IT24: Software Project Management

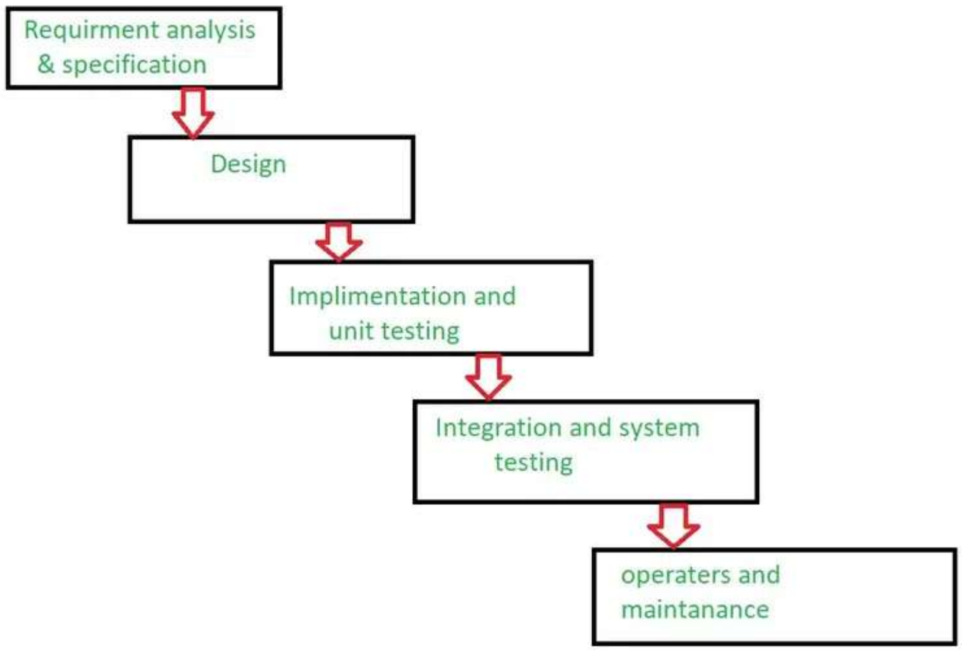
Chapter 1: Introduction to Agile Project Management

Difference between Traditional and Agile Software Development

Traditional Software Development and Agile Software Development are the ways of the designing and developing system software. Both are important types of the software designing.

Traditional Software Development

Traditional Software Development is the software development process used to design and develop simple software. It is used when the security and many other factors of the software are not much important. It is used by freshers to develop the software. It consists of five phases:

