**Basic Aspect of software testing**

**Software Testing**

Testing is a Process of Evaluating a System by Manual OR Automatic Means and Verify that it satisfies as specified Requirements or identify differences between Expected and Actual Results.

It can also be stated as the process of veriyfing and validatingthat a software program or application or product:

* Meets the business and technical requirements that guided it’s design and development
* Works as expected
* Can be implemented with the same characteristic.

**Need/Importance of software Testing**

1. Software testing is really required to point out the defects **a**nd errors that were made during the development phases.
2. It’s essential since it makes sure of the Customer’s reliability and their satisfaction in the application.
3. It is very important to ensure the Quality of the product.  Quality product delivered to the customers helps in gaining their confidence.
4. Testing is necessary in order to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.
5. Testing is required for an effective performance of software application or product.
6. It’s important to ensure that the application should not result into any failures because it can be very expensive in the future or in the later stages of the development.
7. It’s required to stay in the business.

**Seven principles of software testing**

**1. Testing shows the presence of bugs**

Testing an application can only reveal that one or more defects exist in the application, however, testing alone cannot prove that the application is error free. Therefore, it is important to design test cases which find as many defects as possible.

2. **Exhaustive testing is impossible**

Testing everything including all combinations of inputs and preconditions is not possible.

3. **Early testing:** In the software development life cycle testing activities should start as early as possible and should be focused on defined objectives.

4. **Defect clustering:** A small number of modules contains most of the defects discovered during pre-release testing or shows the most operational failures.

5. **Pesticide paradox:** If the same kinds of tests are repeated again and again, eventually the same set of test cases will no longer be able to find any new bugs. To overcome this “Pesticide Paradox”, it is really very important to review the test cases regularly and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.

**6.Testing is context dependent:** Testing is basically context dependent. Different kinds of sites are tested differently. For example, safety – critical software is tested differently from an e-commerce site.

7. **Absence – of – errors fallacy:** If the system built is unusable and does not fulfil the user’s needs and expectations then finding and fixing defects does not help.

**Roles and responsibilities of a tester**

**Test lead/manager: A test lead is responsible for:**

* Defining the testing activities for subordinates – testers or test engineers.
* All responsibilities of test planning.
* To check if the team has all the necessary resources to execute the testing activities.
* To check if testing is going hand in hand with the software development in all phases.
* Prepare the status report of testing activities.
* Required Interactions with customers.
* Updating project manager regularly about the progress of testing activities.

**Test engineers/QA testers/QC testers are responsible for:**

* To read all the documents and understand what needs to be tested.
* Based on the information procured in the above step decide how it is to be tested.
* Inform the test lead about what all resources will be required for software testing.
* Develop test cases and prioritize testing activities.
* Execute all the test case and report defects, define severity and priority for each defect.
* Carry out regression testing every time when changes are made to the code to fix defects.

**Difference between manual testing and automation**

|  |  |
| --- | --- |
| **Manual Testing** | **Automation Testing** |
| 1. Manual Testing is a process which is done manually. | 1. Automation Testing is a process which is done by the help of automated tools. |
| 2. All the famous phases of STLC like test planning, test deployment, result analysis, test execution, bug tracking and reporting tools are obviously comes under the category of Manual Testing and done successfully by human efforts. | 2. In Automation Testing all the popular phases of STLC are done by various open sources and purchased tools like Selenium, J meter, QTP, Load Runner, Win Runner and so on. |
| 3. Manual Testing is a start of Testing, without this testing we can’t start Automation Testing. | 3. Automation Testing is a continuous part of Manual Testing. |
| 4. In Manual Testing testers are allowed to do [Random Testing](http://testingbasicinterviewquestions.blogspot.in/2015/02/what-is-random-testing-with-example.html) to find the Bugs. | 4. In Automation Testing we always test through Running Scripts. |
| 5. In Manual Testing we find more bugs than automation by Error Guessing. | 5. In Automation Testing we test the repetitive functionalities of the application. |
| 6. It takes lot of time. | 6. It takes less time. |
| 7. Manual Testing would be run sequentially. | 7. Automation Testing is done on different machines at same time. |
| 8. [Regression Testing process](http://testingbasicinterviewquestions.blogspot.in/2012/01/what-is-regression-testing-explain-it.html) is tough in Manual Testing | 8. Regression Testing process is easy in Automation Testing by Tools. |
| 9. It is not expensive. | 9. It is expensive. |
| 10. More testers are required in Manual Testing because in this testing test cases need to be executed manually. | 10. Few testers are required in Automation Testing because in this testing test cases need to be executed by using Automation Tools. |
| 11. It gives low accuracy result. | 12. It gives high accuracy result. |
| 12. It is considered as low quality. | 12. It is considered as high quality. |
| 13. In this Testing we cannot do batch testing. | 13. In this Testing we can do multiple types of batch testing. |
| 14. It is considered as less reliable. | 14. It is considered as more reliable. |
| 15. No need of programming in Manual Testing. | 15. Need of programming is must in Automation Testing. |
| 16. It is done without interaction of any Tool. | 16. It is always done using tools. |

