Nagarjuna College of Engineering & Technology

UML AND AGILE PRACTICES -18EET751

Assignment Questions

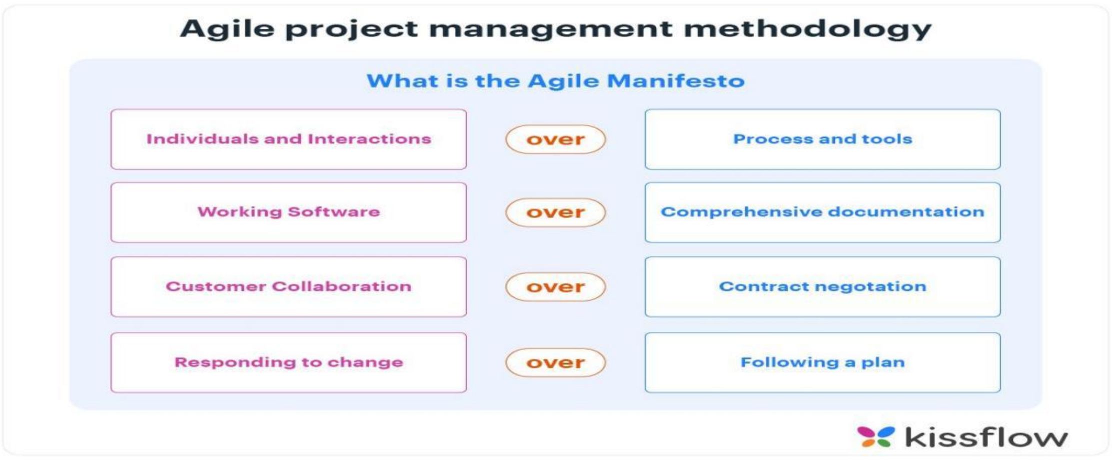
Module 3 & Module 4

1. What is agile development? Briefly explain.

Agile is a time-bound, iterative approach to software delivery that builds software incrementally from the start of the project, instead of trying to deliver all at once.

Agile practices include requirements discovery and solutions improvement through the collaborative effort of self-organizing and cross-functional teams with their customer(s)/end user(s), adaptive planning, evolutionary development, early delivery, continual improvement, and flexible responses to changes in requirements, capacity, and understanding of the problems to be solved.

2. write a note on : (i) agile practices (ii) agile values (iii)agile principles .


Agile project management methodology

What is the Agile Manifesto

Individuals and Interactions	over	Process and tools
Working Software	over	Comprehensive documentation
Customer Collaboration	over	Contract negotiation
Responding to change	over	Following a plan

- **Individuals and interactions** over processes and tools
- **Working software** over comprehensive documentation
- **Customer collaboration** over contract negotiation
- **Responding to change** over following a plan

1. Individuals and interactions over processes and tools: This value of the Agile manifesto focuses on giving importance to communication with the clients. There are several things a client may want to ask and it is the responsibility of the team members to ensure that all questions and suggestions of the clients are promptly dealt with.

2. Working product over comprehensive documentation: In the past, more focus used to be on proper documentation of every aspect of the project. There were several times when this was done at the expense of the final product. The Agile values dictate that the first and foremost duty of the project team is completing the final deliverables as identified by the customers.

3. Customer collaboration over contract negotiation: Agile principles require customers to be involved in all phases of the project. The Waterfall approach or Traditional methodologies only allow customers to negotiate before and after the project. This used to result in wastage of both time and resources. If the customers are kept in the loop during the development process, team members can ensure that the final product meets all the requirements of the client.

4. Responding to change over following a plan: Contrary to the management methodologies of the past, Agile values are against using elaborate plans before the start of the project and continue sticking to them no matter what. Circumstances change and sometimes customers demand extra features in the final product that may change the project scope. In these cases, project managers and their teams must adapt quickly in order to deliver a quality product and ensure 100% customer satisfaction.

Agile Principles

- Agile's highest priority is to satisfy the customer through early and continuous delivery of valuable software.
- Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.
- Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
- Business people and developers must work together daily throughout the project.
- Build projects around motivated individuals. Give them the environment and support they need, and trust them to get the job done.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Working software is the primary measure of progress.
- Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity–the art of maximizing the amount of work not done–is essential.
- The best architectures, requirements, and designs emerge from self-organizing teams.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

3. **What projects can benefit from agile practices ? Explain.**

4. **Explain the advantages of using agile methodologies .**

1. Superior Quality Product: In Agile project management, testing is an integrated part of the project execution phase which means that the overall quality of the final product is greater. The client remains involved in the development process and can ask for changes depending on the market realities. Since Agile is an iterative process, self-organizing teams keep on learning and growing with time and continue improving.

2. Customer satisfaction: In the Agile, the customer is always involved in the decision-making process which leads to greater customer retention. In the traditional framework, the customer is only involved in the planning phase and does not influence execution which affects the flexibility and adaptability. By keeping the customer in the loop and making changes according to their feedback, you deliver value to the customer and ensure that the final product is truly according to their requirements.

Another benefit of Agile Project Management is that the go-to-market time gets significantly reduced. This allows the product owner to successfully capitalize on the opportunity and in some cases, enjoy the first-mover advantage. It's only natural that when customers get to enjoy these benefits because of your performance, they'll come back to you for other projects.

3. Better control: Agile allows managers to have better control over the project due to its transparency, feedback integration, and quality-control features. Quality is ensured throughout the implementation phase of the project and all stakeholders are involved in the process with daily progress reports through advanced reporting tools and techniques.

4. Improved project predictability: With increased visibility, predicting risks, and coming up with effective mitigation plans becomes easier. Within the Agile framework, there are greater ways to identify and predict risks and plan to ensure that the project runs smoothly.

Scrum methodology, for example, uses sprint backlogs and burndown charts to increase the visibility of the project which allows managers to predict performances and plan accordingly.

5. Reduced risks: In theory, any project using an Agile methodology will never fail. Agile works in small sprints that focus on continuous delivery. There is always a small part that can be salvaged and used in the future even if a particular approach doesn't go as planned.

6. .Increased flexibility: When Agile is truly implemented in a project team, it empowers them with unparalleled flexibility. Teams work in smaller bursts and are supplemented by the constant feedback and involvement of the product owner. In other project management methodologies, changes usually are time-consuming and costly.

However, Agile divides the project in short sprints that are both manageable and flexible enough to allow the team to implement changes on short notice. This unmatched flexibility is one of the top reasons why dynamic organizations prefer to use Agile in their project.

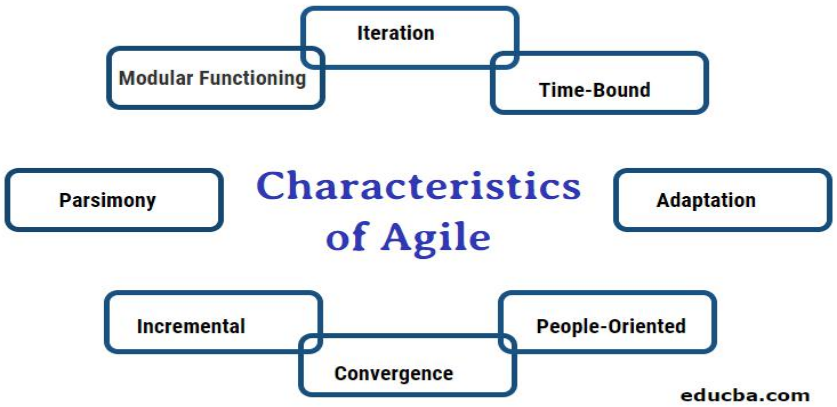
7. Continuous Improvement: Working on self-reflection and striving for continuous improvement is one of the 12 core principles of the Agile manifesto. The methodology works in iterations which means that each sprint will be better than the last one and previous mistakes will not be repeated. Agile methodologies foster an open culture of idea exchange and collaboration which allows team members to learn from shared experiences and improve together.

