Internal -1

What is Git?

**Git** is an **open-source distributed version control system**. It is designed to handle minor to major projects with high speed and efficiency. It is developed to co-ordinate the work among the developers. The version control allows us to track and work together with our team members at the same workspace.

Git is foundation of many services like **GitHub** and **GitLab**, but we can use Git without using any other Git services. Git can be used **privately** and **publicly**.

Git was created by **Linus Torvalds** in **2005** to develop Linux Kernel. It is also used as an important distributed version-control tool for **the DevOps**.

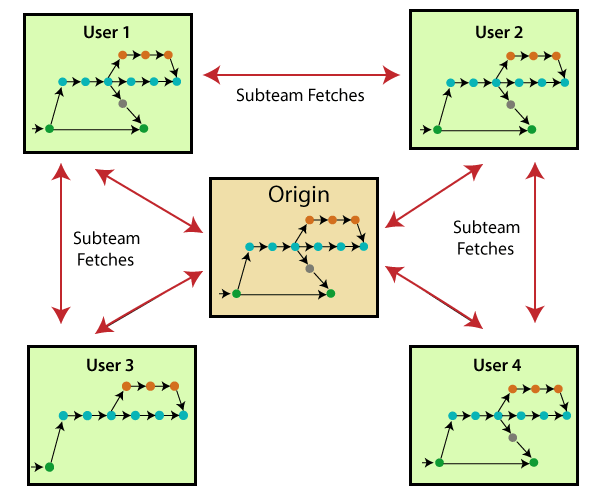
Git is easy to learn, and has fast performance. It is superior to other SCM tools like Subversion, CVS, Perforce, and ClearCase.

Features of Git

Some remarkable features of Git are as follows:



* **Open Source**  
  Git is an **open-source tool**. It is released under the **GPL** (General Public License) license.
* **Scalable**  
  Git is **scalable**, which means when the number of users increases, the Git can easily handle such situations.
* **Distributed**  
  One of Git's great features is that it is **distributed**. Distributed means that instead of switching the project to another machine, we can create a "clone" of the entire repository. Also, instead of just having one central repository that you send changes to, every user has their own repository that contains the entire commit history of the project. We do not need to connect to the remote repository; the change is just stored on our local repository. If necessary, we can push these changes to a remote repository.



* **Security**  
  Git is secure. It uses the **SHA1 (Secure Hash Function)** to name and identify objects within its repository. Files and commits are checked and retrieved by its checksum at the time of checkout. It stores its history in such a way that the ID of particular commits depends upon the complete development history leading up to that commit. Once it is published, one cannot make changes to its old version.
* **Speed**  
  Git is very **fast**, so it can complete all the tasks in a while. Most of the git operations are done on the local repository, so it provides a **huge speed**. Also, a centralized version control system continually communicates with a server somewhere.  
  Performance tests conducted by Mozilla showed that it was **extremely fast compared to other VCSs**. Fetching version history from a locally stored repository is much faster than fetching it from the remote server. The **core part of Git**is **written in C**, which **ignores** runtime overheads associated with other high-level languages.  
  Git was developed to work on the Linux kernel; therefore, it is **capable** enough to **handle large** **repositories** effectively. From the beginning, **speed** and **performance** have been Git's primary goals.
* **Supports non-linear development**  
  Git supports **seamless branching and merging**, which helps in visualizing and navigating a non-linear development. A branch in Git represents a single commit. We can construct the full branch structure with the help of its parental commit.
* **Branching and Merging**  
  **Branching and merging** are the **great feature**s of Git, which makes it different from the other SCM tools. Git allows the **creation of multiple branches** without affecting each other. We can perform tasks like **creation**, **deletion**, and **merging** on branches, and these tasks take a few seconds only. Below are some features that can be achieved by branching:
  + We can **create a separate branch** for a new module of the project, commit and delete it whenever we want.
  + We can have a **production branch**, which always has what goes into production and can be merged for testing in the test branch.
  + We can create a **demo branch** for the experiment and check if it is working. We can also remove it if needed.
  + The core benefit of branching is if we want to push something to a remote repository, we do not have to push all of our branches. We can select a few of our branches, or all of them together.
* **Data Assurance**  
  The Git data model ensures the **cryptographic integrity** of every unit of our project. It provides a **unique commit ID** to every commit through a **SHA algorithm**. We can **retrieve** and **update** the commit by commit ID. Most of the centralized version control systems do not provide such integrity by default.
* **Staging Area**  
  The **Staging area** is also a **unique functionality** of Git. It can be considered as a **preview of our next commit**, moreover, an **intermediate area** where commits can be formatted and reviewed before completion. When you make a commit, Git takes changes that are in the staging area and make them as a new commit. We are allowed to add and remove changes from the staging area. The staging area can be considered as a place where Git stores the changes.  
  Although, Git doesn't have a dedicated staging directory where it can store some objects representing file changes (blobs). Instead of this, it uses a file called index.



