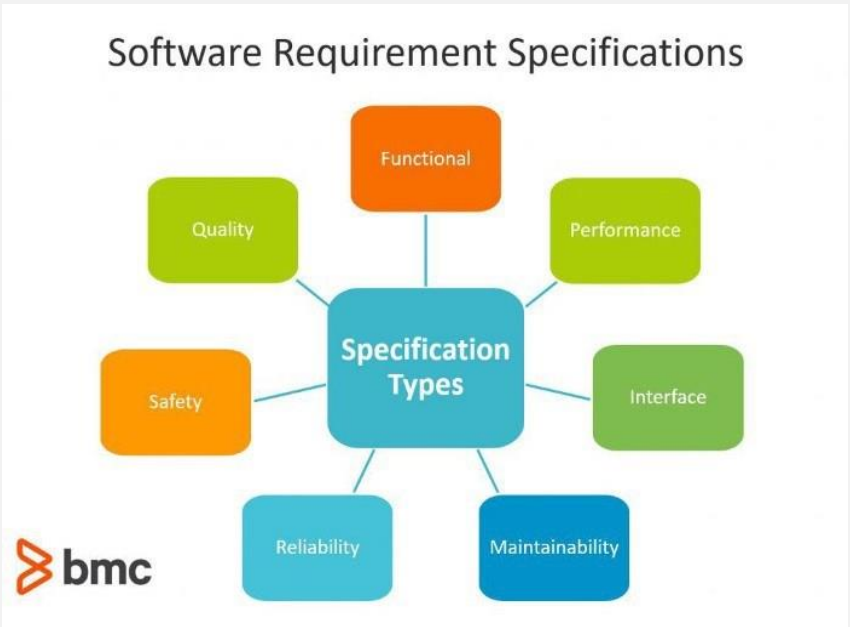
UNIT-3

What is the Software Requirements Specification (SRS)?

A software requirements specification (SRS) is a document explaining how and what the software/system will do. It defines the features and functionality that the product requires to satisfy all stakeholders' (business, users) needs. A standard SRS includes:

- A goal/purpose
- A summary of the whole process
- Specific Requirements

The best SRS documents describe how the program communicates with the embedded hardware or specific software with unique coding culture. The chosen real-life users also account for nice SRS documents.



bmc blog on SRS

Important thing to Note

The *fine line* between Software Requirements Specification and System Requirements Specification:

A Software Requirements Specification **(SRS)** adds in-depth explanations of the software to be built.

A System Requirements specification **(SyRS)** gathers information on the overall system requirements.

Often the "software" and the "system" are used as SRS interchangeably. However, a specification for software needs more detailed information than the specification for the system. In this blog, we will be focusing on **Software Requirements Specifications.**

Before we dive into the main content, a detailed format of making SRS document is provided with the (drive link), please refer the document if you need further guidance on the subject matter.

What does an SRS document contain?

A typical SRS document describes all the software requirements and sometimes even contains a collection of use cases that describe the user interactions needed by the software.

It defines the purpose of a software project, provides the overall definition and specifications of its features.

In general, SRS documents contain three kinds of program requirements:

- ***Functional specifications*** *that include measures to be performed by the system*
- ***Non-functional requirements*** *determining the software system's performance attributes*
- ***Domain requirements*** *that are device limits on the service domain*

Note: If want to know more about Requirement Analysis, please go through my previous blog on this subject matter with the following link.

What is Requirement Analysis in software engineering and Why is it so important? As we know, Requirement Analysis is derived from two words i.e. requirement and analysis where requirement refers to… medium.com

Qualities of a good SRS Document

There must be certain qualities in a well written SRS document, thus it does not cause errors in the development process. The following qualities include:

Correctness

The SRS document is only right if the program meets the required specifications. Thus, no fixed method is given to guarantee the consistency of the SRS, but it can be used to make sure that it stays in line with other superior requirements or documents. An SRS document is said as correct if it covers all the requirements that are actually expected from the system. Users can also decide through feedback whether the SRS represents their needs.

Unambiguousness

As a software development project is based on an SRS document, all the declarations must be simple, concise, and in-depth.

No amphiboles, ambiguous adverbs, words suggesting multiple significance, etc. should be present.

A glossary to clarify each word at the end of the document is often useful. **Completeness**

All the program specifications and answers to input data (valid or invalid) are provided in a complete SRS.

Consistency
As mentioned above, the SRS must be linked to other high-level documents such as the system requirements specifications.
An SRS document must also be reliable with itself, which ensures that the details it contains cannot be changed.
Ranking for importance and/or stability rating
All software requirements are not equally important, some are crucial, and others are less important add-ins.
An SRS document must assess the importance and/or stability of the features of the program in order to enable development teams to complete each task in the right order.
Verifiability
The declarations in the document need to be confirmed, and if the program meets the needs, it can be checked.
This is only attributable to the uncertainty of an SRS, which cannot be supported by unambiguous statements.
Modifiability
Because the process of software creation can change dramatically, software needs can change.
An SRS must be versatile in structure and style, so it can be easily changed if needed.
Traceability
The root of each software requirement must be transparent and future documentation should be easy to reference, which means that the life cycle of each function must be recognizable.
Understandable by Consumers
Here, consumers refer to the end-users who may be an expert in their specific domain. However, they might not have expertise in Computer Science. Therefore, it is very crucial to avoid using formal notations and symbols to make the document as much understandable as possible. And thus, the language used in the document should be kept easy to read and clear to avoid any miscommunications or confusion.

Previous

Home/Software Testing Fundamentals

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing can also provide an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. In this video we describe the fundamentals of software testing.

Different types of Software Testing processes are described below:

- Unit Testing
  It is a method by which individual units of source code are tested to determine if they are fit for use.

- Integration Testing
  Here individual software modules are combined and tested as a group.
- Functionality Testing
  It is a type of black box testing that bases its test cases on the specifications of the software component under test.
- Usability Testing
  It is a technique used to evaluate a product by testing it on users.
- System Testing
  It is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.
- Performance Testing
  It is testing that is performed, to determine how fast some aspect of a system performs under a particular workload.
- Load Testing
  It refers to the practice of modeling the expected usage of a software program by simulating multiple users accessing the program concurrently.
- Stress Testing
  It is a form of testing that is used to determine the stability of a given system or entity.

What is Glass Box Testing?

Glass box testing is a testing technique that examines the program structure and derives test data from the program logic/code. The other names of glass box testing are clear box testing, open box testing, logic driven testing or path driven testing or structural testing.

Glass Box Testing Techniques:

- **Statement Coverage** - This technique is aimed at exercising all programming statements with minimal tests.
- **Branch Coverage** - This technique is running a series of tests to ensure that all branches are tested at least once.
- **Path Coverage** - This technique corresponds to testing all possible paths which means that each statement and branch is covered.

Calculating Structural Testing Effectiveness:

Statement Testing = (Number of Statements Exercised / Total Number of Statements) x 100 %

Branch Testing = (Number of decisions outcomes tested / Total Number of decision Outcomes) x 100 %

Path Coverage = (Number paths exercised / Total Number of paths in the program) x 100 %

Advantages of Glass Box Testing:

- Forces test developer to reason carefully about implementation.
- Reveals errors in "hidden" code.
- Spots the Dead Code or other issues with respect to best programming practices.

Disadvantages of Glass Box Testing:

- Expensive as one has to spend both time and money to perform white box testing.
- Every possibility that few lines of code is missed accidentally.
- In-depth knowledge about the programming language is necessary to perform white box testing.