8. Improved team morale: As Agile teams are self-organized and self-managing, they have increased autonomy and authority over their decisions. The project manager shields the team from interference from sponsors and management.The cross-functional nature of the teams also helps the members learn new project management skills and grow in their current roles. The team gets together frequently to discuss challenges and statuses letting them collaborate better. Since the team size is limited, Agile provides an environment where teams are close-knit and can have flexible team structures.

9. More relevant metrics: The metrics used by Agile teams in estimating time and cost, measuring project performance are more accurate and relevant than the ones used in traditional methodologies. Agile emphasizes on producing results and optimizing performance while the metrics in Waterfall methodology show how closely the project is tracking against the estimated cost and time.Agile produces important metrics like lead time, cycle time, and throughput that helps measure the team's performance, identify bottlenecks and make data-driven decisions to correct them.The Agile framework is a powerful tool that helps managers, team members, and clients. From improving the quality of the product to helping in the professional development of the team members, the benefits of Agile are numerous. It helps teams overcome pitfalls like excessive costs and scope creep

5. Write a note on : (i) characteristics of agile (ii) Project management in agile .

Characteristics of agile:



1. Modular Functioning: Modularity is considered one of the key elements of a good process. Modularity is the element that allows the components to break down and that broken component is called activities. The software development process is just the set of activities that frames or transforms the vision of the software system into reality. Agile Software development process makes use of good tools and is wielded with good software craftsman who is well known to apply those at the right place and right time. These can not be utilized for the production line for manufacturing software products.

2. Iteration: The agile software development process acknowledges the working on attempting wrong before its correct. So, for this reason, agile processes focus on small cycles. Each cycle has a task of defined activities and those activities must be completed in a correct manner, these cycles have a time slot of a week, from starting to completing the activities. The iteration i.e single cycle may or may not get a 100 % correct element. Because of this reason one short cycle is repeated several times until the correct result is achieved.

3. Time-Bound: Software development comes with time limits or the development team must give a delivery date to the customer, to keep things under track the iterations play a good role as it keeps time limit between one and six weeks on each iteration and it can be scheduled accordingly. There are higher chances that it may not schedule all activities in a single iteration, else wise only those activities will be attempted which are necessary to achieve the goals which were set at the beginning of the iteration. Rescheduling or functionality reduction can be done to deliver the project on time, on the allotted time.

4. Parsimony: Agile software development is considered an upgraded version of the traditional approach with time constrains add on. Impossible deadlines are not attempted for rapid delivery, each phase of development is kept in mind as this attempt may take away the quality from the product and that's a big NO. Instead, agile approach focus on parsimony keeps the activities to minimal and only necessary to mitigate risks and achieve their goal.

5. Adaptation: During the development or during iterations there are higher chances of unknown risks they may be exposed; the agile approach is prepared to deal with these unknown risks. If there are changes in different results during the functionality, new activities or functionality can be added to reach the goal.

6. Incremental: Agile system is not built entirely at once, the system is partitioned and look out for increments that can be parallelly developed, at a different time and a different rate. Each increment is tested independently and if found ok then all are integrated into the one system for the result.

7. Convergence: It means that the risks are attacked actively because it is worth to know the risks. This takes the system closer to the results. Risks solving during each iteration is one of the great processes that leads to a successful iteration.

8. People-Oriented: The agile process is known for its priority towards customers over process and technology. The involvement of the customer is done organically. The developers evolve through adaptation and are empowered to raise their productivity and performance. These developers are very aware of dealing with the changes in the system at every stage.

9. Collaboration: The agile process has a very practical approach for discussions that is face-to-face discussion whether it is with the customer or with the team members itself. Good communications play an important role in the success of the project in the software development field. The risk of miscommunication is higher when the system is developed into pieces, it is a must for every member to understand how pieces fit together for creating a final product. The process is more into integration than to communication when individual iterations are completed. Integrating the smaller integrations into larger part developed parallelly requires collaboration with the teams to fix it correctly into the system to get the final product.

Project Mangement in Agile:

1. Project Planning: Like with any project, before beginning your team should understand the end goal, the value to the organization or client, and how it will be achieved.You can develop a project scope here, but remember that the purpose of using Agile project management is to be able to address changes and additions to the project easily, so the project scope shouldn't be seen as unchangeable.

2. **Product Roadmap creation:** A roadmap is a breakdown of the features that will make up the final product. This is a crucial component of the planning stage of Agile, because your team will build these individual features during each sprint. At this point, you will also develop a product backlog, which is a list of all the features and deliverables that will make up the final product. When you plan sprints later on, your team will pull tasks from this backlog.

3. **Release planning:** In traditional waterfall project management, there is one implementation date that comes after an entire project has been developed. When using Agile, however, your project uses shorter development cycles (called sprints) with features released at the end of each cycle. Before kicking off the project, you'll make a high-level plan for feature releases and at the beginning of each sprint, you'll revisit and reassess the release plan for that feature.

4. **Sprint Planning:** Before each sprint begins, the stakeholders need to hold a sprint planning meeting to determine what will be accomplished by each person during that sprint, how it will be achieved, and assess the task load. It's important to share the load evenly among team members so they can accomplish their assigned tasks during the sprint.You'll also need to visually document your workflow for team transparency, shared understanding within the team, and identifying and removing bottlenecks.

5. **Daily Standups:** To help your team accomplish their tasks during each sprint and assess whether any changes need to be made, hold short daily stand-up meetings. During these meetings, each team member will briefly talk about what they accomplished the day before and

what they will be working on that day.These daily meetings should be only 15 minutes long. They aren't meant to be extended problem-solving sessions or a chance to talk about general news items. Some teams will even hold these meetings standing up to keep it brief.

6. **Sprint Review and retrospective:** After the end of each sprint, your team will hold two meetings: first, you will hold a sprint review with the project stakeholders to show them the finished product. This is an important part of keeping open communication with stakeholders. An in-person or video conference meeting allows both groups to build a relationship and discuss product issues that arise.

Second, you will have a sprint retrospective meeting with your stakeholders to discuss what went well during the sprint, what could have been better, whether the task load was too heavy or too light for each member, and what was accomplished during the sprint.

If your team is new to Agile project management, don't skip this essential meeting. It helps you gauge how much your team can tackle during each sprint and the most efficient sprint length for future projects.

6. Explain Lean software development in agile.

Lean Software Development (LSD) is an agile framework that is used to streamline & optimize the software development process. It may also be referred to as Minimum Viable Product (MVP) strategy as these ways of thinking are very much alike since both intend to speed up development by focusing on new deliverables.

Toyota has been credited to inspire the lean development approach which is meant for optimizing production and minimize waste. Seeing Toyota's lean approach many other manufacturing teams started to follow the same strategy. And it was first adopted in software development in 2003.

Advantages of LSD :

LSD has proved to improve software development in the following ways :

1. LSD removes the unnecessary process stages when designing software so that it acts as a time saver as simplifies the development process.
2. With a focus on MVP, Lean Software Development prioritizes essential functions so this removes the risk of spending time on valueless builds.
3. It increases the involvement power of your team as more and more members participate due to which the overall workflow becomes optimized and losses get reduced.

