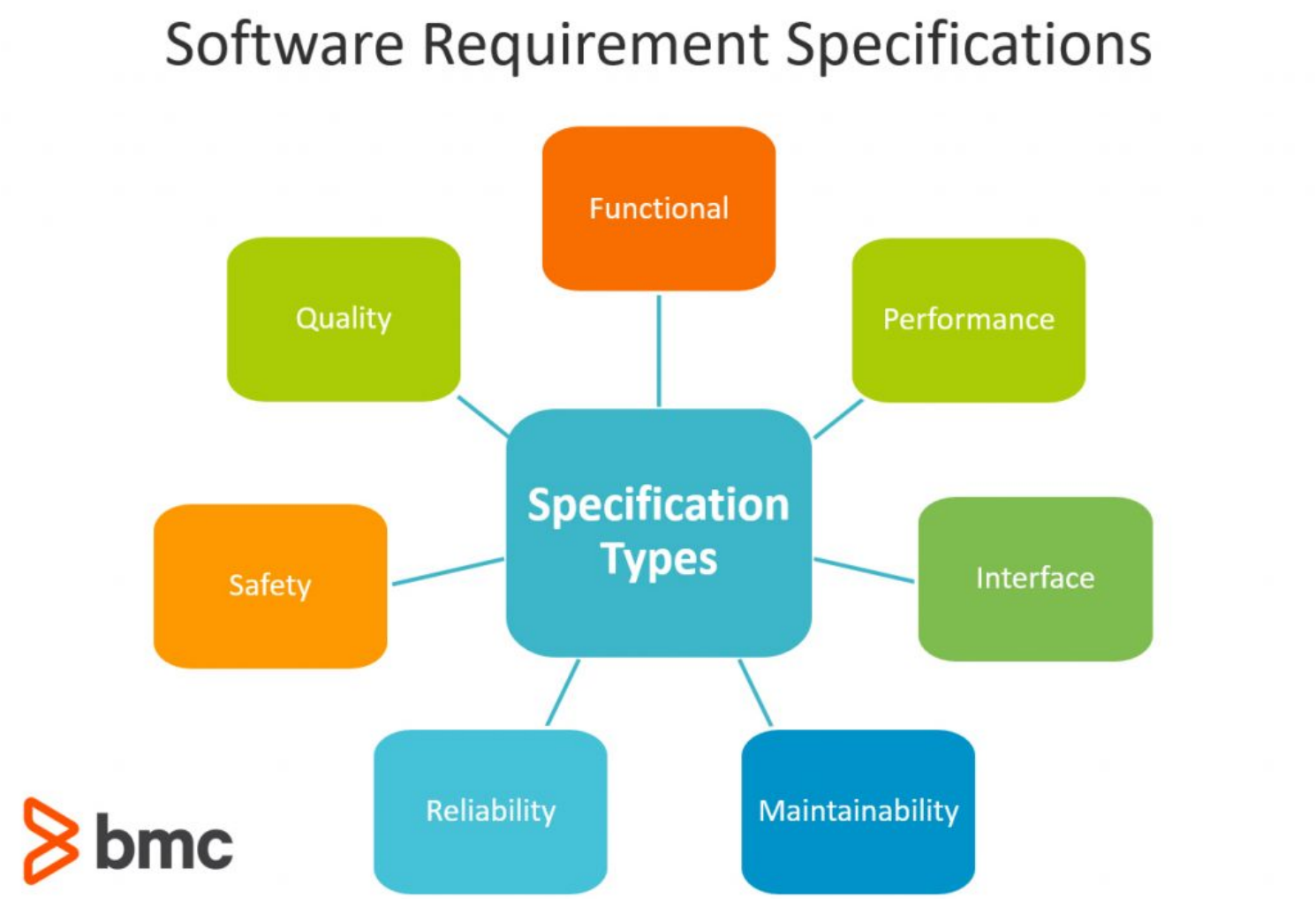
SRS: SOFTWARE REQUIREMENT SPECIFICATIONS BASICS

Software Requirements Specifications, also known as SRS, is the term used to describe an in-depth description of a software product to be developed. It's considered one of the initial stages of the software development lifecycle (SDLC). Think of it like the map that points you to your finished product.

