SRS software Requirement specification-

1. Introduction

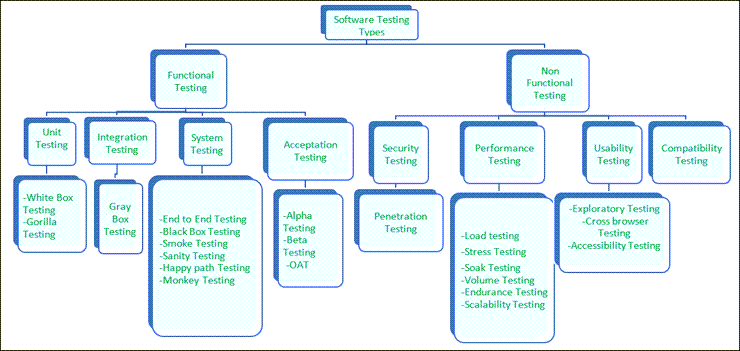
* **(i)** Purpose of this document
* **(ii)** Scope of this document
* **(iii)** Overview

2. General description  
3. Functional Requirements  
4. Interface Requirements  
5. Performance Requirements  
6. Design Constraints  
7. Non-Functional Attributes  
8. Preliminary Schedule and Budget  
9. Appendices

**Software Requirement Specification (SRS) Format** as name suggests, is complete specification and description of requirements of software that needs to be fulfilled for successful development of software system. These requirements can be functional as well as non-functional depending upon type of requirement. The interaction between different customers and contractor is done because its necessary to fully understand needs of customers.

Depending upon information gathered after interaction, SRS is developed which describes requirements of software that may include changes and modifications that is needed to be done to increase quality of product and to satisfy customer’s demand.

1. **Introduction :**
   * **(i) Purpose of this Document –**  
     At first, main aim of why this document is necessary and what’s purpose of document is explained and described.
   * **(ii) Scope of this document –**  
     In this, overall working and main objective of document and what value it will provide to customer is described and explained. It also includes a description of development cost and time required.
   * **(iii) Overview –**  
     In this, description of product is explained. It’s simply summary or overall review of product.
2. **General description :**  
   In this, general functions of product which includes objective of user, a user characteristic, features, benefits, about why its importance is mentioned. It also describes features of user community.
3. **Functional Requirements :**  
   In this, possible outcome of software system which includes effects due to operation of program is fully explained. All functional requirements which may include calculations, data processing, etc. are placed in a ranked order.
4. **Interface Requirements:**   
   In this, software interfaces which mean how software program communicates with each other or users either in form of any language, code, or message are fully described and explained. Examples can be shared memory, data streams, etc.
5. **Performance Requirements:**   
   In this, how a software system performs desired functions under specific condition is explained. It also explains required time, required memory, maximum error rate, etc.
6. **Design Constraints :**  
   In this, constraints which simply means limitation or restriction are specified and explained for design team. Examples may include use of a particular algorithm, hardware and software limitations, etc.
7. **Non-Functional Attributes :**  
   In this, non-functional attributes are explained that are required by software system for better performance. An example may include Security, Portability, Reliability, Reusability, Application compatibility, Data integrity, Scalability capacity, etc.
8. **Preliminary Schedule and Budget :**  
   In this, initial version and budget of project plan are explained which include overall time duration required and overall cost required for development of project.
9. **Appendices :**  
   In this, additional information like references from where information is gathered, definitions of some specific terms, acronyms, abbreviations, etc. are given and explained.

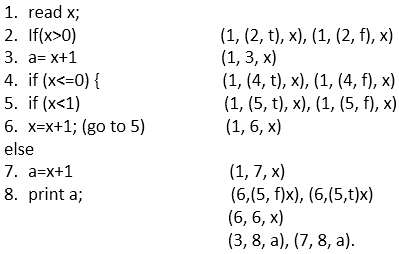
[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2007/08/Classification-of-Software-testing-types.png)

**White Box technique.**

**Checking the internal code of the software.**

* Data flow testing.
* Control flow testing.
* Branch coverage.
* Statement coverage.
* Decision coverage.

Data flow



In this code, we have a total 8 statements, and we will choose a path which covers all the 8 statements. As it is evident in the code, we cannot cover all the statements in a single path because if statement 2 is true then statements 4, 5, 6, 7 not covered, and if statement 4 is true then statement 2 and 3 are not covered.