Key Principles of Lean Software Development :

There are 7 established lean principles that come with a set of tactics, practices, and processes that builds more efficient software products :

1. Eliminating the waste
2. Fast Delivery
3. Amplify Learning
4. Builds Quality
5. Respect Teamwork
6. Delay the commitment
7. Optimizing the whole system

The below figure illustrates the principles of LSD :

Eliminating the Waste
Fast Delivery
Amplify Learning
Builds Quality
Respect Teamwork
Delay the Commitment
Optimizing the whole Systm

LSD PRINCIPLE

Eliminating the Waste: To identify and eliminate wastes e.g. unnecessary code, delay in processes, inefficient communication, the issue with quality, data duplication, more tasks in the log than completed, etc. regular meetings are held by Project Managers. Which allows team members to point out faults and suggest changes in the next turn.

Fast Delivery: Previously long time planning used to be the key success in business, but in the passage of time it is found that engineers spend too much time on building complex systems with unwanted features. So they came up with an MVP strategy which resulted in the building products quickly that included a little functionality and launch the product to market and see the reaction. Such an approach allows them to enhance the product on the basis of customer feedback.

Amplify Learning: Learning is improved through ample code reviewing, meeting that is cross-team applicable. It is also ensured that particular knowledge isn't accumulated by one engineer who's writing a particular piece of code so paired programming is used.

Builds Quality: LSD is all about prevent waste, keeping an eye on not sacrificing quality. Developers often apply test-driven programming to examine the code before it is written. The quality can also be gained to get constant feedback from team members and project managers.

Respect Teamwork: LSD focuses on empowering team members, rather than controlling them. Setting up a collaborative atmosphere, keep perfect balance when there are short deadlines and immense workload. This method becomes very much important when new members join a well-established team.

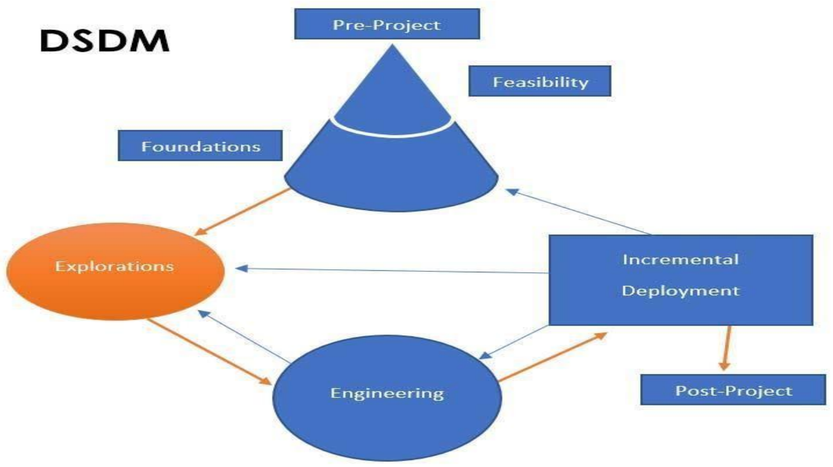
Delay the Commitment: In traditional project management it often happens when you make your application and it turns out to be completely unfit for the market. LSD method recognizes this threat and makes room for improvement by postponing irreversible decisions until all experiment is done. This methodology always constructs software as flexible, so the new knowledge is available and engineers can make improvements.

Optimizing the whole system: lean's principle allows managers to break an issue into small constituent parts to optimize the team's workflow, create unity among members, and inspire a sense of shared responsibility which results in enhancing the team performance.

Weakness in LSD :

1. Make it scalable as other frameworks since it strongly depends on the team involved.
2. It is hard to keep pace so it is not easy for developers to work with team members as conflict may occur between them.
3. It leads to a difficult decision-making process as it is mandatory for customers to clearly set their requirements for the development not to be interrupted.

7. **Explain Dynamic system development method in agile.**



Dynamic Software Development Method (DSDM) was developed in the year 1994 by a group of vendors and experts in the field of Software development. DSDM focuses on Software projects that are characterized by tight budgets and schedules. It focuses on frequent delivery of product cycles, and development is iterative and incremental.

With Dynamic Software Development Method (DSDM), one can design a roadmap of early and continuous deliveries for the project, implementing an incremental solution, adapting from the feedback obtained throughout the process, and checking that the expected benefits are being met.

DSDM is an agile model that can undoubtedly help organizations that are used to working on projects to change their mentality and way of working to improve their capacity to deliver value and reduce time to market.

8. **Write a note on : (i) Crystal methodologies in agile (ii) adaptive software development in agile .**

i) Crystal methodologies in agile:

Crystal method is a agile framework that is considered as a lightweight or agile methodologies which focuses on individuals and the interactions. The methods are color-coded to signify risk to human life. It is mainly for short-term projects by a team of developers working out of a single workspace. Among few Agile Software Development Life Cycle (SDLC) models crystal is considered as one of the Agile SDLC model.

Two core belief of Crystal method :

- Find 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 Crystal Method :

Crystal method was developed by an American scientist named Alistair Cockburn who worked in 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 doesn't any fixed tools but can be but can be changed anytime to meet 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, 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 a 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 team knows exactly what to do, which enables them to focus their attention. This boosts team interaction and work towards the same goal.
6. **Easy access to expert users:** It enhances the team communication with users and get regular feedback from real users.
7. **Technical tooling-**It contains very specific technical tools which to be used by software development team during testing, management and configuration. These tools make it enable for the team to identify any error within less time.

Crystal family consists of many variants like Crystal Clear, Crystal Yellow, Crystal Red, Crystal Sapphire, Crystal Red, Crystal Orange Web, 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 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 use of too much documentation.

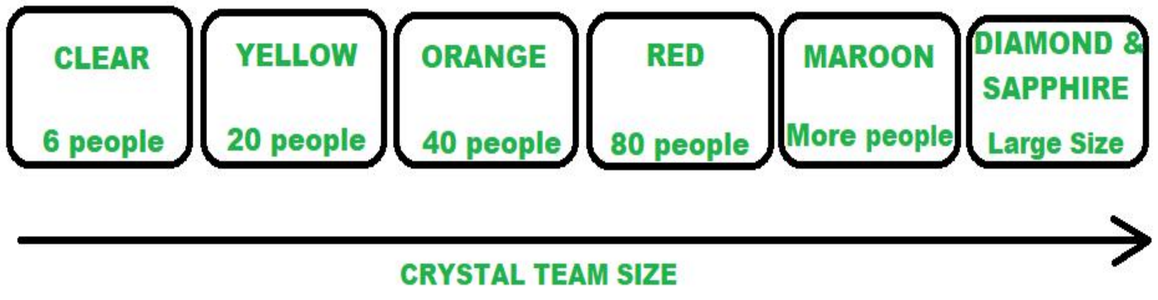
3. **Crystal Orange:** It has a team size of 21-40 members, where team is split according to their functional skills. Here the project generally lasts for 1-2 years and the release is required every 3 to 4 months.

4. **Crystal Orange Web:** It has also the team size of 21-40 members where 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 single project but 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 team size is of 80-200 members 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.



Benefits of using the Crystal Agile Framework :

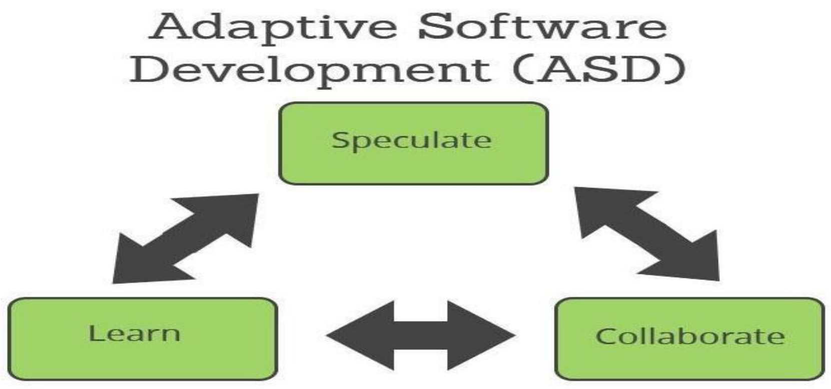
- Facilitate and enhance team communication and accountability.
- The adaptive approach lets the team respond well to the demanding requirements.
- Allows team to work with one they see the most effective.
- Teams talk directly with each other, that reduce management overhead.