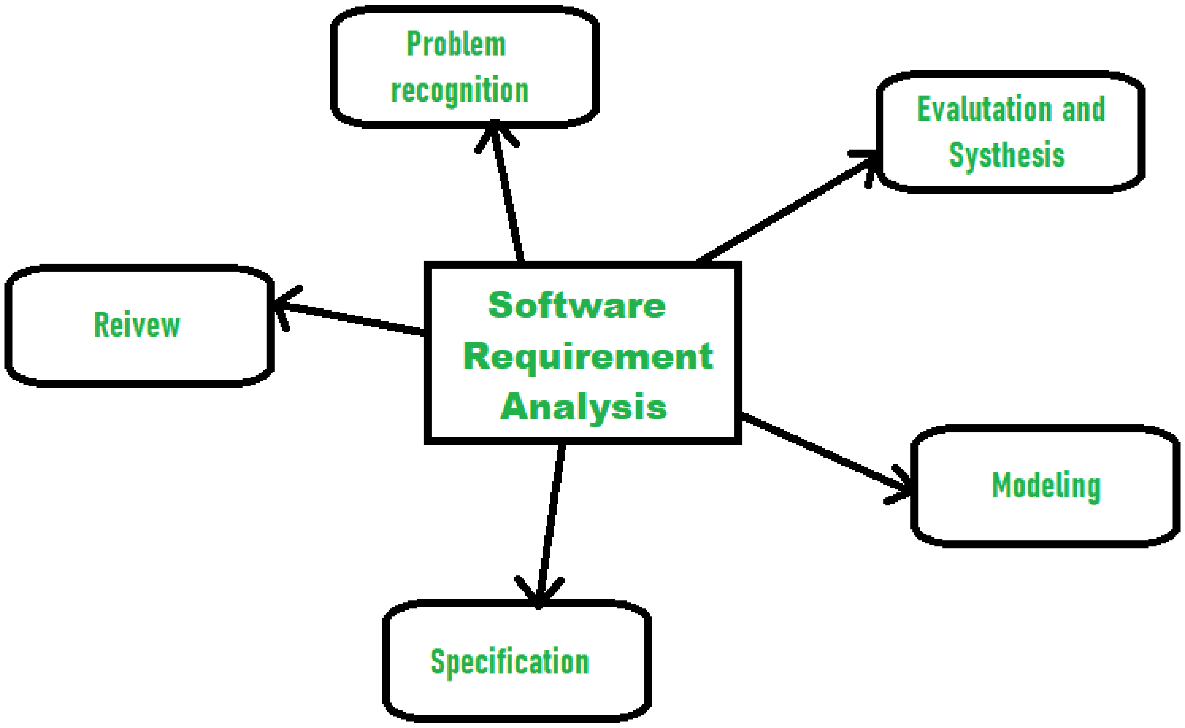

Requirment analysis & specification
Design
Implimentation and unit testing
Integration and system testing
operaters and maintanance

Software requirement means requirement that is needed by software to increase quality of software product. These requirements are generally a type of expectation of user from software product that is important and need to be fulfilled by software. Analysis means to examine something in an organized and specific manner to know complete details about it.

Therefore, **Software requirement analysis** simply means complete study, analyzing, and describing software requirements so that genuine and needed requirements can be fulfilled to solve problems. There are several activities involved in analyzing Software requirements. Some of them are given below :

IT24: Software Project Management

Chapter 1: Introduction to Agile Project Management



1. **Problem Recognition :**
   The main aim of requirement analysis is to fully understand main objective of requirement that includes why it is needed, does it add value to product, will it be beneficial, does it increase quality of the project, does it will have any other effect. All these points are fully recognized in problem recognition so that requirements that are essential can be fulfilled to solve business problems.

2. **Evaluation and Synthesis :**
   Evaluation means judging whether something is worth it or not and synthesis means creating or forming something. Here are some tasks are given that are important in the evaluation and synthesis of software requirements:

   - To define all functions of software that necessary.
   - To define all data objects present externally and easily observable.
   - To evaluate the flow of data is worth or not.
   - To fully understand the overall behavior of a system means the overall working of the system.
   - To identify and discover constraints that are designed.

- To define and establish the character of the system interface to fully understand how a system interacts with two or more components or with one another.

3. **Modeling :**
   After complete gathering of information from above tasks, functional and behavioral models are established after checking function and behavior of system using a domain model that also known as the conceptual model.

4. **Specification :**
   The software requirement specification (SRS) , which specifies whether the requirement is functional or non-functional, should be developed.

5. **Review :**
   After developing the SRS, it must be reviewed to check whether it can be improved or not and must be refined to make it better and increase the quality.

Advantages of Traditional Software Development

- **Well-Established Methodology:** Traditional software development follows a well-established methodology that is widely understood and documented.
- **Clear Requirements:** Traditional software development relies on clear and detailed requirements, which helps to ensure that the final product meets the customer's needs.
- **Structured Approach:** Traditional software development follows a structured approach, with clear phases and milestones, which helps to ensure that the project stays on track.
- **Proven Success:** Traditional software development has a proven track record of success and is widely used in many industries.
- **Quality Control:** Traditional software development typically includes extensive testing and quality control processes, which helps to ensure that the final product is of high quality.

