## 1. Distinguish between agile software and waterfall model of software development

| Agile | Waterfall |
|---|---|
| It separates the project development lifecycle into sprints. | Software development process is divided into distinct phases. |
| It follows an incremental approach | Waterfall methodology is a sequential design process. |
| Agile methodology is known for its flexibility. | Waterfall is a structured software development methodology so most times it can be quite rigid. |
| Agile can be considered as a collection of many different projects. | Software development will be completed as one single project. |
| Agile is quite a flexible method which allows changes to be made in the project development requirements even if the initial planning has been completed. | There is no scope of changing the requirements once the project development starts. |
| Agile methodology, follow an iterative development approach because of this planning, development, prototyping and other software development phases may appear more than once. | All the project development phases like designing, development, testing, etc. are completed once in the Waterfall model. |
| Test plan is reviewed after each sprint | The test plan is rarely discussed during the test phase. |
| Agile development is a process in which the requirements are expected to change and evolve. | The method is ideal for projects which have definite requirements and changes not at all expected. |
| In Agile methodology, testing is performed concurrently with software development. | In this methodology, the "Testing" phase comes after the "Build" phase |
| Agile introduces a product mindset where the software product satisfies needs of its end customers and changes itself as per the customer's demands. | This model shows a project mindset and places its focus completely on accomplishing the project. |
| Agile methdology works exceptionally well with Time & Materials or non-fixed funding. It may increase stress in fixed-price scenarios. | Reduces risk in the firm fixed price contracts by getting risk agreement at the beginning of the process. |
| Prefers small but dedicated teams with a high degree of coordination and synchronization. | Team coordination/synchronization is very limited. |
| Products owner with team prepares requirements just about every day during a project. | Business analysis prepares requirements before the beginning of the project. |
| Test team can take part in the requirements change without problems. | It is difficult for the test to initiate any change in requirements. |
| Description of project details can be altered anytime during the SDLC process. | Detail description needs to implement waterfall software development approach. |
| The Agile Team members are interchangeable, as a result, they work faster. There is also no need for project managers because the projects are managed by the entire team | In the waterfall method, the process is always straightforward so, project manager plays an essential role during every stage of SDLC. |

## 2. Difference between SRS document and Design document

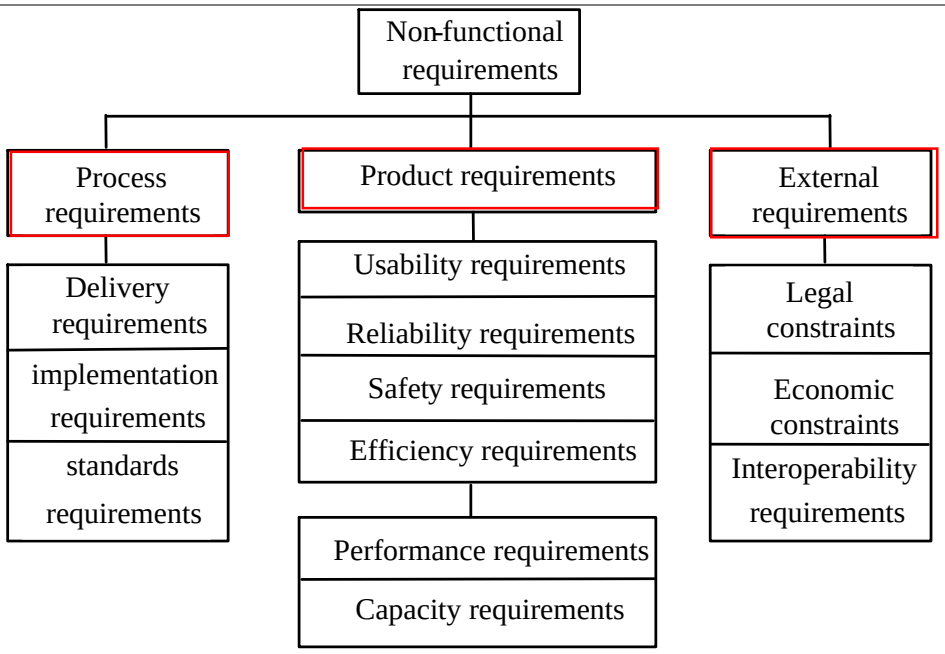
- SRS document is a contract between the development team and the customer.
- A software requirements specification (SRS) is a technical document that describes in detail the externally visible characteristics of a software product. A SRS is not the same as a statement of user needs.
- A requirements specification defines what the customer wants. A design specification defines how you satisfy what the customer wants.
- A Requirement Specification addresses the "what", and a Design Specification addresses the "how". In a perfect world, the Requirement Specification precedes the Design Specification, and each is written by different people with different knowledge and viewpoints.
- Stakeholders communicate the requirements that the product must be designed to meet through the set of requirements in the Requirement Specification. This provides guidance to the design team.
- The Requirement Specification should state what the various stakeholders of the product need in terms of features, functions, performance, constraints, and quality, written in terms of what the product must do or qualities it must have. These stakeholders, including customers, users, maintenance, tech support and others, have the knowledge to define these requirements. Sometimes these stakeholders state their requirements in terms of design, often unintentionally. There are requirement best practices for correcting this problem and for giving the designers the latitude to define the best solution.
- The **Design Specification** (sometimes known as the End-Item Spec) reflects the design and provides directions to the builders and coders of the product. Through this document, designers communicate the design for the product to which the builders or coders must comply.
- The Design Specification should state how the design will meet the requirements. Design is not a one-to-one response to requirements. Design requires discipline knowledge and integration of disciplines in most cases. Design will be concerned with