**Error,defect ,bug, failure**

“A mistake in coding is called Error, error found by tester is called Defect, defect accepted by development team then it is called [Bug](http://www.360logica.com/blog/2012/01/how-to-write-a-good-bug-report-tips-and-tricks.html), build does not meet the requirements then it Is Failure.”

**Error**

An error is a mistake, misconception, or misunderstanding on the part of a software developer.

**Categories of defect**

**Wrong**

 When requirements are implemented not in the right way. This defect is a variance from the given specification. It is Wrong!

**Missing**

A requirement of the customer that was not fulfilled.

**Extra**

A requirement incorporated into the product that was not given by the end customer.

**BUG**

A bug is the result of a coding error. An Error found in the development environment before the product is shipped to the customer.

**FAILURE**

- A failure is the inability of a software system or component to perform its required functions within specified performance requirements. When a defect reaches the end customer it is called a Failure

-a deviation of the software from its expected delivery or service.

**Chances of occurring bugs**

* Error of requirements
* Error of designing
* Programming error
* S/W complexity
* Changing requirement
* Time pressure

# Quality

**QAI**

Quality is much more than absence of defects that allow us to meet customer expectations. Quality requires controlled process improvement, allowing us loyalty in our organizations.

**Five Perspectives of Quality**

* **Transcendent**

I Know it when I see It

* **Product Based**

Possesses Desired Features

* **User Based**

Fitness for Use

* **Development & Manufacturing Based**

Confirms to Requirements

* **Value Based**

At an Acceptable Cost

**Software Quality Factor(SQF)**

* Correctness

System should satisfy the requirements.

* Reliability

System should perform without any malfunction

* Efficiency

Time taken for completing a task

* Integrity

To prevent unauthorized tasks

* Usability

Fit for use

* Maintainability

Effort required for locating and fixing error

* Testability

Testing the program to ensure there is no defect

* Flexibility

Modifying the operations of software

* Reusability

Reusing the part of applications to another.

* Interoperability

Couple one system to another

**Quality Assurance**

* It is a planned and systematic pattern of all actions necessary to provide adequate confidence that the product confirms to established technical requirements that is verifying each and every process. It is a process oriented defect prevention.
* it is defined as an activity that ensures the approaches, techniques, methods and processes designed for the projects are implemented correctly. It recognizes defects in the process.

**Objectives of QA**

* Goal
  + Goal verifies that the system should achieve the objectives of both user and organization
* Method
  + Method verification proves that the structured methodology are standared
* Performance
  + Optimal use of h/w and s/w.

**Roles of QA**

* To establish a process to fix any defect early in and during the product life cycle
* Establish new process and monitor existing process
* Define how defects are mitigated
* Train Organization

**Quality Control**

The purpose of quality control is to identify defects and how them corrected so that defect free product are produced. It is a process of defect detection. It is mainly done by testers.

QC ensures that the approaches, techniques, methods and processes are designed in the project are following correctly. QC activities monitor and verify that the project deliverables meet the defined quality standards

**Difference b/w QA and QC**

|  |  |
| --- | --- |
| **Quality Assurance (QA)** | **Quality Control (QC)** |
| * QA aims to prevent the defect | * QC aims to identify and fix defects |
| * It is a method to manage the quality- Verification | * It is a method to verify the quality-Validation |
| * It does not involve executing the program | * It always involves executing a program |
| * It's a Preventive technique | * It's a Corrective technique |
| * It's a Proactive measure | * It's a Reactive measure |
| * It is the procedure to create the deliverables | * It is the procedure to verify that deliverables |
| * QA involves in full software development life cycle | * QC involves in full software testing life cycle |
| * In order to meet the customer requirements QA defines standards and methodologies | * QC confirms that the standards are followed while working on the product |
| * It is performed before Quality Control | * It is performed only after QA activity is done |

**Verification**

The process of evaluating software to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase.

