UNIT 1

INTRODUCTION TO EXTREME PROGRAMMING (XP) - AGILE DEVELOPMENT

PART-A

SHORT QUESTIONS WITH SOLUTIONS

Q1. What does the term agile refers to?

Answer :

Agile refers to a general philosophy which describes groups of methods and frameworks that represent principles and values of agile manifesto. It works by dividing the project into various functional increments known as "user stories". Then, these stories are prioritized and delivered in a span of two weeks which is referred as an "iteration".

Q2. What benefits are provided by agile model?

Answer :

Agile model provides the following benefits,

1. It enables the developers to deliver a better quality product.
2. It provides excellent visibility.
3. It considered to be less risky as the developer keeps receiving feedback after every iteration and can make the required changes in the subsequent phase.
4. It is cost effective.

Q3. Define organizational success.

Answer : Model Paper-I, Q1(a)

Organizational success refers to the success which is achieved by analyzing where the organization lies in terms of its goals and mission. Organizations must execute strategies and retain employees so as to achieve increased and renewable output. Lack of organizational success might make the members of the team feel that they are not required any more in the organization.

Q4. Write any four advantages of using extreme programming.

Answer : Model Paper-II, Q1(a)

The advantages of using extreme programming are as follows,

1. XP not only emphasizes on common teamwork and structural practices but it also emphasizes strongly on technical practices.
2. It performs detailed examination throughout the development process.
3. It provides clear understanding of its limitations and abilities.
4. It eliminates implementation of activies that are not productive thereby reducing the costs and burden of project team.

Q5. List out the advantages of iterative model.

Answer : Model Paper-I, Q1(b)

The advantages of iterative model are as follows,

1. When less number of people are involved in the project, iterative model is the correct choice.
2. With each iteration, the technical risks involved in the project are reduced.
3. The customer can expect at least a core product in a short span of time from the project team.

Q6. What is root-cause analysis?

Answer :

A root-cause analysis is a simple technique that is performed with the objective of determining the main reasons for the problem occurrence. This analysis is generally conducted whenever a bug is detected during development process. It can be used for analysing both major and minor problems.

Q7. What are the benefits of agile teams?

Answer : Model Paper-III, Q1(a)

The benefits of agile teams are as follows,

1. They reduces the costs.
2. They rarely generate bugs like once in a month.
3. They remove unnecessary data by cancelling faulty projects.
4. They replace development practices that are expensive with simpler ones.
5. They perform quick and accurate interaction.
6. They review the processes regularly and makes improvements in code so that software can be handled easily and gets enhanced over the period of time.

Q8. Why process improvement charts are used?

Answer :

Process improvement charts are used for visualizing a process variable over time. These charts are use by the XP team to improve the following,

(i) To improve amount of paring by the tracking the percentage of time spent on pairing and percentage of time spent on flying individually.

(ii) To improve pair switching by tracking the number of pairing combinations made in each iteration.

(iii) To improve build performance by tracking number of tests executed per second.

(iv) To improve support responsiveness by tracking age of oldest support request.

(v) To improve unnecessary interventions by tracking the number of hours spent on every iterations non-story work.

Q9. What are the recommendations for adopting extreme programming?

Answer : Model Paper-II, Q1(b)

The recommendations for adopting extreme programming (XP) are as follows,

1. A brand-new codebase
2. Strong design skills
3. A language that's easy to refactor
4. An experienced programmer-coach
5. A friendly and cohesive team.

Q10. Write any four principles of agile.

Answer : Model Paper-III, Q1(b)

The following are the principles of agile,

1. The developers and business persons should work collaborately.
2. The team members must perform with cooperation which is the most effective form of communication.
3. Self-organizing teams must provide best designs, requirements and architectures.
4. The customer's satisfaction is the first priority which can be ensured by delivering a valuable software as early as possible.

1.1.2 Beyond Deadlines

Q13. Explain why development of successful project does not depend only on meetings the deadlines.

