- The McCall's Factor Model classifies all software requirements into eleven software quality factors which are grouped into three categories.
- Without software requirements specification (SRS), there will be utter chaos, disarray, disorganization and disappointment among all the relevant stakeholders.
- The characteristics of a good SRS are (i) Correctness, (ii) Completeness, (iii) Consistency, (iv) Unambiguousness, (v) Ranking for importance and stability, (vi) Modifiability, (vii) Verifiability, (viii) Traceable, (ix) Design Independence, (x) Testability, (xi) Understandable and (xii) Right level of abstraction.
- Software Design Specification (SDS) describes the technical details of how the features and functions have to be implemented. It will help the programmer during coding and implementation.



13.9 FURTHER READINGS

1. Wixom, B. H., Roth, R. M., (2008), Systems Analysis and Design, Wiley Publishing
2. Kendall, K. E., Kendall J. E., (2019), Systems Analysis and Design, Pearson



13.10 ANSWERS TO CHECK YOUR PROGRESS

Ans. to Q No1: Conformation to requirement means that the requirements and standards for a product are clearly predefined and the product has to satisfy all these set requirements in order to be considered of good quality. For example, a hatchback car has to have certain requirements like a rear flip-up hatch door, front-wheel drive-train, foldable rear seats, etc. A car has to satisfy all the essential necessities of hatchback car to be considered a quality hatchback car.

- **Functional requirements**: Here the functional requirements of the software are clearly defined, that is, here the importance is given to the essential question: 'What has to be developed'? or 'What are supposed to be the essential functions of the system'? The internal logic like 'how to develop the functionalities' or 'when to develop them' is not concentrated right now. While describing the functional requirements of the system, the views of the users / customers / product owners are primarily taken into consideration.
- **External interface requirements**: Say, for a food delivery mobile app, 'search' will be an essential functional requirement, but the search option itself may have a list of other requirements like search by location, by restaurant, by food, by price, by user satisfaction, etc. Such requirements are called as external interface requirements.They include the elements like themes, designs, list of pages, etc., that will be visible to the client / user while using the system. Depending on the project, external interface requirements can consist of four types: user interface, software interface, hardware interface, communication interface.
- **System requirements**: System requirements describe the minimum as well as maximum hardware requirement for the software to run smoothly. This is especially important for the developers because they should not develop software that does not comply with the already stated hardware requirements.
- **Non-functional requirements**: As the name suggests, here the non-functional requirements like Efficiency, Interoperability, Modularity, Portability, Reliability, Robustness, etc. are elaborated. For functional requirement we deal with the questions like: 'What has to

be developed'? or 'What are supposed to be the essential functions of the system'?; for non-functional requirements we have handle the questions like: 'How each feature / function of the software should work? Given certain conditions and limitations, how it should behave? How to handle accessibility and security issues?' etc. Here we have to describe how the software has to work in terms of Efficiency, Interoperability, Modularity, Scalability, etc. As the non-functionalattributesarevery difficult to describe because different person will have a different point of view, so we can assign score or point to each attribute.

- **Other supplementary requirements**: Finally, at the end of the SRS we include additional details like appendixes, glossaries and references that might be of some help to the developers and engineers.

13.6.2 Characteristics of a Good Software Requirement Specification

The following are the characteristics of a good Software Requirement Specification (SRS):

(i) Correctness: A SRS should accurately describe all the required functionalities and specifications of the software that is to be built, without any conflicts. User / client review can be used to check the correctness of the SRS.

(ii) Completeness: A SRS should fully and completely describe all the functionalities and specifications of the software to be developed. It should clearly address various queries like 'What exactly the software is supposed to do?', 'Given different types of input, how the software is supposed to response?', 'How complex it should be?', 'What security privileges should be there?' etc. It is very difficult to determine completeness as it is very hard to spot the absence of specification. So, one has to vigorously and diligently look for all the different aspects and perspectives of the software.

communication time and storage size and more documentation has to be maintained.

(iv) Strategy of Design: To design a system, there are two possible approaches: Top-down approach and Bottom-up approach. In top-down approach we identify the main components and then decomposing them into their more detailed sub-components. In bottom-up approach we begin with the lower details and move up the hierarchy. This approach is suitable in the case of an existing system.



CHECKYOURPROGRESS

Q8: What is the difference between Software Requirement Specification and Software Design Specification?

Q9: Describe the characteristics of a good Software Design Specification.



13.8 LET US SUM UP

- Quality of a product depends on an individual person's need, interest, opinion and point of view. Different people may visualize it differently.
- To quantify quality in industry, two definitions of quality exist: "conformation to requirement" and "fitness for use".
- Software quality depends not only on functional attributes but also on other non-functional attributes like Complexity, Efficiency, Interoperability, Learnability, Modularity, Portability, Reliability, Resilience, Reusability, Robustness, Scalability, Security, Supportability, Testability and Usability.
- The quality management team should try to find out what are the major non-functional attributes for the system being developed.
- The purpose of Software Quality Assurance (SQA) is to ensure that the software meets the required specifications and standards.