Verification will help to determine whether the software is of high quality, but it will not ensure that the system is useful. Verification is concerned with whether the system is well-engineered and error-free.

**Validation**

The process of evaluating software during or at the end of the development process to determine whether it satisfies specified requirements.

Validation is the process of evaluating the final product to check whether the software meets the customer expectations and requirements.

**Cost of Quality**

**Costs of quality** or**quality costs** does not mean the use of  expensive or very highly quality materials to manufacture a product. The term refers to the costs that are incurred to prevent, detect and remove defects from products. Quality costs are categorized into four main types. These are:

1. Prevention costs
2. Appraisal costs
3. Internal failure costs and
4. External failure costs.

## Prevention costs:

It is much better to prevent defects rather than finding and removing them from products. The costs incurred to avoid or minimize the number of defects at first place are known as **prevention costs**.

**Examples**

* Improvement of manufacturing processes,
* Workers training, quality engineering,
* Statistical process control etc.

## Appraisal costs:

**Appraisal costs** (also known as **inspection costs**) are those cost that are incurred to identify defective products before they are shipped to customers

## Internal failure costs:

**Internal failure** costs are those costs that are incurred to remove defects from the products before shipping them to customers.

Examples

* cost of rework,
* rejected products etc

## External failure costs:

If defective products have been shipped to customers, external failure costs arise. External failure costs include warranties, replacements, lost sales because of bad reputation, payment for damages arising from the use of defective products etc.

**PROCESS**

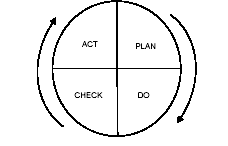
A process can be defined as a set of activities that represent the way work is performed. The outcome from a process is usually a product or service.

**PDCA**

PDCA Cycle is an iterative four-step management method used in business to focus on continuous improvement of processes.

Some of the cases where we use PDCA Cycle are when implementing any changes or when a new improvement project starts or when defining a repetitive process

**PDCA Cycle**



* **PLAN:**

Plan a change (either to solve a problem or to improve some areas) and decide what goal to achieve.

Define the goal and the plan for achieving that goal.

* **DO:**

To design or revise the business requirement as planned

Here we implement the plan (in terms of putting the plan into an action) and test its performance

* **CHECK:**

Evaluate the results to make sure whether we reach the goals as planned

Here we make a checklist to record what went well and what did not work (lessons learnt)

* **ACT:**

If the changes are not as planned then continue the cycle to achieve the goal with a different plan.

Here we take action on what is not working as planned. Task is to keep trying to improve the process with different plan.

**CERTIFICATIONS**

CMMI

Capability Maturity Model Integration (**CMMI**) is a process level improvement training and appraisal program. Administered by the **CMMI** Institute, it was developed at Carnegie Mellon University (CMU).

**CMM's Five Maturity Levels of Software Processes**

* ***Initial Level***: Processes are disorganized, even chaotic. Success is likely to depend on individual efforts, and is not considered to be repeatable, because processes would not be sufficiently defined and documented to allow them to be replicated.
* ***Repeatable Level***:Basic project management techniques are established, and successes could be repeated, because the requisite processes would have been made established, defined, and documented.
* ***Defined Level***: An organization has developed its own standard software process through greater attention to documentation, standardization, and integration.
* ***Managed Level*:** An organization monitors and controls its own processes through data collection and analysis.
* ***Optimizing* Level**: Processes are constantly being improved through monitoring feedback from current processes and introducing innovative processes to better serve the organization's particular needs.

**TMM(TESTING MATURITY MODEL)**

Test Maturity Model is based on the Capability Maturity Model (CMM), and it was first developed by the Illinois Institute of Technology. It is a detailed model for test process improvement.

**SOFTWARE DEVELOPMENT LIFE CYCLE**

There are following six phases in every Software development life cycle model:

1. Requirement gathering and analysis
2. Design
3. Implementation or coding
4. Testing
5. Deployment
6. Maintenance

**1) Requirement gathering and analysis**

 Business requirements are gathered in this phase. Meetings with managers, stake holders and users are held in order to determine the requirements like;

 Who is going to use the system? How will they use the system?  What data should be input into the system?  What data should be output by the system?  These are general questions that get answered during a requirements gathering phase.