Disadvantages of Traditional Software Development

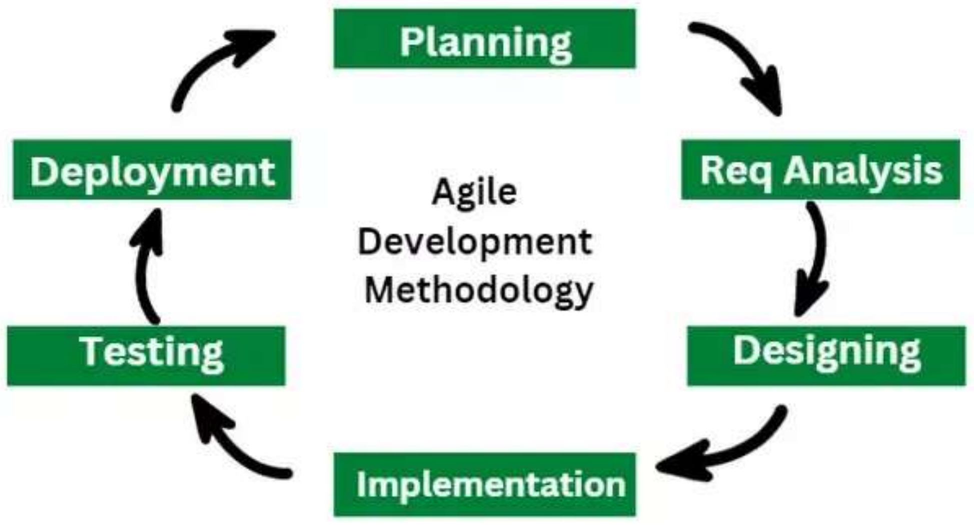
- **Slow Process:** Traditional software development can be a slow process, with lengthy planning and design phases.
- **Lack of Flexibility:** Traditional software development can be inflexible, with changes to requirements or design difficult to implement once development has begun.
- **High Cost:** Traditional software development can be expensive, particularly if the project is large or complex.

- **Limited Customer Involvement:** Traditional software development often limits customer involvement to the planning and design phases, which can result in a product that does not fully meet their needs.
- **Limited Innovation:** Traditional software development can be conservative and risk-averse, which can limit innovation and the development of new ideas.

Agile Software Development

Agile Software Development is the software development process used to design complicated software. It is used when the software is quite sensitive and complicated. It is used when security is much more important. It is used by professionals to develop the software. It consists of three phases:

1. Planning
2. Requirement Analysis
3. Designing
4. Implementation
5. Testing
6. Deployment



Advantages of Agile Software Development

- **Flexibility:** Agile software development is highly flexible and can easily adapt to changes in requirements, design, and scope.
- **Customer Involvement:** Agile software development encourages frequent customer involvement, which can result in a final product that better meets their needs.
- **Continuous Delivery:** Agile software development typically includes continuous delivery, which means that working software is delivered to the customer on a regular basis.

Chapter 1: Introduction to Agile Project Management

- **Collaboration:** Agile software development emphasizes collaboration between team members, which can lead to better communication and problem-solving.
- **Early and Frequent Testing:** Agile software development includes early and frequent testing, which can help to catch issues and bugs early in the development process.

Disadvantages of Agile Software Development

- **Lack of Predictability:** Agile software development can be less predictable than traditional methods, with less certainty about the final product and its delivery schedule.
- **Limited Documentation:** Agile software development often relies less on documentation, which can make it difficult to track changes and understand the system architecture.
- **Time and Resource Constraints:** Agile software development requires a significant commitment of time and resources from all team members.

Difference Between Traditional and Agile Software Development

Traditional Software Development	**Agile Software Development**
It is used to develop simple software.	It is used to develop complicated software.
In this methodology, testing is done once the development phase is completed.	In this methodology, testing and development processes are performed concurrently.
It follows a linear organizational **expectation** structure.	It follows an iterative organizational structure
It provides less security.	It provides high security.
Client involvement is less as compared to Agile development.	Client involvement is high as compared to traditional software development.
It provides less functionality in the software.	It provides all the functionality needed by the users.

IT24: Software Project Management

Chapter 1: Introduction to Agile Project Management

Traditional Software Development	**Agile Software Development**
It supports a fixed development model.	It supports a changeable development model.
It is used by freshers.	It is used by professionals.
Examples • Office productivity suites • Data management software • Media players • Security programs	Examples • Sky • Phillips • JP Morgan Chase
Models based on Traditional Software Development- • Spiral Model • Waterfall Model • V Model • Incremental Model	Models based on Agile Software Development- • Scrum • Extreme Programming (XP) • Crystal • Dynamic Systems Development Method (DSDM) • Feature Driven Development (FDD) • Adaptive Software Development (ASD)

What Is Agile?