trade-offs – between different approaches to meet the requirements and concerning issues such as build or buy. The Design Specification will contain information about the product architecture and describe how each component will contribute to meeting the requirements.

- SRS is prepared using BRD at the Requirement Analysis. It is the first legal binding document between the client and the technical team. It focuses on solution that the system should be able to perform to meet the business needs. It should include the following:
  - Introduction
  - System Requirements
  - Functional Requirements
  - External Requirements
  - Non-Functional Requirements
  - Acceptance Criteria
  - Post Implementation
  - Annexure

**3. Differentiate between functional and non-functional requirements.**

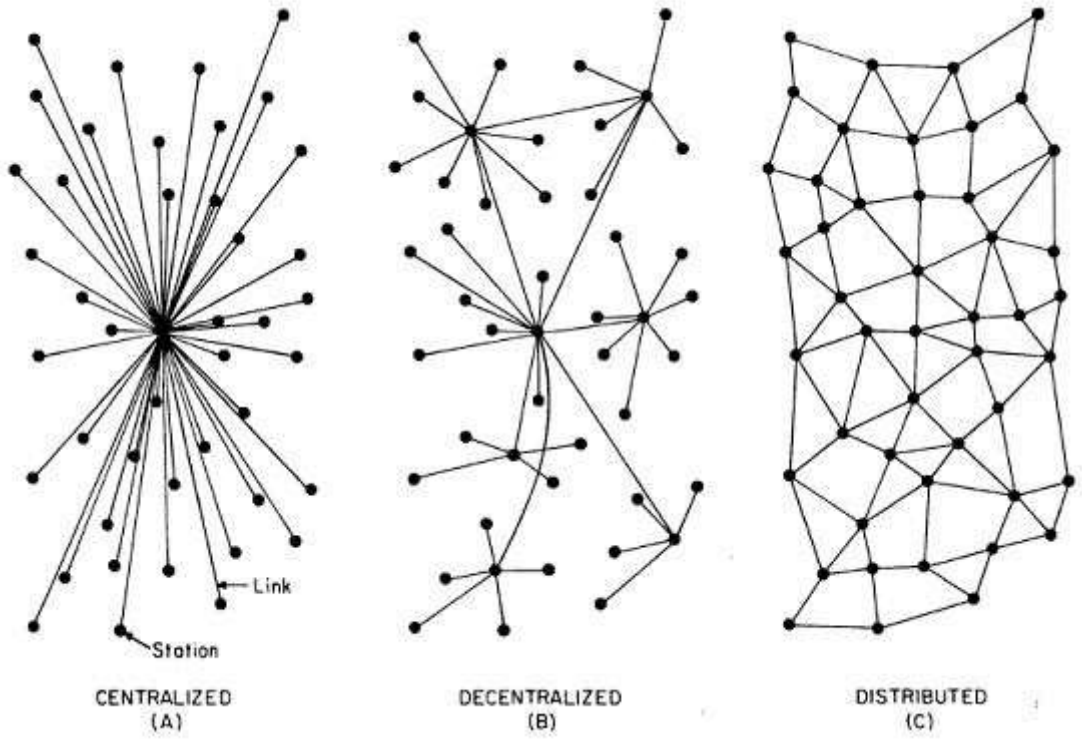
| Functional | Nonfunctional |
|---|---|
| Define functions of a system or its sub systems. | Specify criteria that can be used to judge the operation of the system. |
| Product features | Product properties |
| Describe the work that is done | Describe the character of the work |
| Describe the actions with which the work is concerned | Describe the experience of the user while doing the work |
| Characterized by verbs | Characterized by adjectives |
| Used to describe the functionalities of a system. | Describe the system quality characteristics or the quality attributes. |