Finally, a Requirement Specification document is created which serves the purpose of guideline for the next phase of the model. The testing team follows the Software Testing Life Cycle and starts the [Test Planning](http://istqbexamcertification.com/what-is-the-purpose-and-importance-of-test-plans/) phase after the requirements analysis is completed.

**2)  Design**

**The Design step of the SDLC process can begin when the Customer has approved (signed-off) the Functional Requirements Document .**

In this phase the system and software design is prepared from the requirement specifications which were studied in the first phase.  System Design helps in specifying hardware and system requirements and also helps in defining overall system architecture.

HLD-it gives the architecture of the software product to be developed and is done by architects and senior developers.It covers the system architecture and database design. It describes the relation between various modules and functions of the system. data flow, flow charts and data structures are covered under HLD.

LLD-it is done by senior developers. It describes how each and every feature of product should work and how every component should work.It defines the actual logic for each and every component of the system. Class diagrams with all the methods and relation between classes comes under LLD. Programs specs are covered under LLD.

The outcome form this phase is high level document and low level document.

In this phase the testers comes up with the [Test strategy](http://istqbexamcertification.com/what-are-the-test-approaches-or-strategies-in-software-testing/), where they mention what to test, how to test.

**3)  Implementation / Coding:**

 On receiving system design documents, the work is divided in modules/units and actual coding is started. Developers construct a working software from the approved design. This is the longest phase of the software development life cycle.

**4)**[**Testing**](http://istqbexamcertification.com/what-is-a-software-testing/)**:**

After the code is developed it is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase. During this phase all types of [functional testing](http://istqbexamcertification.com/what-is-functionality-testing-in-software/) like [unit testing](http://istqbexamcertification.com/what-is-unit-testing/), [integration testing](http://istqbexamcertification.com/what-is-integration-testing/), [system testing](http://istqbexamcertification.com/what-is-system-testing/), [acceptance testing](http://istqbexamcertification.com/what-is-acceptance-testing/) are done as well as [non-functional testing](http://istqbexamcertification.com/what-is-non-functional-testing-testing-of-software-product-characteristics/) are also done.

**5)  Deployment:**

After successful testing the product is delivered / deployed to the customer for their use.

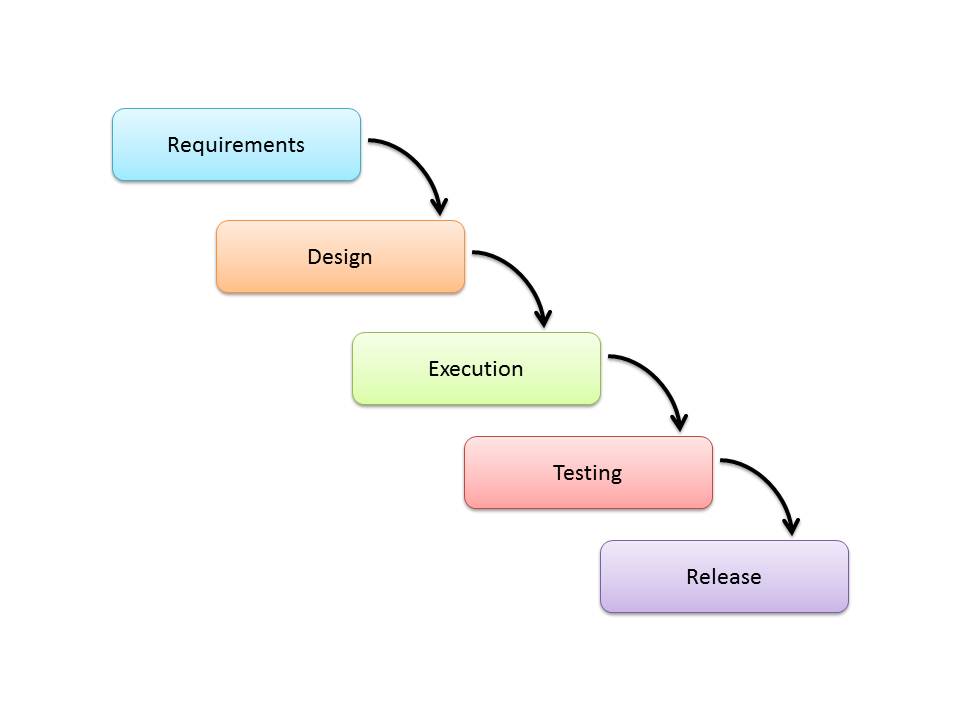
6) **Maintenance:**

Once when the customers starts using the developed system then the actual problems comes up and needs to be solved from time to time. This process where the care is taken for the developed product is known as maintenance.

