Chapter – 2

Software Specification

Introduction:

- A software specification or software requirement specification (SRS) is a requirement specification for a software project/system. It is a complete description of the behavior of a system to be developed and may include set of use-cases that describe interactions the users will have with the software.
- Enlists all necessary requirements that are required for project development.
- Contains the details of what is to be done and what is not to be done.
- Consists of functional and non-functional requirements.

Functional Requirements:

- Describes everything related to how the system works.
- Example: Functional requirement of a phone is to be able to make and receive calls.

Non-functional Requirements:

- Impose constraints on design or implementation (such as performance quality, design constraints, etc.)
- Example: For that some phone, "to be easy to use" is non-functional requirements

Uses of Specification:

i. A statement of the user requirements or needs.
ii. A statement of the interface between the machine and the control environment.
iii. A statement of the requirement for the implementation.
iv. A reference point during product maintenance.
v. A legal contract.

Specification Qualities:

NASA's Software Assurance Technology Center has identified the following as the ten important criteria that any SRS (Software Requirements Specifications) should satisfy:

1. **Complete**:

A complete requirements specification must precisely define all the real world situations that will be encountered and the capability's responses to them. It must not include situations that will not be encountered or unnecessary capability features.

2. **Consistent:**

System functions and performance level must be compatible and the required quality features (reliability, safety, security, etc.) must not contradict the utility of the system. For example, the

only aircraft that is totally safe is one that cannot be started, contains no fuel or other liquids, and is securely tied down.

3. **Correct**:

The specification must define the desired capability's real world operational environment, its interface to that environment and its interaction with that environment. It is the real world aspect of requirements that is the major source of difficulty in achieving specification correctness. The real world environment is not well known for new applications and for mature applications the real world keeps changing. The Y2K problem with the transition from the year 1999 to the year 2000 is an example of the real world moving beyond an application's specified requirements.

4. **Modifiable:**

Related concerns must be grouped together and unrelated concerns must be separated. Requirements document must have a logical structure to be modifiable.

5. **Ranked:**

Ranking specification statements according to stability and/or importance is established in the requirements document's organization and structure. The larger and more complex the problem addressed by the requirements specification, the more difficult the task is to design a document that aids rather than inhibits understanding.

6. **Testable:**

A requirement specification must be stated in such a manner that one can test it against pass/fail or quantitative assessment criteria, all derived from the specification itself and/or referenced information. Requiring that a system must be "easy" to use is subjective and therefore is not testable.

7. **Traceable:**

Each requirement stated within the SRS document must be uniquely identified to achieve traceability. Uniqueness is facilitated by the use of a consistent and logical scheme for assigning identification to each specification statement within the requirements document.

8. **Unambiguous:**

A statement of a requirement is unambiguous if it can only be interpreted one way. This perhaps, is the most difficult attribute to achieve using natural language. The use of weak phrases or poor sentence structure will open the specification statement to misunderstandings.

9. **Valid:**

To validate a requirements specification all the project participants, managers, engineers and customer representatives, must be able to understand, analyze and accept or approve it. This is the primary reason that most specifications are expressed in natural language.

10. **Verifiable:**

In order to be verifiable, requirement specifications at one level of abstraction must be consistent with those at another level of abstraction. Most, if not all, of these attributes are

subjective and a conclusive assessment of the quality of a requirements specification requires review and analysis by technical and operational experts in the domain addressed by the requirements.

CLASSIFICATION OF SPECIFICATION STYLES:

1. **Informal:**

Not formal, more flexible, leave more decision space to implementer.

2. **Formal:**

Use of formalisms make specifications precise and augmented automatic verification possibilities. They have syntax, their semantics fall in one domain and they are able to be used to take useful information.

3. **Semi-Formal:**

Often we use notation which semantics has not been defined so precisely (example: UML)

4. **Operational:**

Describe the system in terms of the expected behavior generally providing a model. Describes the operational detail. Example: DFD, FSM (Finite State Machine), etc.

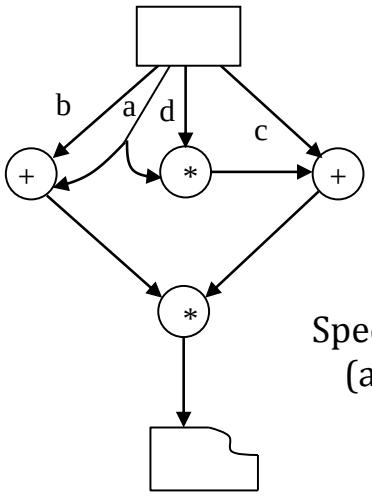
5. **Descriptive Specification:**

Describe the system in terms of desired properties for the system. Example: collaboration diagram, sequence diagram, etc.

OPERATIONAL SPECIFICATION:

1. **DFD (Data Flow Diagram):**

DFDs are well known and widely used notation for specifying the functions of an information system, and how data flow from function to function. They describes system as collection of functions that manipulate data.



Specifies evaluation of
(a + b) * (c + a * d)

2. UML (Unified Modeling Language):

Consider an online subscription system (OSS), its purpose is to provide online learning materials for a specific field. The system has general users and registered users. General users can view and download free subscriptions and asks for membership. Registered users can view, download free subscriptions as well as can view download and search paid subscription after payment is performed. He/she can also asks for extension of validity. OSS administrator can check for validity of contents provided by the editors of the contents and upload the contents to the web. Finance administrator can keep track of payment done and about validity of users. Editors are assigned by OSS administrator in order to collect the online contents for the web.

3. FSM (Finite State Machine):

- Describe control flow
- FSM are widely known and important diagram in UML
- An FSM consists of following things:

a. Finite set of states, Q
b. Finite set of inputs, I
c. Transition function δ = Q * I -> Q
   δ can be some event or values
   Nodes represents states