Another feature of Git that makes it apart from other SCM tools is that **it is possible to quickly stage some of our files and commit them without committing other modified files in our working directory.**

* **Maintain the clean history**  
  Git facilitates with Git Rebase; It is one of the most helpful features of Git. It fetches the latest commits from the master branch and puts our code on top of that. Thus, it maintains a clean history of the project.

# **What is GitHub?**

GitHub is a Git repository hosting service. GitHub also facilitates with many of its features, such as access control and collaboration. It provides a Web-based graphical interface.

GitHub is an American company. It hosts source code of your project in the form of different programming languages and keeps track of the various changes made by programmers.

It offers both **distributed version control and source code management (SCM)** functionality of Git. It also facilitates with some collaboration features such as bug tracking, feature requests, task management for every project.



## Features of GitHub

GitHub is a place where programmers and designers work together. They collaborate, contribute, and fix bugs together. It hosts plenty of open source projects and codes of various programming languages.Fullscreen

Backward Skip 10sPlay VideoForward Skip 10s

Some of its significant features are as follows.

* Collaboration
* Integrated issue and bug tracking
* Graphical representation of branches
* Git repositories hosting
* Project management
* Team management
* Code hosting
* Track and assign tasks
* Conversations
* Wikisc

## Benefits of GitHub

GitHub can be separated as the Git and the Hub. GitHub service includes access controls as well as collaboration features like task management, repository hosting, and team management.

The key benefits of GitHub are as follows.

* It is easy to contribute to open source projects via GitHub.
* It helps to create an excellent document.
* You can attract recruiter by showing off your work. If you have a profile on GitHub, you will have a higher chance of being recruited.
* It allows your work to get out there in front of the public.
* You can track changes in your code across versions.

# **Git Version Control System**

A version control system is a software that tracks changes to a file or set of files over time so that you can recall specific versions later. It also allows you to work together with other programmers.

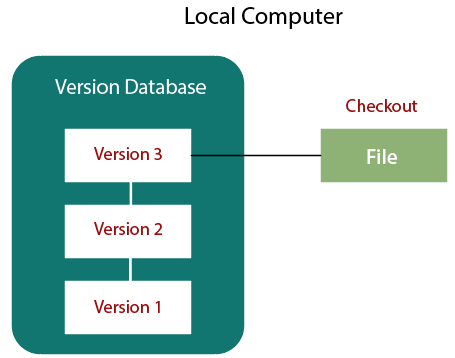
The version control system is a collection of software tools that help a team to manage changes in a source code. It uses a special kind of database to keep track of every modification to the code.

Developers can compare earlier versions of the code with an older version to fix the mistakes.

## Types of Version Control System

* Localized version Control System
* Centralized version control systems
* Distributed version control systems

### **Localized Version Control Systems**



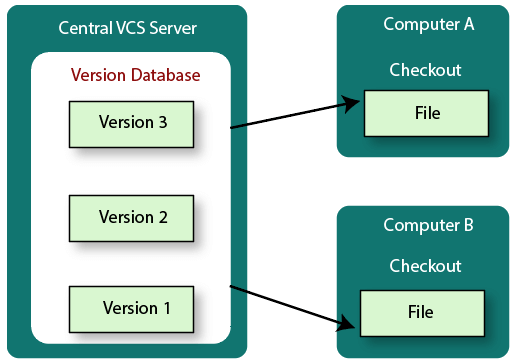
The localized version control method is a common approach because of its simplicity. But this approach leads to a higher chance of error. In this approach, you may forget which directory you're in and accidentally write to the wrong file or copy over files you don't want to.

