- **Definitions and acronyms**: The developer and engineers may use certain technical acronyms whereas the clients / product owners may use certain financial acronyms. This will create confusion among both the parties. So, to remove such misunderstanding, the SRS should have all the acronyms, their definitions, their functions, affiliated stakeholder, etc.

(ii) **Overall description**: This section of the SRS includes user needs, assumptions and dependencies.

- **User needs**: Here we need to decide which user problems we want to solve and what solutions we want to give. If the users are very diverse, then we can also divide them into groups. We should also leave room for modifications, if in case a problem turns out to be more significant than we initially thought.
- **Assumptions and dependencies**: It describes the team's views and expectations about the product and its functionalities. For instance, if we are developing a mobile app to help drivers drive at night, then it will obviously be used mostly in the night. Though some may use it in the day, but our focus and goal should be to develop the features that will be more relevant for night time usage rather than for day time.

(iii) **System features and requirements**: The earlier sections give a broad idea about the product, but this section gives a more in-detail description about the software's different functionalities and its execution criteria. This the most important section of the SRS where we set up 'use cases' and distinctly define how a user will interact with the system. Here each user interaction can be a use case. This section of the SRS includes system requirements, functional and non- functional requirements, external interface requirements and other supplementary requirements.

13	Supportability	It is the ability of a system that satisfies necessary requirements and needs to identifying and solving problems.
14	Testability	It shows how well the system or component facilitates to perform tests to determine whether the predefined test criteria have been met.
15	Usability	It is described as how the user is utilizing a system effectively and the ease of which users can learn to operate or control the system. One of the well-known principles of usability is to make it very user-friendly and keep it as simple as possible.

Table 13.1: Non-functional Requirements

In software industry, the essential conformation to requirement is that software should be able to accept and process valid input, produce the anticipated output, should be free of errors and bugs. The assessment of bugs and errors can be measured in two ways: defect rate (i.e., number of defects per million lines of source code, per function point, or other unit) and reliability (i.e., number of failures per n hours of operation, mean time to failure, or the probability of failure-free operation in a specified time). But software quality also depends on other non-functional attributes like Complexity, Efficiency, Interoperability, Learnability, Modularity, etc. Particular software may not be able to satisfy all the above mentioned attributes — for example, if we want to improve robustness then this might lead to loss of efficiency. Therefore, the quality management team should try to find out what are the major criteria for the system being developed. Say, for example, if efficiency is very important then there is reason why other criteria like scalability or usability may be relaxed. This should be stated in the quality plan so that the engineers working on the software development can cooperate to achieve this.

13.4 LEVELS OF QUALITY ASSURANCE

As we have discussed earlier that software quality depends not only on functional attributes but also on non-functional attributes. Thus Software Quality Assurance (SQA) is a method to assure that the best product or

service is being delivered the customer by ensuring that all the essential functional as well as non-functional attributes are satisfied. Software quality assurance gives importance not only to the quality of the software but also to the various process that were used to develop, test and release the software.

Thus the purpose of Software Quality Assurance (SQA) is to ensure that the software meets the required specifications and standards. It provides a clear picture about the performance of the software, information of resource allocation, existence of risk, deficiencies, etc. To assure software quality, many reviews and audits are performed throughout the life cycle of system development process.

There are several levels of quality assurance which are described as below:

- Level 1 " Code Walk-through: Here the software is checked to see if the official coding rules are followed or not. In code walk-through, the documentation and level of in-code comments are thoroughly examined.
- Level 2 " Compilation and Linking: Here the software is checked to see if it can compile and link in all official platforms and operating systems.
- Level 3 " Routine Running: When placed under different constraints and conditions, it is checked to see if the software can run still properly.
- Level 4 " Performance test: At this final level, it is checked to see whether the performance of the software satisfies the specified performance level.



EXERCISE

1. In industry, two definitions of quality exist: "" and "". (Fill in the blank)
2. Complexity, Efficiency, Interoperability, Learnability, Modularity, etc. are the functional requirements of software. (True or False)
3. Why software quality cannot be determined only by satisfying functional requirements?

will get to know about the technology aspects.Once the SRS has been written, developers can refer it at any point of time to clear any ambiguity, managers can use it to plan the system development and maintenance process, users / clients can use it to understand the software, test engineers can use it to create test cases, develop validations sets, etc. Thus the SRS will provide the system with a concrete and well-defined structure to follow.

13.6.1 Structure of Software Requirement Specification

The structure of a software requirement specification (SRS) documenthas been described in detail, as given below:

(i) Introduction: The first section of SRS is the introduction part where we describe the overall SRS meaning, its scope, and its structure.It includes Purpose, Intended audience, Intended use, Scope and Definitions and acronyms.