Answer :

A successful project development does not depend on just meeting the deadlines but also depends on the ability of a software to meet various factors like safety, cost, time, quality and customer satisfaction. It is assumed by most of the developers that maintainability is the vital key for delivering a successful project. It provides the ability to correct bugs, repair/replace the defected component, prevent unexpected working conditions, increase the life of a product, increase efficiency, reliability and safety. Sometimes, despite of providing good code still some of the projects might fail badly. Moreover, even the projects that are executed correctly can bring yawns from the users. Therefore, beyond just meeting the deadlines, it is necessary to create values for all the employees who are involved in project development so as to motivate them. But, this cannot be done unless the investors receive high returns on their capital consistently. Thus, it means that success refers to delivering value to an organization. Success can be classified as follows,

(i) Personal Success

Personal success refers to accomplishment of personal goals that one wants to achieve. Lack of personal success will provide difficulties in motivating yourself and employees as well.

(ii) Technical Success

Technical success refers to achievement of technical performance that is needed in project specification. Lack of technical success will end up collapsing the source code under its own weight.

(iii) Organizational Success

Organizational success refers to the success which is achieved by analyzing where the organization lies in terms of its goals and mission. Organizations must execute strategies and retain employees so as to achieve increased and renewable output. Lack of organizational success might make the members of the team feel that they are not required any more in the organization.

Figure: Types of Success

1.1.3 Importance of Organizational Success

Q14. Explain briefly about the importance of organizational success.

Answer :

Importance of Organizational Success

Organizational success refers to the success that can be achieved by analyzing where the organization lies in terms of its goals and mission. Organizations must execute strategies and retain employees so as to achieve increased and renewable output. But, since the technical and personal successes can be achieved more easily, most of the software teams often neglects organizational success. However, most of the senior management and executive does not focus on the effectiveness, maintainability and user preference of the software but rather they care about the output. In other words, investors survey whether they are receiving attractive returns on their capital or not. If not, then the software team is asked to make necessary changes such that they receive more returns. As, it is not possible for the senior managers to provide a different solution for every project, the members of the team are expected to handle all the details. But, if the team members fail to achieve the deadlines and fail to satisfy their manager then, the higher officials has to take intensive measures so as to address the problems. So, they target on cost reduction by using either of these methods.

(i) Set aggressive deadlines to minimize the development time.

(ii) Outsource the work to different countries to lower cost of labour.

Although these two techniques can produce efficiency in the organization but they have their own disadvantages like aggressive deadlines can increases the schedules instead of countries minimizing them and outsourcing the work to other countries can give hidden cost to the organization. Therefore, the project team must not only focus on personal or technical success but must also bring organizational success into consideration.

Organizations Values

Although the project values comes through the sales directly, there should be even more than revenue to the organizational values. The project values are not only measured in terms of dollars or cents, there are also many other sources. The sources of values are as follows,

(i) Competitive distinction

(ii) Projection of brand

(iii) Customer loyalty enhancement

(iv) Meeting regulatory requirements

(v) Original research

(vi) Strategic information.

PART-B

ESSAY QUESTIONS WITH SOLUTIONS

1.1 WHY AGILE

Q11. How Agile development is popular? Explain briefly.

Answer : **Model Paper-III, Q2(a)**

Agile is an iterative approach which helps in managing the projects during software development. Here, each iteration is designed in such a way that it can be delivered in short time span. It enables the software team to respond to the issues that may occur throughout the development and make the required changes to the project at right time.

Agile methodology is popular and it is widely used by most of the big companies like Google, Yahoo, Symantec, Microsoft and so on. One of these companies has also changed its name to 'Agili' so as to create awareness among the users. Moreover, it can be expected that the big consulting companies may soon start offering the certified Agile processes and certified Agile consultants at extremely high price.

Agile development was brought into existence inorder to overcome the drawbacks faced by traditional software development methods that lacked flexibility and productivity. Agile approach performs continuous activities which inturn makes it flexible and agile. This model provides the following benefits,

1. It enables the developers to deliver a better quality product.
2. It provides excellent visibility.
3. It considered to be less risky as the developer keeps receiving feedback after every iteration and can make the required changes in the subsequent phase.
4. It is cost effective.

In the year 1986, it was anticipated by Brooks that by the year 1996, none of the technology (or) management technique would provide a denary increase in productivity (or) reliability (or) simplicity, But, however even agile is not a silver bullet. It has no place for novice programmers i.e., only the senior/experienced programmers are capable of making the necessary changes that are required during development phase. Although, Agile methods provide increased productivity but still there is no evidence of improvement in quality.

1.1.1 Understanding Success

Q12. What is the traditional idea of success?