The agile approach in project management strongly emphasizes collaboration, communication, timekeeping, and the ability to adapt quickly to changing situations. Agile project management strongly emphasizes adaptability, cooperation, and client engagement. Customer feedback is properly considered, and regular updates are made available.

Agile frameworks like Kanban and Scrum are extensively adopted. They promote decision-making and avoid wasting time on aspects that are subject to change. Adaptive planning is undoubtedly the best characteristic of agile, which is why project managers worldwide love it.

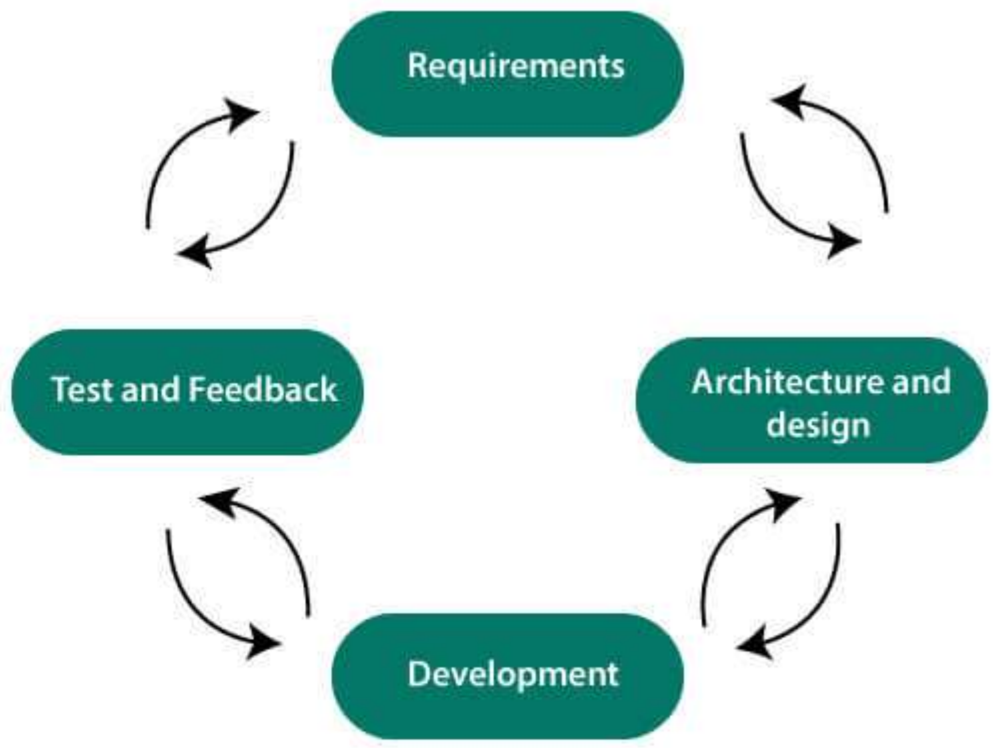
Agile Software Development Life Cycle (SDLC)

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 time and cost estimates.

IT24: Software Project Management

Chapter 1: Introduction to Agile Project Management

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 the rapid delivery of working software products. Agile SDLC breaks down the product into small incremental builds. These builds are provided in iterations.



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.

IT24: Software Project Management

<u>Chapter 1: Introduction to Agile Project Management</u>

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.

What is the Agile Manifesto?