The internet provides many great examples of SRS for developers who are open to learning. The caveat is that, like a map, SRS has to be followed exactly in order for you to arrive at the right destination. To write clear, concise, and easy to follow SRS, you must understand your project. But you must also understand SRS guidelines.

How do you know when your SRS is ready for development? What makes it exceptional? That's what we are going to cover in this article.



Characteristics of exceptional SRS

There are certain things developers should strive to achieve in their SRS document to make it primed for a smooth development project. These can be broken up into three categories:

- Meaningful qualities
- Characteristics that meet goals
- Identifiable requirement smells

Let's take a look.

Meaningful Qualities

The meaningful qualities of SRS are those that are purposeful in helping the developer understand the full scope of the project.

- **Breaks Down the Problem.** A good SRS will break down the problem into chunks that can be solved more readily. This also helps to increase understanding of issues and makes them easier to tackle.
- **Offers Design Input.** Your SRS should contain design details to assist with implementation and deployment.
- **Considers Components for Feedback.** A meaningful quality to users of the finished software is

the opportunity to provide feedback. This should be a consideration when developing a strong SRS.
- **Includes Validation Strategies.** Validation strategies should be implemented to ensure requirements are stated correctly and function the way they are intended to.
- **Requirements are Ranked by Importance.** Ranking the requirements by importance clearly tells both developers and stakeholders where the priorities lie. If the project is coming up on a specific deadline, like the end of a sprint, having a ranking system helps developers shift priorities easily.
- **Complete, Concise, and Modifiable.** The finished product should offer a total picture of the development project as concisely as possible to promote understanding. It should be easily modifiable to account for feedback and changes.

Characteristics that Meet Goals

Each development project should have a pre-established set of goals. These characteristics are used to ensure goals are met and the project stays on the right track.

- **Descriptive Scope of Work.** Having a clear scope of work is one of the most important goals. The scope guides developers through the project. It creates an understanding of what the finished project should be by defining how to get there.
- **Defines Features for the End User.** Customer requirements include certain features for the end user that have to be defined in the SRS.
- **Provides Opportunity for Review with Stakeholders.** One purpose of this document is to have transparency between project managers and stakeholders. That's why reviews of the SRS between both parties are an important benchmark to overall success.
- **Clear Navigation.** A clear, concise document structure with navigation is an important reference point for developers.
- **Testing & Refining.** A goal of any development project is to have a framework for testing. An additional consideration is how you will refine the framework once it's been deployed. The SRS should address both.
- **Estimates Costs.** Importantly, the SRS should be able to estimate costs of development and deployment, as well as operational costs.

Identifiable Requirement Smells

Similar to code smells, requirements smells are indicators that a requirement could be problematic. Developers should pay attention to these characteristics and make changes as necessary.

Resolving them is handled on a case-by-case basis since they don't typically lead to fatal errors in the requirement artifact. That's why they are included among characteristics of exceptional SRS. Developing a fine-tuned nose for these smells will make your work better.

Examples of requirement smells include:

- Ambiguous Adverbs and Adjectives
- Subjective Language
- Superlatives
- Negative Statements

Guidelines for an Exceptional SRS

The content in a SRS can vary from project to project. Even so, each project, no matter how different, should follow a prescribed set of guidelines. These guidelines are easy to remember, since their acronym spells the word **FACTS**.