Answer :

The true meaning of success is often thought to be associated with on time delivery, on budget and as per the customer requirements. Traditionally, success is defined in various forms, some of them are as follows,

(i) Successful

A project that includes all the features and functions as per the user's specification and gets completed within the specified time and budget is referred as "Successful".

(ii) Challenged

A project which possess less number of features and functions when compared to the user's requirement and exceeds the estimated time and budget is referred as "Challenged".

(iii) Impaired

The project which gets terminated at some point during its development process is referred as "Impaired".

According to the CIO magazine, although some of the projects might meet the criteria for success, still they may end up in failure because of their inability to attract the desired users (or) due to their inability to add much value to the business. On the other hand, the projects that are regarded as failure as per the traditional IT metrics might end up in success if they are widely opted by the users thereby providing unexpected value to the business.

1.2 HOW TO BE AGILE – AGILE METHODS

Q16. Explain what does it mean to be agile.

Answer :

Agile

Agile refers to a general philosophy which describes groups of methods and frameworks that represent principles and values of agile manifesto. It works by dividing the project into various functional increments known as "user stories". Then, these stories are prioritized and delivered in a span of two weeks which is referred as an "iteration".

Key Practices of Agile

The following are the key practices of Agile,

(i) Version Control

This system enables the tem to manage and track the changes that are made to source files. It provides deeper understanding of the code to the user. It helps in team collaboration and increases the speed of delivering the product.

(ii) Setting Coding Standards

Coding standards are the rules that programmers must follow during code development. It reduces the risk of bug occurrence. If a common standard is applied consistently throughout the development then it is easier to maintain and extend the code.

(iii) Giving Weekly Demos to the Stakeholders

A team meeting is conducted in every two weeks so as to discuss about how much work has been completed so far. This meeting is attended by the development team and stakeholders (managers, users, other teams, investors, sponsors, customers and so on). The development team demonstrates about the work done during the current sprint and discusses the problems they met on their way. Every member in the the meeting is asked to suggest the possible improvements that can be made to maximize value of the software. The goal of this demo is to keep the progress of product development.

Agile Methods

Extreme programming and scrum are the two agile methods that methods that support agile philosophy. These methods combine the agile practices to develop a lean, powerful and self reinforcing package.

XP is the most important software development methodology of the agile frameworks. Code review, testing, incremental development, simplicity, Design of Good quality and integration testing are some of the good practices of XP. It is dependant on the frequent iteration through which user stories are implemented by the developers. Depending on these user stories, the development team presents their perception of how the system would work. Then, the team decides to construct a simple program inorder to explore the suitability of solution being proposed. One of the most important aspect of XP is to understand the requirements of customers exactly and so it contacts the customers frequently for receiving feedback. XP mainly focuses on few specific features for developing a system that will work efficiently in present time rather than wasting time on speculations of future requirements.

The first and foremost step to be agile is to conduct stand up meetings. These stand up meetings are generally referred as scrums. It is a 15 minutes meeting where the members of the development team discuss about their accomplishment since the last meeting, about their current work and the roadblocks in their development. Scrum is one of the implementations of agile framework where incremental builds are delivered to the customers after every iteration. It is usually implemented in projects where the requirements change rapidly. It enables the team to work together and help them in creating better results faster.

Q17. What are the principles of Agile?

Answer : Model Paper-I, Q2(a)

The following are the principles of Agile,

1. The developers and business persons should work collaborately.
2. The team members must perform with cooperation which is the most effective form of communication.
3. Self-organizing teams must provide best designs, requirements and architectures.
4. The customer's satisfaction is the first priority which can be ensured by delivering a valuable software as early as possible.
5. The team leader of the project should pay continuous attention to get a good design and technical excellence.
6. The primary measure of progress is the working of software.
7. Working software should be delivered frequently within weeks instead of months.
8. The team members must emphasize on how work can be done more efficiently and must be read to adapt change.
9. The amount of work being done must be maximized to the possible extent.
10. The team members must always be read to implement any changes in the software as per the customers even in last stage of development.
11. The projects should be developed through motivated individuals, who deserve support and trust to get the job done.
12. The development process should be sustainable and developers, users and sponsors must maintain stable speed to the possible level.

1.1.4 Introduction to Agility

Q15. How agility helps in attaining success?