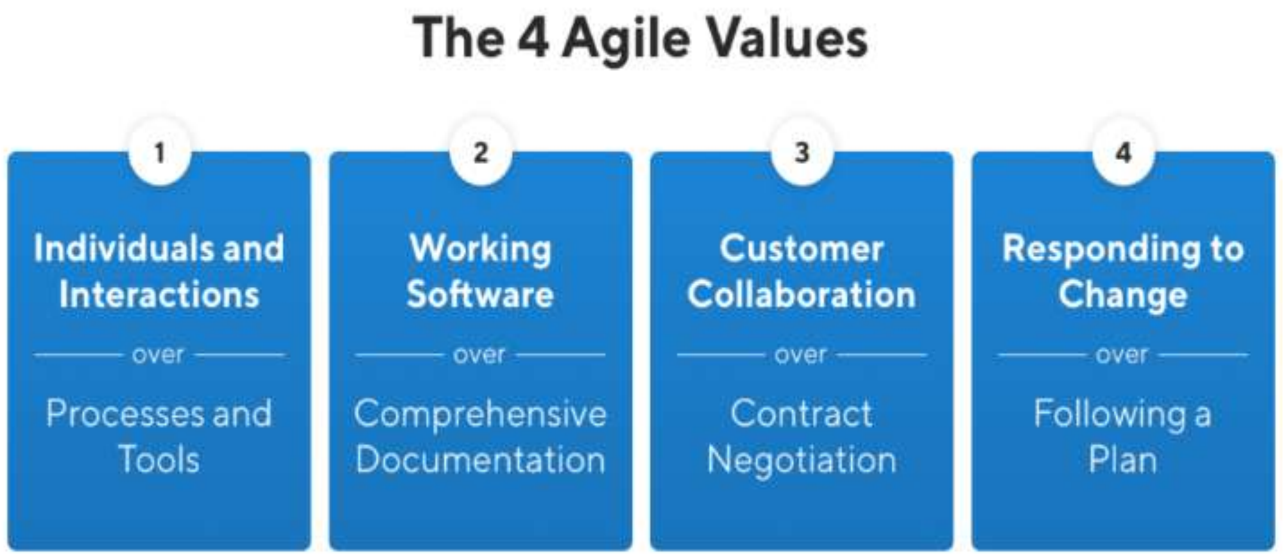
- The Agile Manifesto is a brief document built on 4 values and 12 principles for agile software development.
- The Agile Manifesto was published in February 2001 and is the work of 17 software development practitioners who observed the increasing need for an alternative to documentation-driven and heavyweight software development processes.

The 4 Agile Values

The agile mentality has 4 overarching values that differentiate it from traditional software development processes

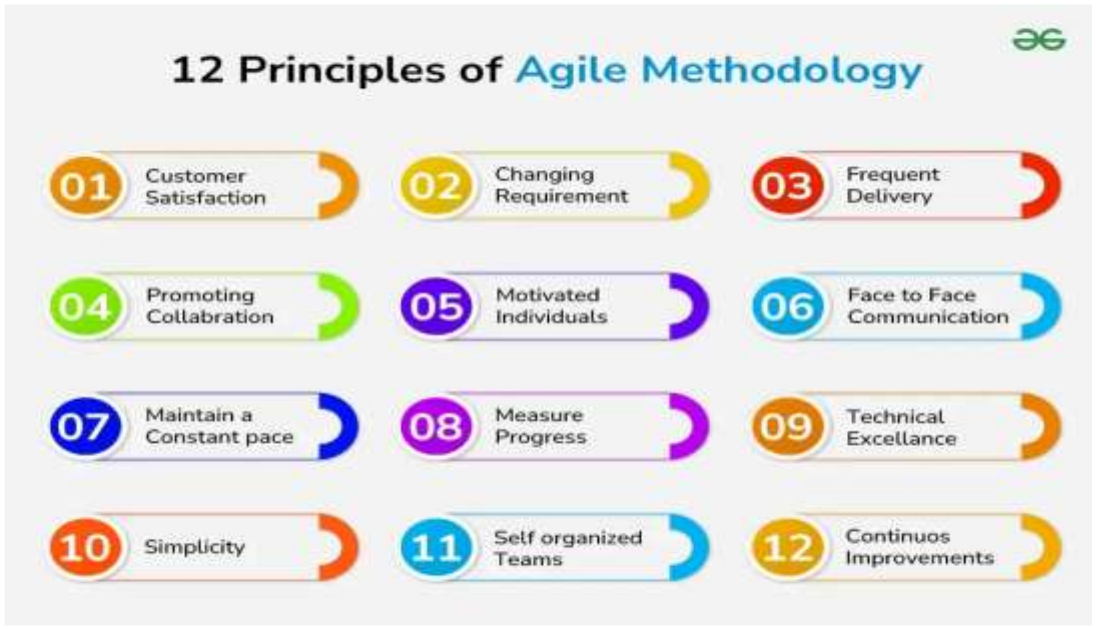
IT24: Software Project Management

Chapter 1: Introduction to Agile Project Management



Agile Principles

There are 12 agile principles mentioned in the [Agile Manifesto](). Agile principles are guidelines for flexible and efficient software development. They emphasize frequent delivery, embracing change, collaboration, and continuous improvement. The focus is on delivering value, maintaining a sustainable work pace, and ensuring technical excellence.