To deal with this issue, programmers developed local VCSs that had a simple database. Such databases kept all the changes to files under revision control. A local version control system keeps local copies of the files.

The major drawback of Local VCS is that it has a single point of failure.

### **Centralized Version Control System**

The developers needed to collaborate with other developers on other systems. The localized version control system failed in this case. To deal with this problem, Centralized Version Control Systems were developed.



These systems have a single server that contains the versioned files, and some clients to check out files from a central place.

Centralized version control systems have many benefits, especially over local VCSs.

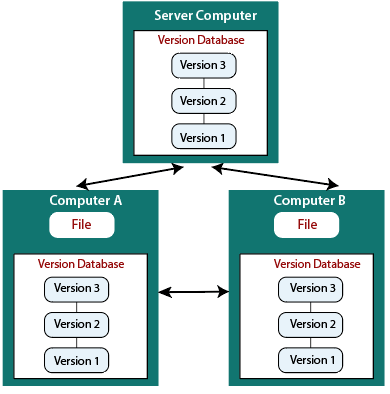
* Everyone on the system has information about the work what others are doing on the project.
* Administrators have control over other developers.
* It is easier to deal with a centralized version control system than a localized version control system.
* A local version control system facilitates with a server software component which stores and manages the different versions of the files.

It also has the same drawback as in local version control system that it also has a single point of failure.

### **Distributed Version Control System**

Centralized Version Control System uses a central server to store all the database and team collaboration. But due to single point failure, which means the failure of the central server, developers do not prefer it. Next, the Distributed Version Control System is developed.

In a Distributed Version Control System (such as Git, Mercurial, Bazaar or Darcs), the user has a local copy of a repository. So, the clients don't just check out the latest snapshot of the files even they can fully mirror the repository. The local repository contains all the files and metadata present in the main repository.



DVCS allows automatic management branching and merging. It speeds up of most operations except pushing and pulling. DVCS enhances the ability to work offline and does not rely on a single location for backups. If any server stops and other systems were collaborating via it, then any of the client repositories could be restored by that server. Every checkout is a full backup of all the data.

These systems do not necessarily depend on a central server to store all the versions of a project file.

## Difference between Centralized Version Control System and Distributed Version Control System

Centralized Version Control Systems are systems that use **client/server** architecture. In a centralized Version Control System, one or more client systems are directly connected to a central server. Contrarily the Distributed Version Control Systems are systems that use **peer-to-peer** architecture.

There are many benefits and drawbacks of using both the version control systems. Let's have a look at some significant differences between Centralized and Distributed version control system.

|  |  |
| --- | --- |
| **Centralized Version Control System** | **Distributed Version Control System** |
| In CVCS, The repository is placed at one place and delivers information to many clients. | In DVCS, Every user has a local copy of the repository in place of the central repository on the server-side. |
| It is based on the client-server approach. | It is based on the client-server approach. |
| It is the most straightforward system based on the concept of the central repository. | It is flexible and has emerged with the concept that everyone has their repository. |
| In CVCS, the server provides the latest code to all the clients across the globe. | In DVCS, every user can check out the snapshot of the code, and they can fully mirror the central repository. |
| CVCS is easy to administrate and has additional control over users and access by its server from one place. | DVCS is fast comparing to CVCS as you don't have to interact with the central server for every command. |
| The popular tools of CVCS are **SVN** (Subversion) and **CVS**. | The popular tools of DVCS are **Git** and **Mercurial**. |
| CVCS is easy to understand for beginners. | DVCS has some complex process for beginners. |
| If the server fails, No system can access data from another system. | if any server fails and other systems were collaborating via it, that server can restore any of the client repositories |

## What is Agile Methodology?

An agile methodology is an iterative approach to software development. Each iteration of agile methodology takes a short time interval of 1 to 4 weeks. The agile development process is aligned to deliver the changing business requirement. It distributes the software with faster and fewer changes.

The single-phase software development takes 6 to 18 months. In single-phase development, all the requirement gathering and risks management factors are predicted initially.

The agile software development process frequently takes the feedback of workable product. The workable product is delivered within 1 to 4 weeks of iteration.