Drawbacks of using the Crystal Agile Framework :

- Lack of pre-defined plans may lead to confusion and loss of focus.
- Lack of structure may slow down inexperienced teams.
- Not clear on how a remote team can share knowledge informally.

The Crystal Method is expandable. It may be used by small teams or large teams to work on simple or complex objects. It places importance on developmental skills, interactions which in turn encourages exchange of ideas. It is also beneficial for the clients as it delivers most important components of the product first. But on the other hand, the Crystal Method does not plan based on the requirements of the projects.

(ii) adaptive software development in agile .



Adaptive Software Development (ASD) was developed by Jim Highsmith and Sam Bayer in the early 1990s. It incorporates the principles of continuous adaptation, i.e., adapt to change and not fight against it. Adaptive Software Development uses a dynamic development cycle known as Speculate, Collaborate, and Learn. This cycle is dedicated to constant learning and intense collaboration between developers and customers due to the constant change in the business environment.

Unlike most Software development methodologies which use a static life cycle i.e., Plan-Design-Build, ASD offers a non-linear iterative life cycle, where each cycle can iterate and be modified while another cycle is being executed. It points towards Rapid Application Development (RAD), which emphasizes development speed to create a high quality, low maintenance product involving the user as much as possible. The main characteristics of ASD are:

1. Speculate: This is the initiation phase of the project where it is necessary to establish the main objectives and goals of the project by understanding the limitations (risk areas) with which the project operates.
2. Collaborate: This is the phase where most of the development is centered, maintaining co-ordination between teams that ensures what is learned by one team is communicated to the rest and does not have to be learned again by other teams from scratch.
3. Learn: The last stage ends with a series of collaboration cycles – the job is to capture what has been learned, both positive and negative. This stage is critical for the effectiveness of the project.

8. Differentiate between agile project development approach with traditional approach.

Characteristics	**Agile approach**	**Traditional approach**
Organizational structure	Iterative	Linear
Scale of projects	Small and medium scale	Large-scale
User requirements	Interactive input	Clearly defined before implementation
Involvement of clients	High	Low
Development model	Evolutionary delivery	Life cycle
Customer involvement	Customers are involved from the time work is being performed	Customers get involved early in the project but not once the execution has started
Escalation management	When problems occur, the entire team works together to resolve it	Escalation to managers when problem arise
Model preference	Agile model favors adaption	Traditional model favors anticipation
Product or process	Less focus on formal and directive processes	More serious about processes than the product
Test documentation	Tests are planned one sprint at a time	Comprehensive test planning
Effort estimation	Scrum master facilitates and the team does the estimation	Project manager provides estimates and gets approval from PO for the entire project
Reviews and approvals	Reviews are done after each iteration	Excessive reviews and approvals by leaders

9. Explain the process of selecting right agile approach .

Development Team's Skills

Development team's skills and capability is a focused area for any agile method. However, the particular skillsets may vary based on the organizations and their chosen agile methodology. Some of the agile methodologies need high programming skills along with effective communication skill for example XP, whereas others may not.

Development team's communication skill: Agile methodology overview describes it as a collaborative approach. Hence, the development team's communication skill is very much responsible for the success of any agile method. This communication mode could be either direct or indirect depending on the project size and team locations.

Development Teams' Competency: While choosing an agile method it is important to evaluate the technical competency of the existing team as well as their real-world experience in a similar domain. Besides, that agile methodology needs continuous and faster turnaround which essentially demands the development team's competency. In this context team members, technical skill is an important parameter to choose the correct agile method as it is impossible for team members to know all the agile methods. Moreover, if an inappropriate agile method is chosen, the project will end up with an unexpected outcome.

Development Teams' Domain Knowledge: Like technical competency, domain knowledge is an important criterion for the project team for the successful implementation of the project. Hence, inappropriate selection of agile method may lead to the poor implementation of the project.

Customer Involvement:

As described above Agile Manifesto considers customer involvement more important over customer negotiation. Because they are the key sources to decide the scope, resources, any clarifications needed to expedite the project and last but not the least they possess higher domain knowledge. Hence, customers' involvement is an important parameter to consider while selecting an agile method.

Customer collaboration: One of the 12 principles of the agile manifesto as described earlier prioritizes customer satisfaction through continuous and early delivery. However, that needs customers' active participation in the project, high motivation towards project goal and their physical availability to the development team. Thus, customer collaboration is a major factor in selecting an agile method as different agile methodologies need a different level of customer collaboration.

Customer commitment: Customer commitment can affect an agile method. Uncommitted customers can severely affect some agile methods like XP. Hence, the selection of an agile method must be aligned with customer commitment.

Customer's Domain Knowledge: Proper domain knowledge is also a key factor because if the functionalities are not clarified appropriately from the customer end, it may put the project implementation at risk.

Organizational Culture

Adopting an agile method is very much under control of the culture of an organization. Because how quickly the organization can adopt the agile method that very much depends on its flexibility level. There are mainly four types of organizational culture observed in the industry which can be associated with the selection of the agile method and these are:

- Hierarchical
- Random
- Collaborative
- Synchronous

Additionally, the type of organizational culture dimensions plays a pivotal role in selecting the agile method. Different culture dimensions are:

- clan
- democratic
- hierarchical
- disciplined

Interestingly democratic culture is the most appropriate for agile methodologies.

Nature of the Project

Agile software development may be complex and may come in different team sizes and with different level of criticalities. Hence, the main three subfactors for the nature of the project i.e. size, criticality and decomposability must be appropriately evaluated before selecting the agile methodology.

Size: It is a known fact that agile methodology is most appropriate for a medium-sized project considering the testing phases. Hence, agile methodology in testing is an essential criterion because running continuous and fast unit and smoke testing for a small project is more manageable than a large one. However, if the project is a large one, then it will require other testing such as hardware and integration testing as well. So, not all agile methodology in testing is feasible or well fitted. For example, the ideal team size for some of the agile methods are as below:

Agile Method	**Project size supported**	**Ideal no of team members**
Crystal Orange method	Large	40
FDD	Very Large	>40
XP	Small	10
Scrum	Small -Medium	6+3

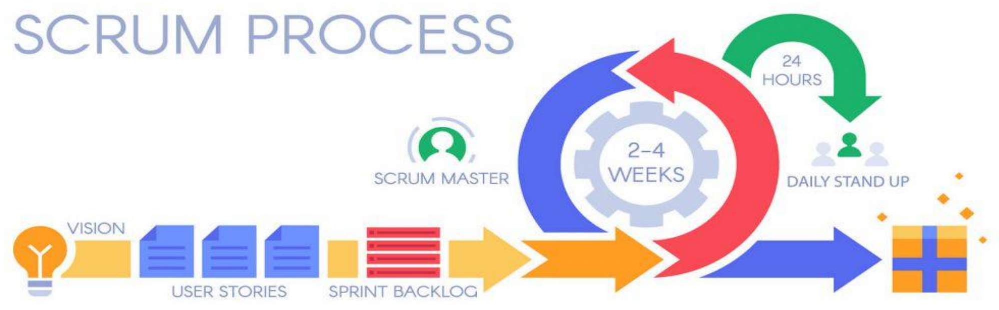
Criticality: Project criticality is a crucial factor that measures the cost of the project. Hence, if the right agile method is not selected based on the criticality, it can drastically increase the project cost. For example, FDD is the best method for critical projects.