The Agile Alliance defines twelve lightness principles for those who need to attain agility:

IT24: Software Project Management

Chapter 1: Introduction to Agile Project Management

1. **Our highest priority is to satisfy the client through early and continuous delivery of valuable computer software**. The best ways to ensure you make customers happy while continuously delivering valuable software are to ship early, iterate frequently, and listen to your market continually.
2. **Welcome dynamic necessities, even late in development. Agile processes harness modification for the customer's competitive advantage.** In the world around us, change is the only constant. Agile principles and values support responding to these changes rather than moving forward in spite of them
3. **Deliver operating computer software often, from a pair of weeks to a couple of months, with a preference to the shorter timescale**. Agile philosophy favors breaking a product's development into smaller components and "shipping" those components frequently. Using an agile approach, therefore—and building in more frequent mini-releases of your product—can speed the product's overall development
4. **Business individuals and developers should work along daily throughout the project.**

   Communication is a critical component of any project or team's success, and agile principles essentially mandate that it's a daily event.

   It takes a village to raise a child they say, and that applies to product as well.

   Regular communication between business people and developers helps improve alignment across the organization by building trust and transparency.
5. **The build comes around actuated people. offer them the setting and support they have, and trust them to urge the task done.**

   A key part of the agile philosophy is empowering individuals and teams through trust and autonomy. The agile team needs to be carefully built to include the right people and skill sets to get the job done, and responsibilities need to be clearly defined before the beginning of a project.
6. **The foremost economical and effective methodology of conveyancing info to and among a development team is face-to-face speech**.

   With so many distributed or remote development teams these days, this principle gets a bit of critique. But at the root of it, effective communication with developers means getting these conversations out of Slack and email and favoring more human interaction (even if done by video conference calls).

   The overall objective behind this principle is to encourage product people and developers to truly communicate in real time about the product, requirements, and the high-level strategy driving those things.
7. **Working with computer software is the primary life of progress**.

   Proponents of the agile philosophy are quick to remind us that we're in the business of building software, and that's where our time should be spent. Perfect, detailed documentation is secondary to working software.

This mentality pushes to get products to the market quickly rather than let documentation or an "it's not done until it's perfect" mentality become a bottleneck. The ultimate measure for success is a working product that customers love.

8. **Agile processes promote property development. The sponsors, developers, and users will be able to maintain a relentless pace indefinitely.**

   Keeping up with a demanding, rapid-release schedule can be taxing on a team. Especially if expectations are set too high. Agile principles encourage us to be mindful of this and set realistic, clear expectations.

   The idea is to keep morale high and improve work-life balance to prevent burnout and turnover among members of cross-functional teams.

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

   While the agile philosophy encourages shorter cycles and frequent releases, it also puts emphasis on the importance of keeping things neat and tidy so they don't cause problems in the future.

   Product managers often forget about this aspect of development because they mostly don't spend their days wading through their products' codebases, but it is still of the utmost importance to them

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

    You've probably heard of the 80/20 rule—the concept that you can usually get 80% of your intended results with just 20% of the work. Agile principles encourage thinking this way; doing the things that can have the most impact

11. **The most effective architectures, necessities, and styles emerge from self–organizing groups.**

    In traditional software development methodologies, you'll often see pyramid-shaped teams where management makes key decisions for contributors. Agile principles suggest the use of self-organizing teams that work with a more "flat" management style where decisions are made as a group rather than by a singular manager or management team.

12. **At regular intervals, the team reflects on a way to become simpler, then tunes and adjusts its behavior consequently**.

    If you're truly living by agile principles, there is no place for "we can't change because we've always done it this way." Just like we're always learning new things about our customers and markets, we're also learning from the processes we're using to learn those things.

    Agile is not about following a strictly defined process for every sprint and release, it's about continuous improvement. And that continuous improvement must also extend to processes and teams.