# **Agile Manifesto**

In February 2001, at the Snowbird resort in Utah, a team of 17 software developers met to discuss lightweight development methods. The result of their meeting was the following Agile Manifesto for software development:-

We are uncovering the better ways of developing software by doing it and helping others to do it. Through this meeting, we have come to value -

* Individuals and interactions over Processes and tools.
* Working software over comprehensive documentation.
* Customers are collaboration over contact negotiation.
* Responding to change over following a plan.

So that, while there is value in the items on the right, we value the items on the left more.

## The Twelve Principle of Agile Manifesto

1. **Customer Satisfaction:** Manifesto provides high priority to satisfy the costumer's requirements. This is done through early and continuous delivery of valuable software.
2. **Welcome Change:** Making changes during software development is common and inevitable. Every changingrequirement should be welcome, evenin the late development phase. Agile process works to increase the customers' competitive advantage.
3. **Deliver the Working Software:** Deliver the working software frequently, ranging from a few weeks to a few months with considering the shortest timeperiod.
4. **Collaboration:** Business people (Scrum Master and Project Owner) and developers must work together during the entire life of a project development phase.
5. **Motivation:** Projects should be build around motivated team members. Provide such environment that supportsindividual team members and trust them. It makes them feel responsible for gettingthe job donethoroughly.
6. **Face-to-face Conversation:** Face-to-face conversation betweenScrum Master anddevelopment team and between the Scrum Master and customers for the most efficient and effective method of conveying information to and within a development team.
7. **Measure the Progress as per the Working Software:** The working software is the key and primary measure of the progress.
8. **Maintain Constant Pace:** The aim of agile development is sustainable development. All the businesses and users should be able to maintain a constant pace with the project.
9. **Monitoring:** Pay regular attention to technical excellence and good design to maximize agility.
10. **Simplicity:** Keep things simple and use simple terms to measure the work that is not completed.
11. **Self-organized Teams:** The Agile team should be self-organized. They should not be depending heavily on other teams because the best architectures, requirements, and designs emerge from self-organized teams.
12. **Review the Work Regularly:** The work should be reviewed at regular intervals, so that the team canreflect on how to become more productive and adjust its behavior accordingly.

## 12 Agile Manifesto Principles

### **1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.** You may concentrate on the project's primary goal—delivering what the customer wants, not what you planned—by cutting down on time it takes between the project documentation, reporting to your client, and then obtaining feedback.

### **2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.** It is time consuming to handle large and complicated work while managing project activities. Therefore, a better strategy is to break the task into manageable, sizeable chunks. In addition, it would be simpler for the team members to see possible bottlenecks and deal with delays if the clients were always kept informed.

### **3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference for a shorter timescale.** According to the Agile methodology, working software is frequently delivered in a shorter amount of time. Team members must consistently raise their performance standards as a result of this iterative process.

### **4. Business people and developers must work together daily throughout the project.** In order to ensure that the business and development sides of the project can communicate effectively and, more importantly, collaborate, a bridge between them must be built. To facilitate an intellectual exchange that both parties can agree on, make use of the same tools you would have used in managing remote teams.

### **5. Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.** The project manager must establish a supportive and stimulating environment where team members are free to express their ideas and make recommendations for enhancing the output of the group. This results in a massive improvement in their general performance, eventually aiding the project.

### **6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.** Efficient communication among the parties concerned is stressed strongly in the Agile manifesto. Thanks to improvements in communication technologies, it's now simpler. Instead of having a quick conference in the office, all participants can now meet via video conferencing.

### **7. Working software is the primary measure of progress.** Delivering a functional product that pleases the consumer is the single determinant that can guarantee success. Before Agile, numerous success metrics decreased the quality of the finished product.

### **8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.** Burnout will occur if you work on a project for a long time. It's inevitable. Avoid placing too much of a workload on your employees. The value of your project will be affected. So, assemble the best team for the job that will work hard but refrain from

### **9. Continuous attention to technical excellence and good design enhances agility.** Any Agile team's main goal should be to provide value to the client. Therefore, a multi-skilled team that can manage all the project's technical components and offers the chance for continual improvement is crucial.

### **10. Simplicity — the art of maximizing the amount of work not done — is essential.** You should avoid adding extraneous complexity to a project if you want to complete it swiftly. You can accomplish this in various ways, including by using agile tools, which eliminate busywork and offer you more significant influence over all project-related decisions.