Decomposability: Some projects may be very critical which needs to be decomposed for an in-depth analysis of the project. Hence, in such scenarios, if we select the agile method which does not support decomposability, then it will increase the overall complexity of the project with paralyzing the solution.

Project Constraints

Delivery date (or schedule), cost, scope, quality, and value are crucial considerations in any project implementation. The correct selection of the agile method with these factors determines the implementation success of any project.

10. Explain SCRUM in agile software development .



Scrum is one of the most popular frameworks for implementing agile. Its so popular, in fact, that many people think scrum and agile are the same thing. Many frameworks can be used to implement agile, such as kanban for example, but scrum has a unique flavor because of the commitment to short iterations of work.Scrum is an innovative approach to getting work done in efficient way. It is iterative & incremental agile software development method. These iterations are time boxed with various iterations & each iteration is called Sprint. The Sprint is basically 2-4 week long & each sprint requires sprint planning estimation. According to latest surveys Scrum is the most popular agile project management methodology in software development. The term Scrum is formed from Rugby.

Scrum is ideally used where highly emergent or rapidly changing requirements. Scrum is basically worked on a self-organizing, cross-functional team. In the overall scrum team there is no team leader who assign the task to team rather whole scrum members work as a team & they decides the task on which they will work on. Also the problem will be resolve by team.

Each Agile Development Scrum team having three core scrum roles: Product Owner, Scrum Master & The Team.

1) Product Owner: The Product Owner is the person who represents the stakeholders and is the voice of the customer. Product owner writes the User Stories, ordered priorities and add in the Product Backlog. It is recommended that Agile Scrum Master should not mix with Product Owner.

2) Scrum Master: The Scrum-Master is a facilitator, team leader who ensures that the team adheres to its chosen process and removes blocking issues to deliver the sprint deliverable/goal. Scrum Master is not a team leader but act as a shield for the team from external interference's & also removes barriers.

3) The Team: The scrum development team is generally size of 5-9 peoples with self-organizing and cross-functional skills who do actual work like Analysis, Design, Development, Testing, Documentation etc.

11. Explain disadvantages of agile methodology .

1. **Lack of documentation:**This is one of the biggest issues faced when teams transition from Waterfall project management to an Agile framework. Agile teams condense large volumes of data into smaller user stories, which don't contain a great amount of detail. This can make it difficult for a developer to grasp the exact customer requirements. Without a clearly documented plan or an official process to follow, team members can easily get confused when moving through project stages.

2. **Scope creep:** Another major obstacle is scope creep. Customer needs change constantly, inevitably leading to a widening of the project scope. Deliverables multiply quickly, and new features are often added to the workload. Some requirements may need to be rewritten entirely or replaced with updated ones. Teams can become overwhelmed and lose track of these requirements, unsure of which ones to prioritize.

3. **High demands on time:**Time is another consideration to add to the list of Agile challenges. Team members must make room in their schedule for daily standup meetings, which can disrupt their workflow. What's more, the Agile philosophy requires developers to engage in constant collaboration with testers, clients, and other project stakeholders. This high level of interaction can place a significant strain on Agile team members and their time management abilities.

4. **Unsuitable for long-term projects:** Finally, one of the most common Agile problems occurs when teams try to make the methodology work for unsuitable projects. Agile iterations are designed to produce smaller deliverables incrementally, which is ideal for software development. However, this level of fragmentation would not be compatible with a long-term project. For example, in a construction project such as building a house, the final deliverable is fixed and change is undesirable, making it more suited to a Waterfall framework. To limit the potential disadvantages of Agile, you should research your preferred Agile project management framework thoroughly before implementing it in your organization. If you already use an Agile framework, consider the question posed by professional services firm Deloitte: "How fragile is your Agile?" Document your existing pain points and brainstorm ways to strengthen your future Agile projects.

12. Explain agile Testing .

AGILE TESTING is a testing practice that follows the rules and principles of agile software development. Unlike the Waterfall method, Agile Testing can begin at the start of the project with continuous integration between development and testing. Agile Testing methodology is not sequential (in the sense it's executed only after coding phase) but continuous.

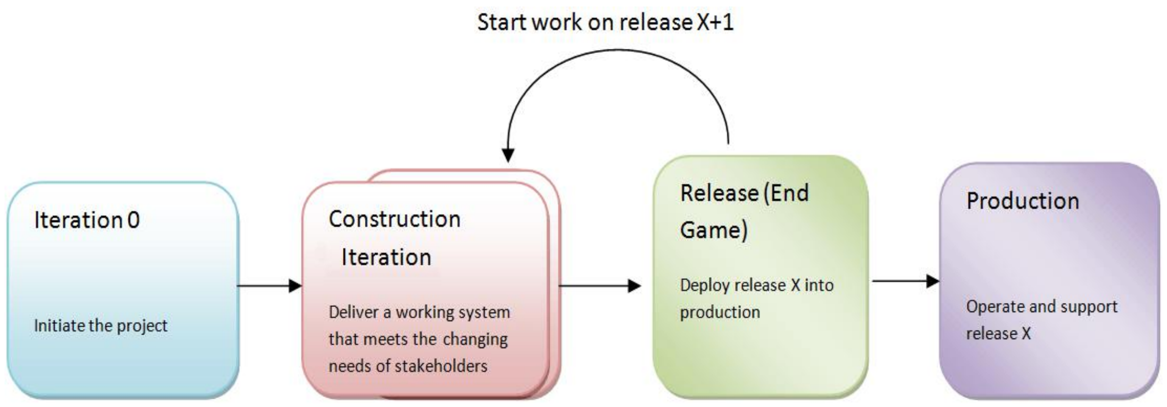
Agile test plan includes types of testing done in that iteration like test data requirements, infrastructure, test environments, and test results. Unlike the waterfall model, in an agile model, a test plan is written and updated for every release. Typical test plans in agile includes

1. Testing Scope
2. New functionalities which are being tested
3. Level or Types of testing based on the features complexity
4. Load and Performance Testing
5. Infrastructure Consideration

6. Mitigation or Risks Plan
7. Resourcing
8. Deliverables and Milestones

Agile Testing Strategies

Agile testing life cycle spans through four stages


Start work on release X+1
Iteration 0
Initiate the project
Construction Iteration
Deliver a working system that meets the changing needs of stakeholders
Release (End Game)
Deploy release X into production
Production
Operate and support release X

Agile Testing Strategy

Iteration 0

During the first stage or iteration 0, you perform initial setup tasks. It includes identifying people for testing, installing testing tools, scheduling resources (usability testing lab), etc. The following steps are set to achieve in Iteration 0

a) Establishing a business case for the project

b) Establish the boundary conditions and the project scope

c) Outline the key requirements and use cases that will drive the design trade-offs

d) Outline one or more candidate architectures

e) Identifying the risk

f) Cost estimation and prepare a preliminary project

(b) Construction Iterations

The second phase of agile testing methodology is Construction Iterations, the majority of the testing occurs during this phase. This phase is observed as a set of iterations to build an increment of the solution. In order to do that, within each iteration, **the team implements** a hybrid of practices from XP, Scrum, Agile modeling, and agile data and so on.

In construction iteration, the agile team follows the prioritized requirement practice: With each iteration, they take the most essential requirements remaining from the work item stack and implement them.

Construction iteration is classified into two, confirmatory testing and investigative testing. **Confirmatory testing concentrates** on verifying that the system fulfills the intent of the stakeholders as described to the team to date, and is performed by the team. While the investigative testing detects the problem that confirmatory team has skipped or ignored. In Investigative testing, tester determines the potential problems in the form of defect stories.