Chapter 1: Introduction to Agile Project Management

Agile Team

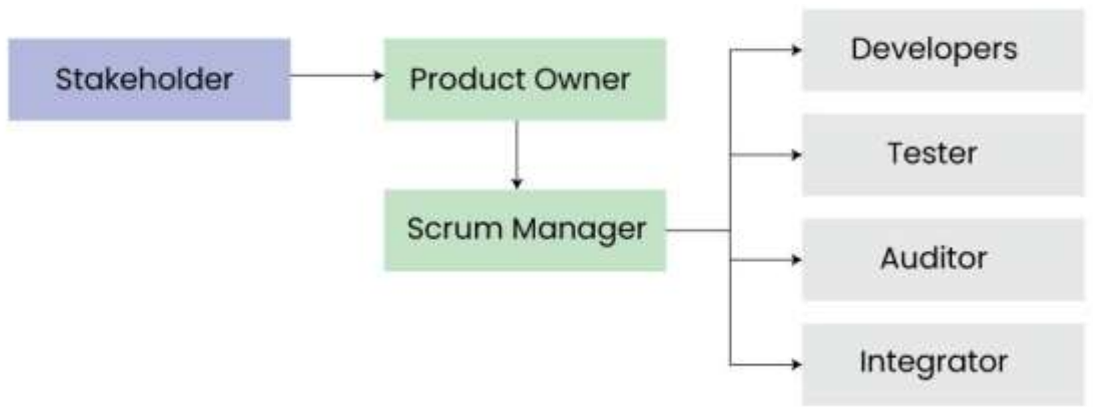
An agile team is a small, cross-functional group of people dedicated to collaboratively executing an agile project. The team is self-organizing and shares accountability for meeting customer requirements through continuous delivery of working products. All the team members contribute diverse expertise across technical, business, and interpersonal domains, taking on varied roles as needed to complete project goals. The leadership in the agile team is distributed, with the members empowered to make decisions by consensus.

What is an Agile Team?

A collection of people arranged to collaborate effectively and deliver useful goods or services in a flexible and versatile way is called an agile team. Although, software development gave rise to the idea of agile, it has subsequently been used in a variety of fields and project kinds.

- In order to react quickly to evolving needs, the Agile methodology places a strong emphasis on iterative development, flexibility and customer input.
- **Extreme Programming (XP)**, **Kanban** and **Scrum** are popular frameworks for putting agile techniques into practice. To achieve the concepts and ideals of agility, agile teams may stick to particular guidelines and practices provided by these frameworks.

Roles and Responsibilities of an Agile Team



Roles in Agile Team

IT24: Software Project Management

Chapter 1: Introduction to Agile Project Management

- **Product Owner**: The product owner is responsible for representing the stakeholders and the end customers. They define the product vision and roadmap, prioritize the product backlog, and accept or reject the resulting work or outcome. They also manage stakeholders expectation, communicate goals, and ensures the team is delivering maximum business value.
- **Scrum Master:** The scrum master is responsible for managing the agile processes and removing obstacles that prevent the development team from being productive. They facilitate meetings, promote collaboration, and encourage the team to improve its practices. In simple terms, they act as a binding force towards reaching conclusion of the project.
- **Development Members/Developers:** The developers are a cross-functional group that carries out the actual work to build the product incrementally. They collaborate daily in short, high-paced meetings to analyse, design, develop, test, and implement the user stories from the product backlog. The development team are usually structured yet flexible and self organized while carrying their task execution.
- **Stakeholders**: Stakeholders are individuals or groups who are impacted by the project. They may be within or external to the organization. They provide feedback on the requirements while reviewing progress of the project and also evaluate the final product. The stakeholders are usually engaged and kept in loop to allow the project team to meet their required business needs.
- **Integrator:** The integrator is responsible for technical integration of work by the development team. They ensure that the software/hardware components fit together properly. The integrator also makes sure the system meets quality standards and integrates smoothly with external systems.
- **Independent Auditor and Tester:** The independent tester objectively evaluates the system to ensure the quality of the product. They audit the product for bugs and flaws from an unbiased perspective. The tester identifies defects and works with the team to get them fixed.