- **Purpose**: It is a short section where we describe the purpose of the SRS documentation, its structure, type of requirements andthe different stakeholders who will work with the document.
- **Intended audience**: It is a detail description of what the stakeholders like product owners, investors, business analysts, developers, testers, etc will do and how they will participate in its creation.
- **Intended use**: It the information about the situations under which the stakeholders will use the SRS document. For example: designing and brainstorming new features, planning project duration, estimating costs, evaluating risks, monitoring team's success, clearing conflicting scenarios etc.
- **Scope**: Here we have to describe the different users who will use the software. We also have to briefly describe all essential parts of the architecture, how our solution is different from others and what the marketing strategies are.

think that apart from a few additional features, your previous knowledge is enough to design the new software. You have some verbal discussions with your team members as well as with the client and start making the software. You may end up with following scenarios: the end product may not be exactly as the client wanted and there may be some features missing.Say for example you did not know that the teachers can attend or take leave from the school in the middle of the day; or say you did not know that apart from attendance, the teachers are required to register the detail progress of the students. Another case may be that you have added a functionality which is not required by the client at all. For example, say you have given the provision that after the teacher has entered the marks, the students can view their marks immediately; but in order to prevent any confusion among the students, the school wants to release the marks only at a certain date after the marks of all the subjects have been entered,evaluated and properly checked.There is another scenario: while working on certain functionality, there is dispute and disagreement among your own team members. For example, some of your team members might say that the software should be fully accessible to the teachers even outside the school premises, whereas some might argue that outside the school premises the teachers should have limited accessibility. Thus you will eventually discover that even though you might have previous experiences in similar areas, but every organization has its own unique sets of requirements and constraints, which along with all the stakeholders, has to be clearly discussed, thoroughly examined and properly planned, otherwise there will be utter chaos, disarray, disorganization and disappointment.

With the software requirements specification (SRS), the developers, the product owners, and various other stakeholders will have to meet, discuss and formally write down about the system's detailed description and purpose, its functionalities, its interfaces, performance criteria, functional and non-functional requirements, quality standards, security issues, financial as well as time constraints,resource allocation, risk involved, deficiencies, etc., all has to be noted down. Here, the developers will get to know what goals and vision the clientexactly has in his or her mind, whereas the client



CHECK YOUR PROGRESS

Q1: In terms of quality, what does "conformation to requirement" and "fitness for use" mean?

Q2: What are the functional requirements and non-functional requirements of software?

Q3: What is the purpose of Software Quality Assurance?

13.5 QUALITY FACTOR SPECIFICATION

Software may successfully fulfill the basic requirement of essentially having no inherent bugs or errors, but it may not stand strong on other non-measurable attributes like Reusability, Robustness, etc.As a result, the overall quality of the software may be hampered. Thus, software quality is assessed not only by measureable attributes like bugs, logical errors, etc., but also depends on other non-functional attributes like Interoperability, Learnability, Maintainability, etc. So, we need an all-inclusive method which will cover all the attributes and aspects of software quality.

13.5.1 McCall's Factor Model

The McCall's Factor Model classifies all software requirements into eleven software quality factors. The 11 factors are grouped into three categories – *product operation*, *product revision*, and *product transition factors*.*Product operation*has five factors " Correctness, Reliability, Efficiency, Integrity, Usability.*Product revision*has three factors" Maintainability, Flexibility, Testability. And *Product transition*also has three factors " Portability, Reusability, Interoperability.

(i) Product operation factors: It has five software quality factors. It relates to the requirements that directly affect the daily operation of the software. Each one of them are described as given below:

- Correctness: It says how accurately and correctly the software can do its predefined task and provide the required output. It has many elements like:

- Output mission.
- The required accuracy of output that can be negatively affected by inaccurate data or inaccurate calculations.
- The completeness of the output information, which can be affected by incomplete data.
- The up-to-dateness of the information defined as the time between the event and the response by the software system.
- The availability of the information.
- The standards for coding and documenting the software system.

- Reliability: It relates to the extent the software can keep functioning in the event of environmental failures. They determine the maximum allowed failure rate of the software system, and can refer to the entire system or to one or more of its separate functions.
- Efficiency: It is about the hardware resources required for the software to run and provide the required services. It includes processing capabilities (in MHz), storage capacity (in MB or GB), data communication capability (in MBPS or GBPS) and the time required between recharging of the system's portable units, such as, information system units located in portable computers, or meteorological units placed outdoors.
- Integrity: It says how the software provides security by giving valid access to authorized person or group of persons, the level of access (like read/ write permission, etc.) and firmly blocking people with unauthorized access.
- Usability: Usability requirement relates to the extent of effort required for a user to learn, operate, interpret and understand the functions of the software.

(ii) Product revision factors: It has three factors which are as follows:

- Maintainability: This factor specifies the effort required to identify the software failure, to find the reason for the failure, to correct the failure, and to finally verify its correctness.
- Flexibility: In the event of accommodating additional features and requirements, whether the software has the ability to adapt and how much effort is required to do so without bringing major changes to the software.
- Testability: It deals with testing all the components and functions of the software and obtaining a report about the detected faults. To find whether all the components are working fine predefined intermediate results and log files are used; as well as automatic diagnostics are performed by the software prior to starting the system.

(iii) Product transition factors: There are three factors which are as follows:

- Portability: Portability requirement depicts how well the software is able to adapt in different hardware and operating systems and still perform its required functions.
- Reusability: This factor deals with the ability to reuse the various modules of the software in order to augment new features and capabilities in the existing system.
- Interoperability: This factor focus on the ability to integrate and interact with other software systems or other equipment firmware. For example, the firmware of the production machinery and testing equipment interfaces with the production control software.

13.6 SOFTWARE REQUIREMENT SPECIFICATION

Let us say a client / customer has requested us to make a comprehensive School Management Software. Let us assume that you have experience in developing Employee Management System and you