Investigative testing deals with common issues like integration testing, load/stress testing, and security testing.

Again for, confirmatory testing there are two aspects **developer testing** and **agile acceptance testing. Both of them** are automated to enable continuous regression testing throughout the lifecycle. Confirmatory testing is the agile equivalent of testing to the specification.

Agile acceptance testing is a combination of traditional functional testing and traditional acceptance testing as the development team, and stakeholders are doing it together. While developer testing is a mix of traditional unit testing and traditional service integration testing. Developer testing verifies both the application code and the database schema.

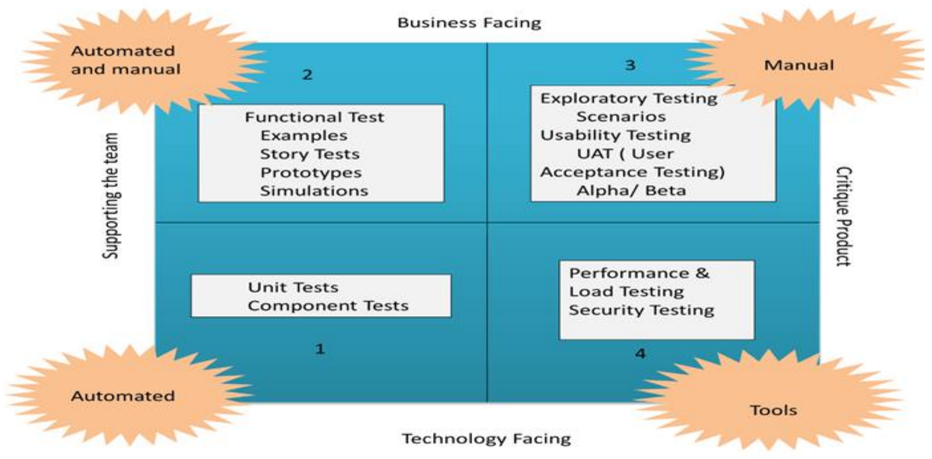
(c) Release End Game Or Transition Phase

The goal of "Release, End Game" is to deploy your system successfully into production. The activities include in this phase are training of end users, support people and operational people. Also, it includes marketing of the product release, back-up & restoration, finalization of system and user documentation.

The final agile methodology testing stage includes full system testing and acceptance testing. In accordance to finish your final testing stage without any obstacles, you should have to test the product more rigorously while it is in construction iterations. During the end game, testers will be working on its defect stories.

(d) Production

After the release stage, the product will move to the production stage.


Business Facing
Automated and manual
2
3
Manual
Functional Test Examples Story Tests Prototypes Simulations
Exploratory Testing Scenarios Usability Testing UAT (User Acceptance Testing) Alpha/ Beta
Supporting the team
Critique Product
Unit Tests Component Tests
Performance & Load Testing Security Testing
1
4
Automated
Tools
Technology Facing

The agile testing quadrants separate the whole process in four Quadrants and help to understand how agile testing is performed.

a) **Agile Quadrant I** – The internal code quality is the main focus in this quadrant, and it consists of test cases which are technology driven and are implemented to support the team, it includes

1. Unit Tests

2.Component Tests

b) **Agile Quadrant II** – It **contains** test cases that are **business driven and are implemented** to support the team. This Quadrant focuses on the requirements. The kind of test performed in this phase is

1. Testing of examples of possible scenarios and workflows

2. Testing of User experience such as prototypes

3. Pair testing

c) **Agile Quadrant III** – This quadrant provides feedback to quadrants one and two. The test cases can be used as the basis to perform automation testing. In this quadrant, many rounds of iteration reviews are carried out which builds confidence in the product. The kind of testing done in this quadrant is

1. Usability Testing

2. Exploratory Testing

3. Pair testing with customers

4. Collaborative testing

5. User acceptance testing

d) **Agile Quadrant IV** – **This quadrant concentrates** on the non-functional requirements such as performance, security, stability, etc. With the help of this quadrant, the application is made to deliver the non-functional qualities and expected value.

1. Non-functional tests such as stress and performance testing

2. Security testing with respect to **authentication** and hacking

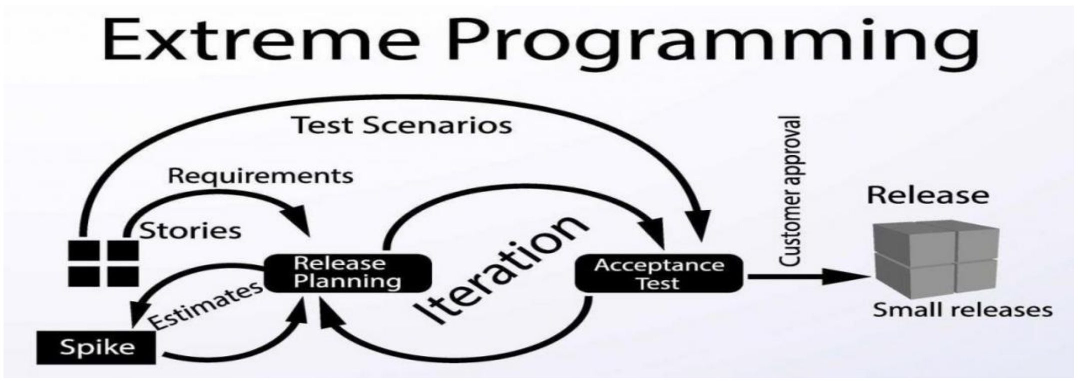
3. Infrastructure testing

4. Data migration testing

5. Scalability testing

6. Load testing

13. What is Extreme programming (XP) in agile software development . Explain principles and practices in Extreme programming .



Extreme programming is a software development methodology that's part of what's collectively known as agile methodologies. XP is built upon values, principles, and practices, and its goal is to allow small to mid-sized teams to produce high-quality software and adapt to evolving and changing requirements.What sets XP apart from the other agile methodologies is that XP emphasizes the technical aspects of software development. Extreme programming is precise about *how* engineers work since following engineering practices allows teams to deliver high-quality code at a sustainable pace.

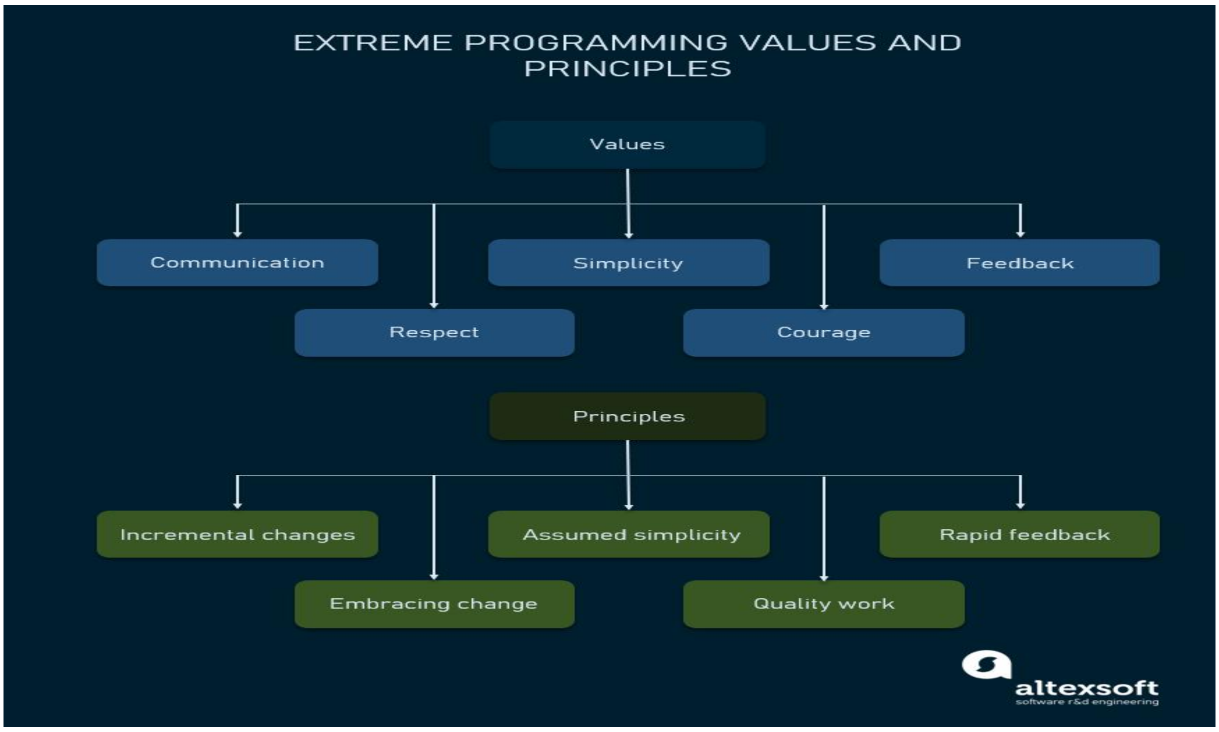
Extreme programming is, in a nutshell, about good practices taken to an extreme. Since pair-programming is good, let's do it all of the time. Since testing early is good, let's test before the production code is even written.