Answer : Model Paper-III, Q2(b)

Agility

Agile methodology plays a vital role in enabling an organization to become more successful. It helps in attaining personal, technical and organizational successes by minimizing various issues as follows,

Organizational Success

The agile methods mainly emphasizes on the delivery value and cost minimization so as to achieve organizational success. It sets the desired target during the inception phase itself, so that incase of organizational failure, organization can detect the errors at early stage and can terminate it before spending huge amount of money.

Agile teams particularly includes the business experts who focus on the development efforts on the core values of the organization provided by the projects. These agile projects generally releases the valueable features first and then it keeps releasing other versions in short durations. According to the business needs and new information discovering the agile teams also changes its direction to match with those and comes upon with the experienced agile teams, unexpected opportunities to improve its plans.

Benefits of Agile Teams

1. They reduces the costs.
2. They rarely generate bugs like once in a month.
3. They remove unnecessary data by cancelling faulty projects.
4. They replace development practices that are expensive with simpler ones.
5. They perform quick and accurate interaction.
6. They review the processes regularly and makes improvements in code so that software can be handled easily and gets enhanced over the period of time.

Technical Success

Extreme programming is the most specific form of the frameworks that mainly aims at achieving technical success. In XP, all the people who play some part in development of a project work together on a daily basis so as to accomplish specific outcome. They make sure that atleast two of the programmers review each piece of code and perform continuous code integration. This helps the programmers to detect bugs and fix them sooner.

The entire team concentrates completely only on one project feature before starting the new one. This prevents from the unexpected delays.

They also incorporate advanced technical practices that results in technical excellence. Once of the most well-known practices is 'test-driven development' which helps the programmers to write the code that exactly generates the output they want.

Benefits of XP Teams

XP teams develop simple and ever-evolving designs which can be modified easily incase of any plan variation.

Personal Success

The personal success is private and satisfactory. Although, agile does not meet all the requirements for personal success but when it comes in use, then one can be attracted to its features. Some of the factors are as follows,

(i) Senior Management and Executives

The efforts made by the team in generating huge returns to the investors and providing durability of the software are appreciated by senior management and executives.

(ii) Domain Experts, Users, Product Managers and Stakeholders

They appreciate the teams ability in handling a software. Because, the team supply useful, valuable software with high delivery frequency.

(iii) Product Managers and Projects

They appreciate the team's efforts in making necessary changes as per the business requirements and providing improvised stakeholder satisfaction.

(iv) Developers

They appreciate the improved quality of life which resulted due to the greater technical quality, increased impact over the estimates and schedules and also the team autonomy.

(v) Testers

They appreciate the entire team for bringing changes in quality in all stages of project and for providing less repetitions work which is very challenging.

Therefore, the agile methodology changes the perspective of developer. Though, delivering of software involves huge work, but when it is carried rigorously using agile, one may find it very easy and simple.

Q18. Illustrate the manifesto for Agile software development.

Answer :

The project developers may sometimes intend to create their own agile method by combining practices from the agile methods. But, there is no practice or process that is inherently Agile. In other words, agile has no stable from. Since, every project is different it is not necessary that if some thing works for one project then it will work for the other project too. The requirements of customers may change over a specific period of time. Thus, the workflow must be reevaluated continuously using the principles of agile. If the team does not ensure whether the workflow is adapting to changes in user requirements then this may result in project failure. So, instead of making your own agile method continue working with an existing and proven method. Moreover, it is also essential for the project team to have better understanding of the manifesto for agile software development.

The manifesto for Agile Software Development

(i) Individuals and Interactions over Processes and Tools

The first principle of agile manifesto is to value the project team and their interactions rather than processes and tools. When every member in the team is allowed to contribute in development process then the result can be powerful. This makes communication clear, quick and effective. Enabling frequent interactions among the team makes the teamwork strong and team members can be self organized.

(ii) Working Software over Comprehensive Documentation

The second principle of agile manifesto is to value working software over comprehensive documentation. It means that writing documentation must not hinder the progress of working of a software. By writing comprehensive documentation you may end up in designing the parts of the software that are not needed and the documentation will be out of date by the time it gets completed.