So, we are taking two paths to cover all the statements.

1. **x= 1**

**Path** - 1, 2, 3, 8

**Output = 2**

When we set value of x as 1 first it come on step 1 to read and assign the value of x (we took 1 in path) then come on statement 2 (x>0 (we took 2 in path)) which is true and it comes on statement 3 (a= x+1 (we took 3 in path)) at last it comes on statement 8 to print the value of x (output is 2).

For the second path, we take the value of x is 1

2. **Set x= -1**

**Path** = 1, 2, 4, 5, 6, 5, 6, 5, 7, 8

**Output = 2**

**Control Flow Diagram**

## **Notations used for Control Flow Graph**

1. Node
2. Edge
3. Decision Node
4. Junction node
5. **public** **class** VoteEligiblityAge{
7. **public** **static** **void** main(String []args){
8. **int** n=45;
9. **if**(n>=18)
10. {
11. System.out.println("You are eligible for voting");
12. }  **else**
13. {
14. System.out.println("You are not eligible for voting");
15. }
16. }
17. }

Control Flow Testing in white box testing Link

**Branch coverage:**

Branch Coverage

1. Read X
2. Read Y
3. IF X+Y > 500 THEN
4. Print "Large"
5. ENDIF
6. If X + Y<500 THEN
7. Print "Small"
8. ENDIF

Branch Coverage

**Path 1** - A1-B2-C4 -5-D6-E8

**Path 2** - A1-B3-5-D7

Branch Coverage (BC) = Number of paths

=2

**Path 1** - A1-B2-C4-5-D6-E8

**Path 2** - A1-B3-5-D7

Branch Coverage (BC) = Number of paths

=2

|  |  |  |  |
| --- | --- | --- | --- |
| **Case** | **Covered Branches** | **Path** | **Branch coverage** |
| Yes | 1, 2, 4, 5, 6, 8 | **A1-B2-C4-D6-E8** | 2 |
| No | 3,7 | A1-B3-5-D7 |

**Statement Coverage:**

Statement Coverage 

1. print (**int** a, **int** b) {
2. **int** sum = a+b;
3. **if** (sum>0)
4. print ("This is a positive result")
5. **else**
6. print ("This is negative result")
7. }

Total number of statements = 7

Number of executed statements = 5

Statement Coverage link

Statement coverage = 5/7\*100

= 500/7

= 71%

Statement Coverage link

**Scenario 2:**  
**If A = -2, B = -7**

1. print (**int** a, **int** b) {
2. **int** sum = a+b;
3. **if** (sum>0)
4. print ("This is a positive result")
5. **else**
6. print ("This is negative result")
7. }

Total number of statements = 7  
Number of executed statements = 6

1. Statement coverage = 6/7\*100 <br>
2. = 600/7
3. = 85%

Statement Coverage link

Statement Coverage link

But, we can see all the statements are covered in both scenario and we can consider that the overall statement coverage is 100%.

Statement Coverage link

Decision Coverage Testing

Decision Coverage technique in whitebox testing link

**Scenario 1:**  
**Value of a is 7 (a=7)**

1. Test (**int** a=7)
2. { **if** (a>4)
3. a=a\*3
4. print (a)
5. }

The code highlighted in yellow is executed code. The outcome of this code is "True" if condition (a>4) is checked.

Control flow graph when the value of a is 7.

Decision Coverage technique in whitebox testing link

Calculation of Decision Coverage percent:

Decision Coverage technique in whitebox testing link

Decision Coverage = ½\*100  (Only "True" is exercised)

                    =100/2

                    = 50

Decision Coverage is 50%

**Scenario 2:**  
**Value of a is 3 (a=3)**

1. Test (**int** a=3)
2. { **if** (a>4)
3. a=a\*3
4. print (a)
5. }

The code highlighted in yellow will be executed. The outcome of this code is ?False? if condition (a>4) is checked.

Control flow graph when the value of a is 3

Decision Coverage technique in whitebox testing link

Calculation of Decision Coverage percent:

Decision Coverage technique in whitebox testing link

1. = ½\*100  (Only "False" is exercised) <br>
2. =100/2
3. = 50
4. Decision Coverage = 50%

## **Result table of Decision Coverage:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Value of A** | **Output** | **Decision Coverage** |
| 1 | 3 | 3 | 50% |
| 2 | 7 | 21 | 50% |