XP, unlike other methodologies, is very opinionated when it comes to engineering practices.

Besides practices, XP is built upon values and principles.

Values provide purpose to teams. They act as a "north star" to guide your decisions in a high-level way. However, values are abstract and too fuzzy for specific guidance. For instance: saying that you value communication can result in many different outcomes.

Practices are, in some ways, the opposite of values. They're concrete and down to earth, defining the specifics of what to do. Practices help teams hold themselves accountable to the values. For instance, the practice of Informative Workspaces favors transparent and simple communication.



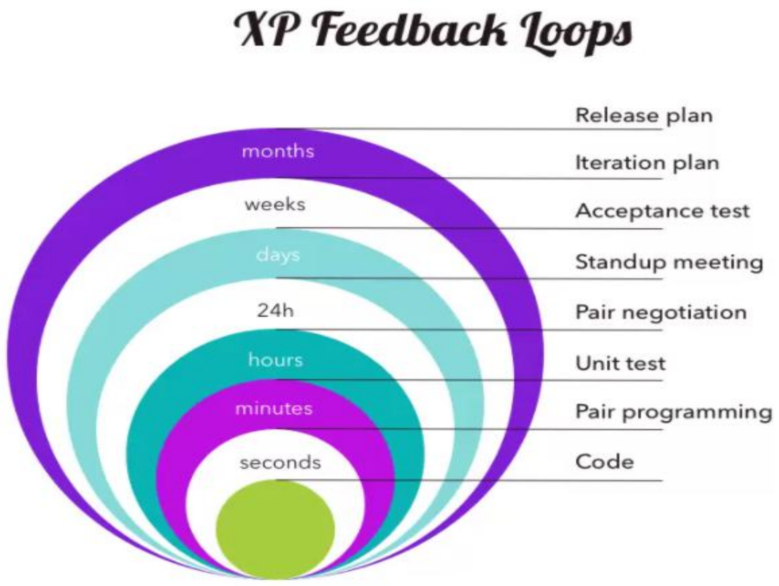
<u>Communication:</u> Lack of communication prevents knowledge from flowing inside a team. Often, when there's a problem, someone already knows how to solve it. But lack of communication prevents them from learning about the problem or contributing to its solution. So, the problem ends up being solved twice, generating waste.

<u>Simplicity:</u> Simplicity says you always strive to do the simplest thing that works. It's often misunderstood and taken as *the simplest thing*, period, ignoring the "that works" part.

It's also crucial to remember that simplicity is highly contextual. What's simple for one team, is complex for another depending entirely on each team's skills, experience, and knowledge.

<u>Feedback:</u> Feedback in more traditional, waterfall-like software development methodologies often is "too little, too late".

XP, however, embraces change and XP teams strive to receive early, constant feedback. If there is a need to course-correct, XPers want to know that as soon as possible.



Feedback comes in many shapes and sizes. When you're pair programming, the comments of your peer are vital feedback. So are the opinions from other team members about an idea, including the customer who, ideally, is a member of the team.

Tests are another source of precious feedback that go beyond the test results. Whether writing tests is easy or hard is feedback too. If you're having a hard time writing tests, your design is probably too complex. Listen to the feedback and simplify your design.

Something that sounds like a great idea might not work that well in practice. So, finished code is also a source of feedback, as is a deployed product.

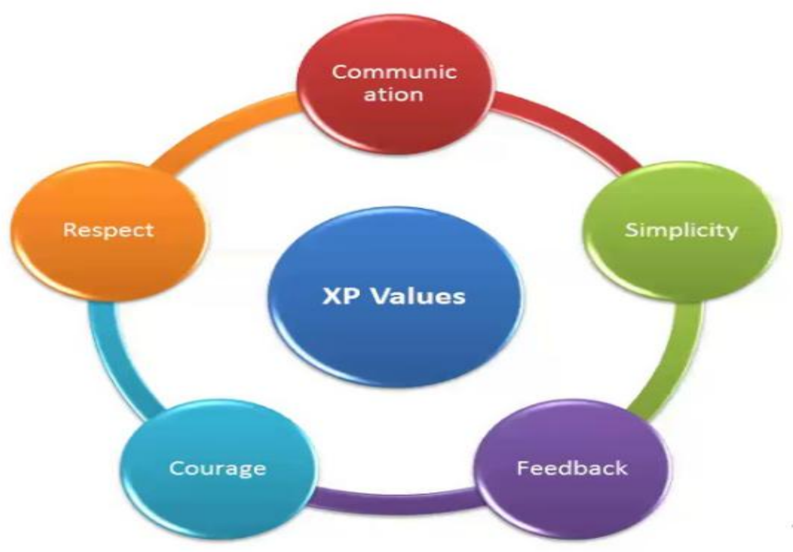
Finally, bear in mind that there's such a thing as too much feedback. If a team generates more feedback than it can handle, important feedback may drop off the radar. It's essential to then slow down and figure out what causes the feedback excess and fix it.

Courage: Kent Beck defines courage as "effective action in the face of fear." As a software engineer, you have plenty to be afraid of and therefore plenty of opportunities to show courage.

It takes courage to speak the truth, especially unpleasant ones — for instance, honest estimates. Giving and receiving feedback also takes courage. And it takes courage to avoid falling into the sunk cost fallacy and discard a failing solution that received substantial investments.

Respect: A fundamental premise of XP is that everyone cares about their work. No amount of technical excellence can save a project if there's no care and respect.

Every person is worthy of dignity and respect, and that includes, of course, the people affected by a software development project. When you and your team members respect and care about each other, the customer, the project, and its future users, everyone gains



For instance, based on the value of courage alone, you could conclude that it's advisable to tackle a big change at once in the program. However, the principle of Baby Steps tells us that big changes are risky. So, you want to favor tiny ones instead.

Humanity: Humans create software for humans, a fact that's often overlooked. But taking human basic needs, strengths, and weaknesses into account creates products humans *want* to use. And a work environment that gives you the opportunity for accomplishment and growth, the feeling of belonging, and basic safety, is a place where you more easily take others' needs into account.

Economics: In XP, teams heed the economic realities of software development all the time, they constantly assess the economic risks and needs of the project.

For instance, they'd implement user stories according to their business value rather than technical concerns.