### **11. The best architectures, requirements, and designs emerge from self-organizing teams.** Simply said, a self-organized workforce with decision-making autonomy would function better since each team member would be responsible for meeting client expectations rather than a lone project manager.

### **12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.** Agile techniques are constructed on the notion of iteration, where teams consistently enhance their game by learning from their previous wrongdoings. Project managers should inspire team meetings where everyone evaluates their work and discusses how to develop their management and technical skills.

# **What is Scrum Framework?**

**Scrum is a framework** that helps agile teams to work together. Using it, the team members can deliver and sustain the complex product. It encourages the team to learn through practice, self-organize while working on the problem. Scum is a work done through the framework and continuously shipping values to customers.

It is the most frequent software that is used by the development team. Its principle and lessons can be applied to all kinds of teamwork. Its policy and experiences is a reason of popularity of Scrum framework. The Scrum describes a set of tools, meetings, and roles that help the teams structure. It also manages the work done by the team

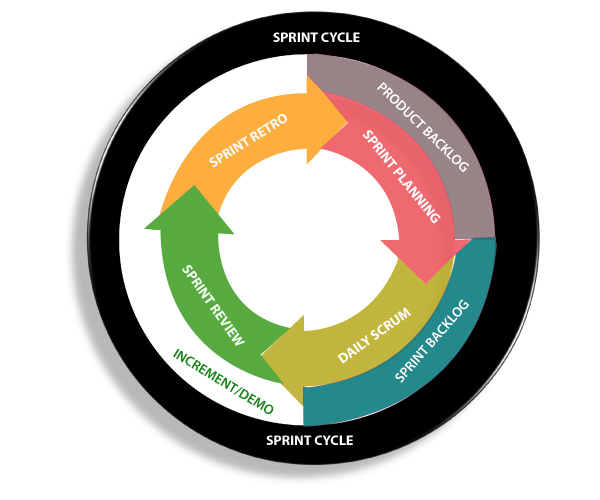
## The framework

Scrum and agile are not the same thing because Scrum focused on continuous improvement, which is a core foundation of agile. Scrum framework focuses on ongoing getting work done.

## What are sprints?

With scrum, a product is built in a series of repetition called **sprints**. It breaks down big complex projects into bite-size pieces. It makes projects more manageable, allows teams to ship high quality, work faster, and more frequently. The sprints give them more flexibility to adapt to the changes.DIFFERENT! Soundcore Motion X600

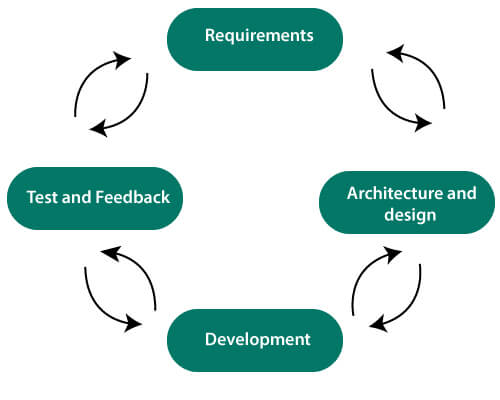
Sprints are a short, time-boxed period for Scrum team that works to complete a set amount of work. Sprints are the core component of Scrum and agile methodology. The right sprints will help our agile team to ship better software.



# **Agile Software Development Life Cycle (SDLC)**

**Software development life cycle (SDLC)** is a phenomenon to **design**, **develop** and, **test** high-quality software. The primary aim of SDLC is to produce high-quality software that fulfills the customer requirement within times and cost estimates.

**Agile Software Development Life Cycle (SDLC)** is the combination of both iterative and incremental process models. It focuses on process adaptability and customer satisfaction by rapid delivery of working software product. Agile SDLC breaks down the product into small incremental builds. These builds are provided into iterations.



In the agile SDLC development process, the customer is able to see the result and understand whether he/she is satisfied with it or not. This is one of the advantages of the agile SDLC model. One of its disadvantages is the absence of defined requirements so, it is difficult to estimate the resources and development cost.