- **Product requirements**: Requirements which specify that the delivered product must behave in a particular way i.e. Specify product behaviour e.g. execution speed, reliability, etc.
- **Process requirements**: Requirements which are a consequence of organizational policies and procedures i.e. Derived from policies and procedures e.g. process standards used, implementation requirements, etc.
- **External requirements**: Requirements which arise from factors which are external to the system and its development process i.e. Derived from factors external to the system and its development process e.g. interoperability requirements, legislative requirements, etc.

## 4. Differentiate between distributed software and centralized software system.

| | Centralized systems | Decentralized | Distributed systems |
|---|---|---|---|
| **Points of Failure / Maintenance** | easy to maintain as there is only a single point of failure | have more but still finite | most difficult to maintain. |
| **Fault Tolerance / Stability** | highly unstable. | Kill the leader for a decentralized system and you will have many decentralized systems | very stable and a single failure doesn't do much harm. |
| **Scalability / Max Population** | low scalability | Moderate | Infinite |
| **Ease of development / Creation** | created really fast, you pick up a framework and apply it everywhere | first work out the lower level details like resource sharing (trade) and communications (transport) | first work out the lower level details like resource sharing (trade) and communications (transport) |
| **Evolution / Diversity** | follow a single framework, they don't have diversity and evolve slowly | once the basic infrastructure is in place, evolution is tremendous. | once the basic infrastructure is in place, evolution is tremendous. |




## 5. Differentiate between fat-client and thin-client

| Thin Clients | Thick Clients |
|---|---|
| - Easy to deploy as they require no extra or specialized software installation<br>- Needs to validate with the server after data capture<br>- If the server goes down, data collection is halted as the client needs constant communication with the server<br>- Cannot be interfaced with other equipment (in plants or factory settings for example)<br>- Clients run only and exactly as specified by the server<br>- More downtime<br>-Portability in that all applications are on the server so any workstation can access<br>- Opportunity to use older, outdated PCs as clients<br>- Reduced security threat | - More expensive to deploy and more work for IT to deploy<br>- Data verified by client not server (immediate validation)<br>- Robust technology provides better uptime<br>- Only needs intermittent communication with server<br>- More expensive to deploy and more work for IT to deploy<br>- Require more resources but less servers<br>- Can store local files and applications<br>- Reduced server demands<br>- Increased security issues |

## 6. Differentiate between Verification and validation.

| Criteria | Verification | Validation |
|---|---|---|
| **Definition** | The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase. | The process of evaluating software during or at the end of the development process to determine whether it satisfies specified business requirements. |
| **Objective** | To ensure that the product is being built according to the requirements and design specifications.,In other words, to ensure that work products meet their specified requirements. | To ensure that the product actually meets the user's needs, and that the specifications were correct in the first place.,In other words, to demonstrate that the product fulfils its intended use when placed in its intended environment. |
| **Question** | Are we building the product right? | Are we building the right product? |
| **Evaluation Items** | Plans, Requirement Specs, Design Specs, Code, Test Cases | The actual product/software. |
| - | Execution of code is not comes under Verification | Execution of code is comes under Validation |
| **Cost of Errors** | Cost of errors caught in Verification,is less than errors found in Validation. | Cost of errors caught in Validation is more than errors found in Verification. |
| **Method** | It is basically manually checking the of documents and files like requirement specifications etc. | It is basically checking of developed program based on the requirement specifications documents & files |
| **When Carried Out** | Verification is carried out before the Validation. | Validation activity is carried out just after the Verification. |
| **Process** | Verification process explains whether the outputs are according to inputs or not. | Validation process describes whether the software is accepted by the user or not. |
| **Activities** | Reviews<br>Walkthroughs<br>Inspections | White box testing, Black box testing |