**Software Development Models**

**Waterfall Model**

The waterfall Model is a linear sequential flow. In which progress is seen as flowing steadily downwards through the phases of software implementation. This means that any phase in the development process begins only if the previous phase is complete. The waterfall approach is the earliest approach and most widely known that was used for software development.



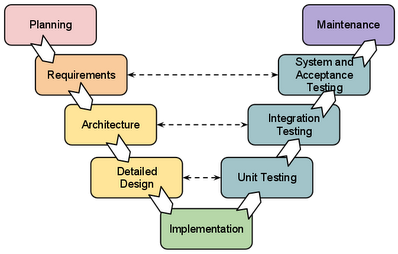
**Usage**

Projects which not focus on changing the requirements, for example, projects initiated from request for proposals ([RFPs](http://en.wikipedia.org/wiki/Request_for_proposal)), the customer has a very clear documented requirements.

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Easy to explain to the users. * Structures approach. * Stages and activities are well defined. * Helps to plan and schedule the project. * Verification at each stage ensures early detection of errors/misunderstanding. * Each phase has specific deliverables. | * Long wait for workable products * no continuous improvement * quality compromised * Very difficult to go back to any stage after it finished. * A little flexibility and adjusting scope is difficult and expensive. * Costly and required more time, in addition to the detailed plan. |

## V-Shaped Model

It is an extension of the waterfall model, Instead of moving down in a linear way, the process steps are bent upwards after the implementation and coding phase, to form the typical V shape. The major difference between V-shaped model and waterfall model is the early test planning in the V-shaped model.



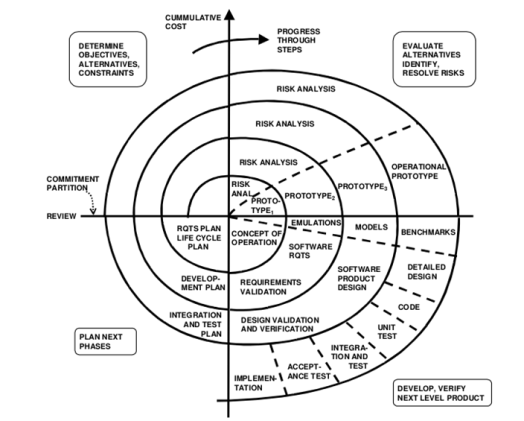
### Usage

* Software requirements clearly defined and known
* Software development technologies and tools are well-known

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Simple and easy to use * Each phase has specific deliverables. * Higher chance of success over the waterfall model due to the development of test plans early on during the life cycle. * Works well for where requirements are easily understood. * Verification and validation of the product in early stages of product development. | * Very inflexible, like the waterfall model. * Adjusting scope is difficult and expensive. * The software is developed during the implementation phase, so no early prototypes of the software are produced. * The model doesn’t provide a clear path for problems found during testing phases. * Costly and required more time, in addition to detailed plan |

## Spiral Model (SDM)

It is combining elements of both design and prototyping-in-stages, in an effort to combine advantages of top-down and bottom-up concepts. The spiral model is favored for large, expensive, and complicated projects. This model uses many of the same phases as the waterfall model, in essentially the same order, separated by planning, risk assessment, and the building of prototypes and simulations.



Usage

It is used in the large applications and systems which built-in small phases or segments.

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Estimates (i.e. budget, schedule, etc.) become more realistic as work progressed because important issues are discovered earlier. * Early involvement of developers. * Manages risks and develops the system into phases. | * High cost and time to reach the final product. * Needs special skills to evaluate the risks and assumptions. * Highly customized limiting re-usability |

## Iterative Model

## In the Iterative model, iterative process starts with a simple implementation of a small set of the software requirements and iteratively enhances the evolving versions until the complete system is implemented and ready to be deployed.

## Development begins by specifying and implementing just part of the software, which is then reviewed to identify further requirements. This process is then repeated, producing a new version of the software at the end of each iteration of the model.