**Each iteration of agile SDLC consists of cross-functional teams working on various phases:**

1. Requirement gathering and analysis
2. Design the requirements
3. Construction/ iteration
4. Deployment
5. Testing
6. Feedback

**Requirements gathering and analysis** In this phase, you must define the requirements. You should explain business opportunities and plan the time and effort needed to build the project. Based on this information, you can evaluate technical and economic feasibility.

**Design the requirements** When you have identified the project, work with stakeholders to define requirements. You can use the user flow diagram or the high-level UML diagram to show the work of new features and show how it will apply to your existing system.

**Construction/ Iteration** When the team defines the requirements, the work begins. The designers and developers start working on their project. The aims of designers and developers deploy the working product within the estimated time. The product will go into various stages of improvement, so it includes simple, minimal functionality.

### **Deployment**

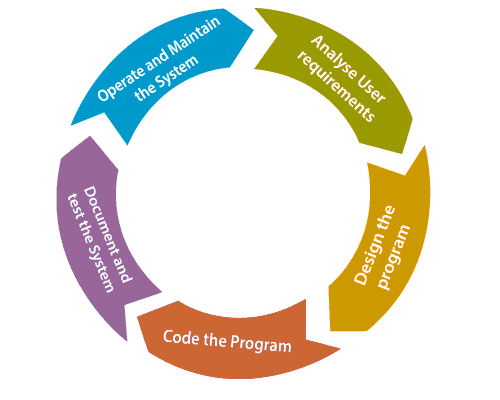
### In this phase, the team issues a product for the user's work environment.

### **Testing**

### In this phase, the Quality Assurance team examine the product's performance and look for the bug.

### **Feedback**

### After releasing of the product, the last step is to feedback it. In this step, the team receives feedback about the product and works through the feedback.



## Agile SDLC Process Flow

1. **Concept:** Project are imagined and prioritized.
2. **Inception:** Team members are created, funding is put in place, and basic environments and requirements are discussed.
3. **Iteration/Constriction:** The software development team works to deliver working software. It is based on requirement and feedback.
4. **Release:** Perform quality assurance (QA) testing, provides internal and external training, documentation development, and final version of iteration into the product.
5. **Production:** It is ongoing support of the software.

## Advantages of Agile SDLC

1. Project is divided into short and transparent iterations.
2. It has a flexible change process.
3. It minimizes the risk of software development.
4. Quick release of the first product version.
5. The correctness of functional requirement is implemented into the development process.
6. Customer can see the result and understand whether he/she is satisfied with it or not.

## Disadvantages of Agile SDLC

1. The development team should be highly professional and client-oriented.
2. New requirement may be a conflict with the existing architecture.
3. With further correction and change, there may be chances that the project will cross the expected time.
4. There may be difficult to estimate the final coast of the project due to constant iteration.
5. A defined requirement is absent.

**Diagram agile methodology**



**Explain crystal agile methodology.**

**Crystal methods in Agile Development/Framework:**The crystal method is an agile framework that is considered a lightweight or agile methodology that focuses on individuals and their interactions. The methods are color-coded to significant risk to human life. It is mainly for short-term projects by a team of developers working out of a single workspace. Among a few Agile [Software Development Life Cycle (SDLC)](https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/) models crystal is considered as one of the Agile SDLC models.  
Two core beliefs of the Crystal method:

* Find your own way and methods to optimize workflow.
* Make use of unique methods to make the project unique and dynamic.

**Let’s know about the history of the Crystal Method**: The crystal method was developed by an American scientist named Alistair Cockburn who worked at IBM. He decided not to focus on step-by-step developmental strategies, but to develop team collaboration and communication. Some of the traits of Cockburn’s Crystal method were:

* Human-powered i.e. the project should be flexible and people involved in preferred work.
* Adaptive i.e. approaches don’t have any fixed tools but can be changed anytime to meet the team’s specific needs.
* Ultra-light i.e. this methodology doesn’t require much documentation.