(iii) Customer Collaboration over Contract Negotiation

The third principle of agile manifesto is to value customer collaboration over contract negotiation. This implies that project team and customers must be encouraged to work collaborately instead of viewing one another as adversaries. Inorder to develop successful software, it is necessary to communicate with the customers and end users regularly along the entire development process.

(iv) Responding to change over following a plan

The fourth principle of agile manifesto is to respond to change over following a plan. Planning is an ongoing activity which is throughout the project development. It must not be too detailed but should be simply enough to provide the team with basic objectives. The code must be designed in such a way that it can adapt changes in technology or stakeholder's priorities easily.

1.2.1 Don't Make Your Own Method

Q19. Why creating a brand-new agile method is a bad idea? Explain.

Answer : **Model Paper-II, Q2(a)**

The user on create their own agile methods by combining of already existing agile practices. At first, the task of creating their own agile methods appear to be easy, but the user faces difficulty as he get numerous good agile practices for selection. So, without knowing about agile development where user never used them earlier then creating directly a brand-new agile method is a bad idea. It is believed that simply writing code is not enough in programming, the user should get the output. So, similarly there is more to agile development than just practices.

The practices are nothing but the agile fundamental principles. It is necessary to understand there principles thoroughly and also, it is advisable to master the art of agile development. So, unless, the user is not skilled in this, he or she is unlikely to select right practice.

The agile practices performs the work more number of times like twice and thrice in solving the multiple problems concurrently in the software development and also substantiates each of them in smart and extraordinary ways.

As, all the projects and situations are distinctive, so it is recommended to have an own agile method which is designed to solve the particular situation. Therefore, instead of designing agile method from beginning, it is better to take the existing one which would be used and tested. This can be refined repeatedly. Then, later on, it can be applied according to the situation of the project and can be recorded at those points where it is working and not working. Also make a precise estimation about the improvement and iteration.

Q20. Why extreme programming is considered as a well-defined agile method?

Answer :

Extreme programming is considered as a well defined agile method as it provides the following advantages,

1. XP not only emphasizes on common teamwork and structural practices but it also emphasizes strongly on technical practices.
2. It performs detailed examination throughout the development process.
3. It provides clear understanding of its limitations and abilities.
4. It eliminates implementation of activities that are not productive thereby reducing the costs and burden of project team.
5. It enables customers to make wise decisions based on the budget.
6. It helps in increasing employee satisfaction and retention.
7. It saves time required for project realization.
8. It ensures that client receives exactly what they require.
9. It helps the team to increase their speed in software development.

1.2.2 Road to Mastery

Q21. What are the steps to be followed to master the art of agile development?

Answer :

The Road to Mastery

To master the art of agile development, it is necessary for the user to gain real word experience. This experience can be acquire, by implementing a particular and well defined agile method. So, inorder to accomplish this, extreme programming is used. It has the following advantages,

1. The extreme programming (XP) is one of the most complete agile method, in all the agile methods. Apart from team teamwork and structural practices, the XP gives atmost importance to technical practices.
2. The XP was designed after making it undergo multiple testing phases. It consists of thousands of pages of explanations, experience reports and assessments. In addition to this, it also addresses the XP's capabilities and limitations.

There are some of the practical suggestions shared by the experts which helps in applying XP more easily. The steps to be followed in mastering the art of agile development i.e., by using XP are as follows,

1. The experts should put their question forward that, incorporating agile method brings success to the team and organization or not? If yes, then how?
2. Identify, if the art of agile development approach works to the team (or) not?
3. Incorporate maximum XP's practices. As these practices are self sustainable and prove to be highly efficient when used all together.
4. Adhere to the XP practices strictly and consistently. However, in doing so, practice fails the experts must try to follow more of a book approach.

   The novice teams mostly under-utilize their practices this is because they expects the starting two (or) three months to be relaxed with the practices and later two to six months becomes as second nature.
5. Once, the team becomes confident, i.e., they are assured it is known that the practicing of XP by the team is proper and exact. With this, they give more months to establish new experiment by making some changes which are not in agile development. In this new experiment, every time the changes being done are observed and recorded, so that improvements can be accelerated.

The road to mastery representative of the agile development is as follows, where it consists of learner, passenger and expert.