13.7.2 Software Design Principles

Software design principles are applied to effectively handle the complexity of software design. By following the software design principles we can reduce the effort of design as well asreduce the occurrence of errors. Following are the principles of Software Design:

(i) Problem Partitioning: Though small projects may be easier to handle, the same cannot be said of large projects. Large projects cannot be handled all at once, so, we have to divide the large projects into smaller parts and deal with the smaller parts separately. Breaking it into smaller pieces makes the large project easier to understand, test, modify and maintain. Since the smaller parts are fragments of a large project, the smaller parts cannot be independent of each other and has to communicate with each other to complete the whole project.

(ii) Abstraction: In this method the designer is not concerned about the internal details about the implementation. There are two common abstraction mechanisms: Functional abstraction and Data abstraction. In functional abstraction,functions are used to specifythe job it is supposed to do. The details of the algorithm to accomplish the functions are not visible to the user of the function. It is done by using function oriented design approaches. In data abstraction, details of the data elements are not visible to the users of data. It is done by using object oriented design approaches.

(iii) Modularity: In modular design, the system is divided into modules such that each module is well-defined, more readable, less complex, eachmodule has specified objectives, can be separately compiled and saved in the library, can be separately written by different people, provides more checkpoints to measure progress, provides a framework for comprehensive testing. There are some disadvantages of modularity like longer compilation time, execution time, loading time, inter-module

(iii) Consistency: SRS is said to consistent if there is no conflict and disagreement between any requirements. There should not be any logical conflicts like for example say, one part of the SRS says the user can access its configuration page but another part in the SRS says only administrator can have access to the configuration page, or for example one part in the SRS says that the administrator can perform backup any time, but another part says backup is restricted during peak hours as it might slow the system. There should not even be conflicts in simple terminology definition, for example different requirements may use different terms to refer to the same objects.

(iv) Unambiguousness: A SRS is said to unambiguous when every requirement and every element of the software has been uniquely interpreted and distinctively explained. To avoid ambiguity, one should avoid using words, phrases or sentences which can have vague or loose translations, and pursue using methods like ER diagrams, reviews, inspections, etc.

(v) Ranking for importance and stability: SRS should clearly classify each functional requirement, feature, use case, or user story into its level of importance or significance to mark its rank or stability. Such ranking will come handy if there aresome budget cuts, schedule overruns, loss of resources or addition of new requirements. Here stability implies the probability of changes in the requirement in future.

(vi) Modifiability: A SRS is said to be modifiable if to some extent the SRS can be modified and changes can be incorporated, without affecting other characteristics like unambiguity, correctness, etc. Also it is necessary to maintain the history of the changes that has been made.

(vii) Verifiability: A SRS is verifiable if there exists some method to measure the extent to which each requirement has been met. To determine whether each requirement has been met, we can perform tests or should at least be able to use other verification approaches, such as inspection, reviews or demonstrations.

(viii) Traceable: The SRS is traceable if each requirement in the SRS is uniquely labeled and identified and that it is forward traceable as well as backward traceable. Forward traceable means we are able to trace a requirement to its design component and then to the code segment. Backward traceable means we are able to trace each requirement explicitly back to its source in the earlier documents.

(ix) Design Independence: The SRS should not include any implementation detail so that there are multiple design alternatives for the system.

(x) Testability: A SRS is testable if it is written in such a way that it is very easy to generate test cases and test plans from the report.

(xi) Understandable: The language used in writing the SRS should be simple and very clear; and the use of formal notations and symbols should be limited as much as possible so that the client / user / investors can easily understand it.

(xii) Right level of abstraction: The SRS should be written according to the appropriate level of abstraction. If the SRS is written for the requirements phase, the details should be explained explicitly, whereas, for a feasibility study, fewer details can be used.



EXERCISE

1. The McCall's Factor Model classifies all software requirements into _____software quality factors which are grouped into _____categories.
2. What are the factors included in each of the three categories of McCall's Factor Model.
3. What will happen if you do not prepare the SRS?
4. Name some stakeholders of software product.



CHECK YOUR PROGRESS

Q4: Why do we have to use McCall's Factor Model?

Q5: State the significance of Software Requirement Specification (SRS).

Q6: While constructing the Software Requirement Specification, how to construct the functional and non-functional requirements?

Q7: State the characteristics of a good SRS.

13.7 SOFTWARE DESIGN SPECIFICATION

We have understood that Software Requirement Specification (SRS) describes what the various stakeholders (like customers, users, project managers, software developers, engineers, investors, test engineers, etc.) want in terms of features, functions, performance, constraints, quality, security, etc. Software Design Specification (SDS) describes the technical details of how the features and functions have to be implemented. It will contain information about the software architecture and describe how each component will contribute to meeting the requirements. It will help the programmer during coding and implementation.

13.7.1 Characteristics of a Good Software Design Specification

A good design specification should have the following characteristics:

(i) Correctness: Software design should be correct as per requirement.

(ii) Completeness: The design should have all components like data structures, modules, and external interfaces, etc.

(iii) Efficiency: It should allocate resources very efficiently.

(iv) Flexibility: In case of occurrence of changes, it should be modifiable.

(v) Consistency: There should not be any inconsistency in the design.

(vi) Maintainability: The design should be as simple as possible so that it can be easily maintained by other designers.