Mutual Benefit: Following XP, you avoid solutions that benefit one party to the detriment of another. For instance, extensive specifications might help someone else understand it, but it takes you away from implementing it, and it delays it for your users.

A mutually beneficial solution is to use automated acceptance tests. You get immediate feedback on your implementation, your colleagues get precise specifications in code, and users get their features sooner. Plus, all of you get a safety net against regressions.

Self-similarity: If a given solution works at a level, it might also work at a higher or lower level. Fo**Redundancy:** The redundancy principle says that if a given problem is critical, you must employ many tactics to counter it.

Take defects. There isn't a single tactic that can prevent all defects from escaping to production.

So XP's solution is to stack a bunch of quality measures. Pair programming, tests, continuous integration. Each a single line of defense, together a virtually impenetrable wall.

Failure: Failure isn't waste when it results in knowledge. Acting and quickly learning what doesn't work is way more productive than inaction caused by indecision in choosing between many options.

Quality: Often people think there's a dilemma between quality and speed. It's the opposite: pushing for quality improvements is what makes you go faster.

For instance, refactoring — changing the structure of the code without altering its behavior — is a practice that makes the code easier to understand and change. As a result, you become less likely to introduce defects to the code, which allows you to deliver more value sooner by not having to fix bugs.

Baby Steps: Big changes are risky. XP mitigates that risk by doing changes in tiny steps, at all levels. Programmers write code in tiny steps using test-driven development. They integrate their code to the mainline several times a day, instead of only every few weeks or even months. The project itself occurs in short cycles rather than long-lasting phases.

Accepted Responsibility: In XP, responsibility should be accepted, never assigned. Responsibility should be accompanied by the authority to make decisions on what you're responsible for. The reverse also applies. You don't want people making decisions if they don't have to live with their consequences.

r instance, obtaining early and constant feedback is at play at various levels in XP.

- at the developer level, programmers receive feedback from their work using the test-first approach;

- at the team level, the continuous integration pipeline integrates, builds, and tests the code several times a day;

- at the organization level, the weekly and quarterly cycles allow teams to get feedback and improve their work as needed.

Improvement: According to the principle of improvement, teams don't strive for perfection in an initial implementation but for a good enough one, and to then learn and improve it continuously with feedback from real users.

Diversity: You and your coworkers benefit from a diversity of perspectives, skills, and attitudes. Such diversity often leads to conflict, but that's okay.

Conflict and disagreement are opportunities for better ideas to arise when everyone plays by the values of courage and respect. Courage to express opposing points of view, respect to expressing those in a civil and empathetic way. And all of this is an exercise in effective communication.

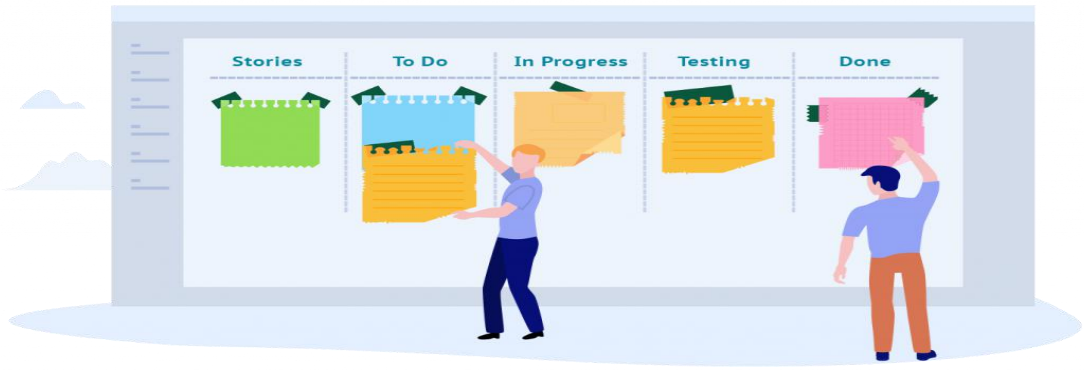
Reflection: Great teams reflect on their work and analyze how to be better. XP offers plenty of opportunities for that. Not just in its weekly and quarterly cycles, but in every practice it promotes.

Feelings are important to consider in addition to logical analysis. Your gut can inform you before you can reason about something. And so can talking to non-technical people, they can ask questions that open up entirely new possibilities.

Flow: Traditional software development methodologies have discrete phases, which last for a long time and have few feedback and course correction opportunities. Instead, software development in XP occurs in activities that happen all of the time, in a consistent "flow" of value.

<u>Opportunity:</u> Problems are inevitable in software development. However, every problem is an opportunity for improvement. Learn to look at them that way and you're far more likely to come up with creative, goal-oriented solutions that also serve to prevent them from happening again.

14. Explain Kanban in agile software development .



The Kanban Method was defined as the opposite of that – a non-disruptive evolutionary method for improvement, that ultimately enables teams to deliver continuously instead of in time-buckets of 2-3 weeks, get feedback faster and reduce the lead time to deliver value to the customer.

Kanban is a visual system for managing work as it moves through a process. Kanban visualizes both the process (the workflow) and the actual work passing through that process. The goal of Kanban is to identify potential bottlenecks in your process and fix them, so work can flow through it cost-effectively at an optimal speed or throughput.

Kanban is defined as a highly effective and efficient production system. The origin of the Kanban methodology lies in the "just-in-time" (JIT) production processes devised by Toyota, in which cards were used to identify material needs in the production chain.

15. Consider following scenario :

Adobe is working on project to come up with a competing product for Microsoft Word, that provides all the features provided by Microsoft Word and any other features requested by the marketing team. The final product needs to be ready in 10 months of time.

Compare how the project is implemented using traditional waterfall model and agile method . conclude which is best method to implement above scenario .

A Software company named **ABC** wants to make a new working project for the latest release of its operating system. The deadline for the task is 10 months. The company's head assigned two teams named **Team A** and **Team B** for this task. In order to motivate the teams, the company head says that the first team to develop the browser would be given a salary hike and a one-week full-sponsored travel plan. With the dreams of their wild travel fantasies, the two teams set out on the journey of the web browser. Team A decided to play by the book and decided to choose the Waterfall model for the development. Team B after a heavy discussion decided to take a leap of faith and choose Agile as their development model.

The Development plan of the Team A is as follows:

- Requirement analysis and Gathering – 1.5 Months
- Design of System – 2 Months
- Coding phase – 4 Months
- System Integration and Testing – 2 Months
- User Acceptance Testing – 5 Weeks

The Development plan for the Team B is as follows:

- Since this was an Agile, the project was broken up into several iterations.
- The iterations are all of the same time duration.
- At the end of each iteration, a working product with a new feature has to be delivered.
- Instead of Spending 1.5 months on requirements gathering, They will decide the core features that are required in the product and decide which of these features can be developed in the first iteration.
- Any remaining features that cannot be delivered in the first iteration will be delivered in the next subsequent iteration, based on the priority
- At the end of the first iterations, the team will deliver working software with the core basic features.

Both the team have put their best efforts to get the product to a complete stage. But then out of blue due to the rapidly changing environment, the company's head come up with an entirely new set of features and want to be implemented as quickly as possible and wanted to push out a working model in 2 days. Team A was now in a fix, they were still in their design phase and did not yet start coding and they had no working model to display. And moreover, it was practically impossible for them to implement new features since waterfall model there is not reverting back to the old phase once you proceed to the next stage, which means they would have to start from the square one again. That would incur their heavy cost and a lot of overtime. Team B was ahead of Team A in a lot of aspects, all thanks to Agile Development. They also had the working product with most of the core requirements since the first increment. And it was a piece of cake for them to add the new requirements. All they had to do is schedule these requirements for the next increment and then implement them.