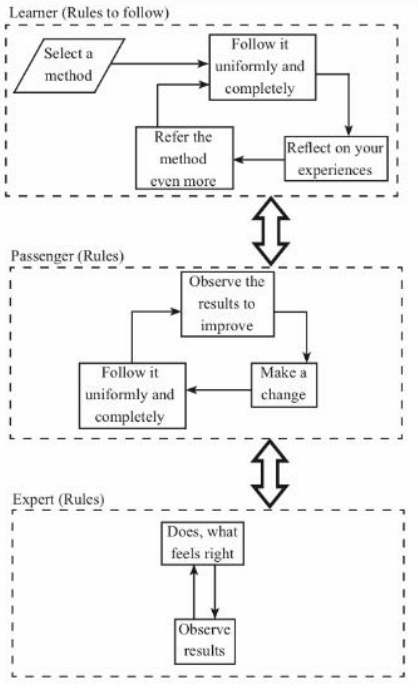


Figure: Road to Mastery

Q22. Explain the need of a mentor.

Answer :

Mentor

The implementation of XP can entail lot of problems and challenges to the organization. Based upon the situation, numerous solutions are provided to common problems. However, some situations are left unsolved. In such type of situations, there is a need of an exterior expert who has mastered the art of agile development i.e., a "mentor".

In finding a mentor, the difficult part is to find someone with an adequate experience in agile development. The sources for this are as follows,

(i) In one organization, different groups practicing XP.

(ii) In one area, different companies practicing XP.

(iii) An agile user group/A local XP.

(iv) The Agile consultants/The XP.

(v) The XP mailing list : extremeprogramming@yahoogroups.com

A mentor may not anticipate each and every difficulty the project may encounter, but certainly he can help when the things are going wrong. With the art of agile development, if the practices of own couldn't substantiate any betterment week-wise, then it could be an explicit sign of trouble in the project. As a result, the project should be slow down until next week and during this period figure out the errors and take help of a mentor.

The right solutions can be facilitated if the detailed situation is explained. The mentor can help out in addressing difficulties in the project and present an definite suggestion based on the situation.

1.3 UNDERSTANDING XP (EXTREME PROGRAMMING)

Q23. Write about extreme programming (XP).

Answer : Model Paper-I, Q2(b)

Extreme Programming

XP in a Software development methodology is intended to improve software quality and responsiveness with respect to changing customer requirement. It is an important and well-known approach among the agile processes. It is a method in which the changes in the requirements are inevitable and during the development process, the software should be capable of accepting them. Hence, a lightweight and quick to respond development process is built for this purpose. Thus, the software must be developed in small iterations. Also, it must avoid the usage of detailed and multiple documents that are difficult to maintain. In order to ensure that the changes are accepted quickly and correctly, the software must have a very simple design structure, a continuous feedback and a face-to-face communication.

Apart from this, it also makes use of the ideas of team headfulness, egoless programming and chief programmer teams. Moreover it can be considered as a parameter of collective mind. It employs the following approaches to improve communication and coordination in a standard software development environment.

- Providing more documentation.
- Enhancing essential software products such as software code and test data. For instance, coding conventions are followed so that the code clearly reflects its purpose. Also, test cases and expected results are generated prior to the code forming a good specification.
- The user representation should be made available to deal with the user needs.
- Integration tests must be applied repeatedly so as to effectively integrate the software.
- The pair of developers just like chief programmer and co-pilot must be appointed to support software development.

The five values which provide a foundation for all work performed as a part of XP are as follows,

1. Communication
2. Simplicity
3. Feedback
4. Courage
5. Respect.

1. Communication

XP focuses upon informal collaboration between customers and developers to procure good communication between software engineers and stakeholders. Apart from this, it also sets up an effective figure of speech or image for interacting concepts, continuous feedback and also preventing the use of large bulky documentation as communication medium.

2. Simplicity

XP limits the developers to plan only necessary requirements but not future requirements. The idea is to achieve a simple design which can be restructured later.

3. Feedback

XP provides the feedback through three sources which are the deployed software itself, customers and other members of the software team. The feedback to the agile team is obtained by planning, implementing effective testing strategy and the software.

4. Courage

XP demands courage by following its practices strictly. It is often referred to as discipline. The designing team anticipates that designing and planning for future will definitely conserve the time and efforts in later times. But the agile XP team should be discipline enough to design for today since the future requirements might vary. Thus, this requires large amount of work, for code implementation and designing.