- **Functional Requirements.** The function of the SRS is separate from that of the development project itself. The functional requirements of this document to provide a framework for implementation should be obvious throughout the document.
- **Analysis Model.** The analysis model allows you to drill down into the specification of certain requirements. An example is if the requirement is "Add Product to Cart," a command that doesn't account for other details like size and quantity. These can be fleshed out with the Analysis Model since it connects functional requirements with the design.
- **Cognitive Model.** This is the model of development that helps developers understand how a system is going to be perceived by others, typically end users.
- **The Content & Structure of the Specification.** This is also known as a data dictionary. It should include all the data surrounding each entity in addition to organizational flow charts.
- **Specification.** Guidelines for the specification itself must be robust enough to tell a story of the development project, and flexible enough to allow changes in scope and scale.

The Structure of Exceptional SRS

There's no one way to structure your SRS, although there are several models to serve as examples. If you've followed the characteristics and guidelines thus far, you're off to a good start.

When it comes to putting the document together, your framework might look something like this:

Purpose/Introduction

- Definitions
- System overview
- References

Overall description

- Product perspective
    - *System Interfaces*
    - *User Interfaces*
    - *Hardware Interfaces*
    - *Software Interfaces*
    - *Communication Interfaces*
    - *Memory Constraints*
- Design constraints
    - *Operations*
    - *Site Adaptation Requirements*
- Product functions
- User characteristics
- Constraints, assumptions and dependencies

Specific requirements

- External interface requirements
- Functional requirements
- Performance requirements
- Logical database requirement
- Software System attributes
    - *Reliability*
    - *Availability*
    - *Security*
    - *Maintainability*
    - *Portability*
- Organizing Specific Requirements

The above example is adapted from IEEE Guide to Software Requirements Specifications (Std 830-1993). The IEEE is an organization that sets the industry standards for SRS requirements. It is the most widely used set of standards when creating an SRS and can be adapted to the needs of each agency.

Defining the Structure

A few key components of the above example are as follows:

Purpose/Introduction

The purpose section should summarize the entire SRS document. It's similar to the executive summary of business documents, and it sets the tone for the project. Typically, key components of this section include definitions, systems overview, and references. These help to establish important themes in the development project.

Overall Description

The overall description gives an overview of the requirements and other subsections. The requirements will be described in greater detail in the specific requirements section. The function of the overall description is to consider determining factors that impact the requirements.

Subsections of the overall description are product perspective, design constraints, product functions, user characteristics and constraints, assumptions, and dependencies. These all have to do with anticipating the needs and challenges that stand in the way of completing the requirements. Design constraints, for example, includes everything from consideration of software compliance to hardware constraints.

Specific Requirements

The purpose of the specific requirements section is to detail all the requirements necessary for development. This section provides a framework for designers to create the product in accordance with requirements.

The specific requirements section is where you'll find external interface requirements, functional

requirements, performance requirements, logical database requirements, and software system attributes. Each of these subsections details a set of requirements necessary for the overall functioning of the program.

Creating an Exceptional SRS

Now you know how to create an exceptional SRS document. A quick search will reveal a number of templates you can apply this new knowledge to if you still aren't 100% confident in your newly learned ability.

It's important to get it right the first time because the SRS is the basis for your entire development project. Ultimately, remember the goal of this document is to assist in a smooth implementation of program development rather than having perfect SRS. Among the major components we discussed, your SRS should be flexible, modifiable, and scalable so that it can change with the demands of the project.

If this seems like a lot of information to take in at once, that's because it is. This article provides a high-level summary of a complex practice. The best way to approach your SRS research is similar to how you should want to frame all of your development projects to stakeholders—in easy to understand pieces of information.

Take it in chunks as you move through each section of the document. When it comes to your next development project, you'll be thanking yourself for taking the time to learn more. As with all things, practice will make your SRS stronger. But these guidelines, characteristics, and structure recommendations are a good start.

Additional Resources

- BMC DevOps Blog
- Deployment Pipelines (CI/CD) in Software Engineering
- What Is Extreme Programming?
- Python vs Go: What's The Difference?
- What Is ADDM? Application Discovery & Dependency Mapping Explained
- Wardley Value Chain Mapping: What Is It & How To Create Yours