## 7. Software quality and software quality assurance

- Quality software is reasonably bug or defect free, delivered on time and within budget, meets requirements and/or expectations, and is maintainable.
- Key aspects of quality for the customer include:
  - Good design – looks and style
  - Good functionality – it does the job well
  - Reliable – acceptable level of breakdowns or failure
  - Consistency
  - Durable – lasts as long as it should
  - Good after sales service
  - Value for money
- QA is a series of activities that is determined before production begins. These activities start together with a project and take place during the whole cycle. QA ensures that all of the agreed methods, approaches and techniques are implemented without deviations in order to prevent mistakes and, as a result, satisfy the customer's needs. Thus, the main goal of QA is to organize flawless development and to protect the final product from possible defects. It's sometimes deemed the "zero defect" approach.
- SQA helps ensure the development of high-quality software. SQA practices are implemented in most types of software development, regardless of the underlying software development model being used. In a broader sense, SQA incorporates and implements software testing methodologies to test software. Rather than checking for quality after completion, SQA processes test for quality in each phase of development until the software is complete. With SQA, the software development process moves into the next phase only once the current/previous phase complies with the required quality standards.

## 8. QA vs AC

QC deals with the output. The main function of this practice is to verify deliverables and detect mistakes if any, so that a defective solution doesn't reach a customer. It is a reactive technique that determines whether a developed product meets the customer's expectations and conforms to the defined standards. Thus, QC is a final checkpoint before the delivery.

| QA | QC |
|---|---|
| A managing tool | A corrective tool |
| Process-oriented | Product-oriented |
| Proactive strategy | Reactive strategy |
| Prevention of defects | Detection of defects |
| Everyone's responsibility | Testing team's responsibility |

| Performed in parallel with a project | Performed after the final product is ready |
|---|---|
| Quality Assurance Activities:<br>○ Establishing standards<br>○ Project planning<br>○ Internal and external audits<br>○ Process analysis<br>○ Process documentation<br>○ Selection of tracking tools<br>○ Checklist inspection<br>○ Training courses for team members | Quality Control Activities:<br>○ Measurements<br>○ Inspection<br>○ Check analysis<br>○ Manual and automated testing<br>○ Verification and validation<br>○ Random batches control<br>○ Peer reviews |

## 9. Why software configuration management is required for complex software systems?

Software Configuration Management is the process that defines how to control and manage the changes.

The need for an SCM process came when there are many developers and many versions of the software. Suffice to say that in a complex scenario where bug fixing should happen on multiple productions of systems and enhancements must be continued on the main code base, SCM acts as the backbone which can make this happen.

The SCM processes further defines the need to trace the changes and the ability to verify that the final delivered software has all the planned enhancements that are supposed to be part of the release.

The traditional SCM identifies some procedures that must be defined for each software project to ensure that a good SCM process is implemented. Those are following:

- Identification
- Control
- Status Accounting/Monitoring
- Authentication

Most of this section will cover traditional SCM theory. Do not consider this as boring subject since this section defines and explains the terms that will be used throughout this document.

**Identification**

Software is usually made up of several programs. Each program has it's related documentation and data can be called as a "configurable item"(CI). The number of CI in any software project that make up a CI is a decision made of the project. The end product is made up of a bunches of CIs.

The status of the CIs at a given point in time is called as a baseline. The baseline serves as a reference point in the software development life cycle. Each new baseline is the sum total of an older baseline plus a series of approved changes made on the CIs.

**Control**

The process of deciding, coordinating the approved changes for the proposed CIs and implementing all the changes on the appropriate baseline is called Configuration control. It should be kept in mind that configuration controls only address the process after changes are approved. The act of evaluating and approving changes to the software comes under the purview of an entirely different process is called change control.

**Status Accounting/Monitoring**

Configuration status accounting is the keeping process of each release. This procedure involves tracking what all is in each version of software and the changes that lead to this version. Configuration status accounting keeps a record of all the changes made to the previous baseline to reach of the new baseline.

**Authentication**

The reviews and audits that verify the physical existence of CIs and checks that they are correctly recorded and parts list.

Configuration authentication (CA) is a process of assuring that a new baseline has all the planned and approved changes incorporated. The process involves verifying that all the functional aspects of the software is completed and also the completeness of the delivery in terms of the right programs, documentation and data are being delivered.

The configuration authentication is an audit performed on the delivery before it is opened to the entire world.

Free software tools that help in SCM are:

1. Concurrent Versions System (CVS)
2. Revision Control System (RCS)
3. Source Code Control System (SCCS)