## SDLC Iterative Model

This model is most often used in the following scenarios

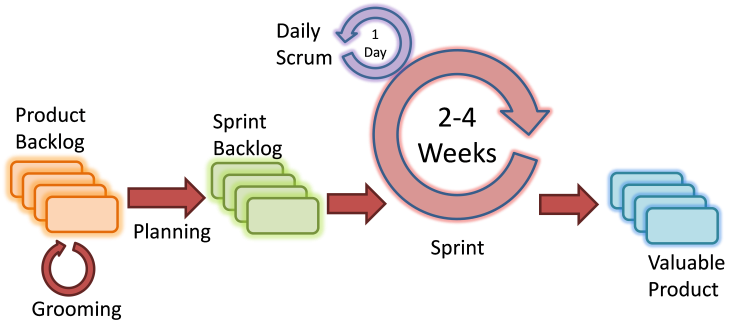
* Requirements of the complete system are clearly defined and understood.
* Major requirements must be defined; however, some functionalities or requested enhancements may evolve with time.
* There is a time to the market constraint.
* A new technology is being used and is being learnt by the development team while working on the project.
* Resources with needed skill sets are not available and are planned to be used on contract basis for specific iterations.
* There are some high-risk features and goals which may change in the future.

## Incremental Model

## In incremental model the whole requirement is divided into various builds. In this model, each module passes through the requirements, design, implementation and [testing](http://istqbexamcertification.com/what-is-a-software-testing/) phases. A working version of software is produced during the first module, so you have working software early on during the [software life cycle](http://istqbexamcertification.com/what-are-the-software-development-life-cycle-phases/).

## Agile Model

It is based on iterative and incremental development, where requirements and solutions evolve through collaboration between cross-functional teams.



**The usage**

It can be used with any type of the project, but it needs more engagement from the customer and to be interactive. Also, it can be used when the customer needs to have some functional requirement ready in less than three weeks and the requirements are not clear enough.

|  |  |
| --- | --- |
| Advantages | Disadvantages |
| * Decrease the time required to avail some system features. * Face to face communication and continuous inputs from customer representative leaves no space for guesswork. * The end result is the high-quality software in the least possible time duration and satisfied customer. | * Scalability. * The ability and collaboration of the customer to express user needs. * Documentation is done at later stages. * Reduce the usability of components. * Needs special skills for the team. |

**Principles of agile methodology**

**1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.**

Customer satisfaction is crucial to a product’s early and ongoing success. This principle emphasizes the importance of a continuous cycle of feedback and improvement. A minimum viable product (MVP) is released to the market and the response informs future releases.

**2. Welcome changing requirements, even late in development. Agile processes harness change for the customer’s competitive advantage.**

Development teams react to issues and change the product to satisfy customer needs. Strategies and processes may be reconsidered to safeguard the product’s quality.

**3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.**

Work on achieving goals on smaller scales, ultimately contributing to the product’s overall completion. Teams have tighter structures and more concrete goals to work towards.

**4. Business people and developers must work together daily throughout the project.**

Agile principles unify different departments, prioritizing regular collaboration and communication to share information/resources.

**5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.**

Appointing the right people with the right skills to the right roles is vital to achieving success with agile principles. They should be trusted to do their job properly, without disruptive micromanagement.

**6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.**

This emphasizes the importance of ongoing collaboration and idea-sharing, with daily meetings, sprint planning, demos, and more.

**7. Working software is the primary measure of progress.**

Development teams work on Minimum Viable Features instead of trying to perfect complete feature sets. Idea testing should be fast, as useful products released now are better than those released a year down the line.

**8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.**

It’s vital for product teams to have realistic goals and manageable expectations during sprints. This aids morale and prevents staff from becoming burned out.

**9. Continuous attention to technical excellence and good design enhances agility.**

Products should be reviewed after each iteration to ensure real improvement is taking place.

**10. Simplicity—the art of maximizing the amount of work not done—is essential.**

Agile is about keeping processes simple and streamlining the entire cycle, and the Agile principles help keep that on track. Even the most minor distractions or unnecessary tasks can slow progress. Embrace automation tools whenever possible.

**11. The best architectures, requirements, and designs emerge from self-organizing teams.**

Teams should be autonomous and capable of acting faster, without having to secure permission on every little task.

**12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.**

Teams should be encouraged to reflect on their progress and make changes to the product, rather than moving ahead blindly.