**Properties of Crystal Agile Framework:**

1. **Frequent Delivery-** It allows you regularly deliver the products and test code to real users. Without this, you might build a product that nobody needs.
2. **Reflective Improvement-**No matter how good you have done or how bad you have done. Since there are always areas where the product can be improved, so the teams can implement to improve their future practices.
3. **Osmotic Communication-**Alistair stated that having the teams in the same physical phase is very much important as it allows information to flow in between members of a team as in osmosis.
4. **Personal Safety-**There are no bad suggestions in a crystal team, team members should feel safe to discuss ideas openly without any fear.
5. **Focus-**Each member of the team knows exactly what to do, which enables them to focus their attention. This boosts team interaction and works towards the same goal.
6. **Easy access to expert users-**It enhances team communication with users and gets regular feedback from real users.
7. **Technical tooling-** It contains very specific technical tools which to be used by the software development team during testing, management, and configuration. These tools make it enable the team to identify any error within less time.
8. **Continuous learning –** The framework emphasizes on continuous learning, enabling team members to acquire new skills and knowledge, and apply them in their work.
9. **Teamwork –** The framework stresses on the importance of teamwork, promoting collaboration, and mutual support among team members.
10. **Timeboxing –**The framework adopts timeboxing to manage project deadlines, ensuring that the team delivers within set timelines.
11. **Incremental development –**The framework promotes incremental development, enabling the team to deliver working software frequently, and adapt to changes as they arise.
12. **Automated testing –**The framework emphasizes on automated testing, enabling the team to detect and fix bugs early, reducing the cost of fixing errors at later stages.
13. **Customer involvement –** The framework emphasizes on involving customers in the development process, promoting customer satisfaction, and delivering products that meet their needs.
14. **Leadership –** The framework encourages leadership, enabling team members to take ownership of their work and make decisions that contribute to the success of the project.

**How does Crystal function?**

Till now, we got to know that crystal is a family of various developmental approaches, and it is not a group of prescribed developmental tools and methods. In the beginning, the approach is set by considering the business requirements and the needs of the project. Various methodologies in the Crystal family also known as weights of the Crystal approach are represented by different colors of the spectrum.  
Crystal family consists of many variants like Crystal Clear, Crystal Yellow, Crystal Red, Crystal Sapphire, Crystal Red, Crystal Orange Web, and Crystal Diamond.

1. **Crystal Clear-**The team consists of only 1-6 members that is suitable for short-term projects where members work out in a single workspace.
2. **Crystal Yellow-** It has a small team size of 7-20 members, where feedback is taken from Real Users. This variant involves automated testing which resolves bugs faster and reduces the use of too much documentation.
3. **Crystal Orange-** It has a team size of 21-40 members, where the team is split according to their functional skills. Here the project generally lasts for 1-2 years and the release is required every 3 to 4 months.
4. **Crystal Orange Web-**It has also a team size of 21-40 members were the projects that have a continually evolving code base that is being used by the public. It is also similar to Crystal Orange but here they do not deal with a single project but a series of initiatives that required programming.
5. **Crystal Red-** The software development is led by 40-80 members where the teams can be formed and divided according to requirements.
6. **Crystal Maroon-** It involves large-sized projects where the team size is 80-200 members and where methods are different and as per the requirement of the software.
7. **Crystal Diamond & Sapphire-** This variant is used in large projects where there is a potential risk to human life.

The below figure illustrates about crystal team



*CRYSTAL FAMILY (TEAM MEMBERS)*

**Benefits of using the Crystal Agile Framework :**

* Facilitate and enhance team communication and accountability.
* The adaptive approach lets the team respond well to the demanding requirements.
* Allows team to work with the one they see as the most effective.
* Teams talk directly with each other, which reduces management overhead.
* **Faster delivery –** The framework enables the team to deliver working software faster, which can help gain a competitive advantage in the market.
* **Higher quality –**The framework emphasizes on quality, enabling the team to detect and fix defects early in the development process, resulting in a higher quality product.
* **Improved customer satisfaction –** The framework promotes customer involvement, enabling the team to deliver products that meet customer needs, resulting in higher customer satisfaction.
* **Increased productivity –**The framework enables the team to focus on delivering the highest value features, which can increase productivity and reduce waste.
* **Flexibility –**The framework is highly adaptable, enabling the team to adjust to changing requirements, and make decisions based on real-time feedback.
* **Empowerment –** The framework promotes empowerment, enabling team members to take ownership of their work, and make decisions that contribute to the success of the project.
* **Reduced risk –** The framework promotes risk management, enabling the team to identify and mitigate potential risks early in the development process, reducing the likelihood of project failure.