5. Respect

Once all these values are implemented, a respect is developed among members of team, among stakeholders and for software itself. After delivery of software, increment team inculcates respect for XP process.

Q24. Explain about waterfall model and iterative/ incremental model.

Answer :

Waterfall Model

It is also known as classic life cycle model, which divides the entire software development process into five main phases. Following is the diagrammatic representation of waterfall model,

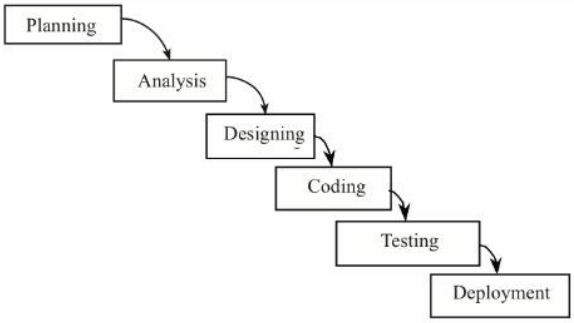


Figure: Phases of Waterfall Model

1. **Planning Phase**

   This forms the initial phase of software development process, where the customer requirements are retrieved for the preparation of customer requirements specification.

   There are two important facts about this phase i.e.,

   (a) Project initiation and

   (b) Requirements gathering.

2. **Analysis Phase**

   In this phase, the entire project strategy is devised. The main addressable issues in this phase are as follows:

   (b) A thorough analysis of resources requirements is made i.e., estimation

   (c) Number of people involved in the project is determined i.e., scheduling

   (d) Duration and expected cost of the project are estimated i.e., tracking

3. **Designing Phase**

   In this phase, the entire project scenario is analyzed, designed diagrammatically. Designing remains important for successful completion of the project.

4. **Coding Phase**

   In the coding phase of the software, each component of the software is coded and is suitably integrated.

5. **Testing Phase**

   Once the coding part is completed, the entire software is thoroughly tested.

6. **Deployment Phase**

   This forms the final phase where the software is usually delivered to the end users for its effective implementation. Later, feedback is collected from the users and if, the software fails to meet the users requirements, it is modified. Hence, the important activities involved in this phase are,

   (a) Perfect delivery

   (b) Support

   (c) Feedbacks.

Iterative/Incremental Model

It is also called as incremental model. Following is a diagram depicting an overview of integrated model.

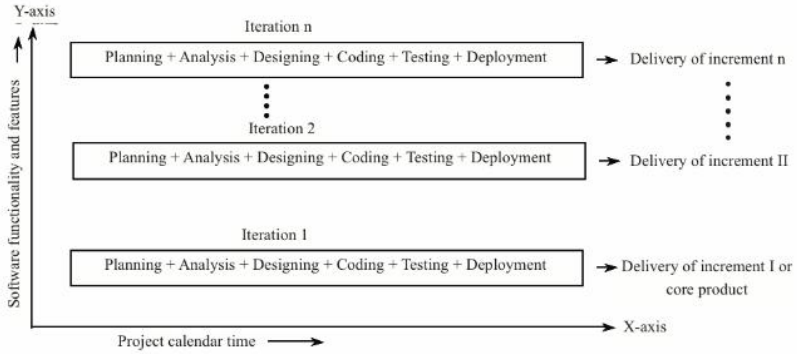


Figure: Representation of Iterative Model

It can be observed from the above figure that the iterative model divides its process into certain number of increments where each increment comprises of six phases of waterfall model. At the end of each iteration, an effective module of a given software is developed. The main purpose of this software is to extract any number of files from a given large source. The main components of this software would be,

(a) File extraction

(b) Playing of file

(c) Selection of initial point

(d) Selection of final point

(e) Storing of abstracted file.

In this process, the software components i.e., file extraction, playing of file can be developed in first iteration, selection of initial and final points in the next iteration and finally storing of abstracted file in the final increment.

Hence, a core part of the entire software has resulted from the first iteration. Later this core part is delivered to the users and feedbacks are claimed. Based on these feedbacks and with an intention to provide certain new functionalities to the software, next iteration is made. These iteration are repeated until the whole project is completed.

Advantages

1. When less number of people are involved in the project, iterative model is the correct choice.
2. With each iteration, the technical risks involved in the project are reduced.
3. The customer can expect at least a core product in a short span of time from the project team.

1.3.1 XP Life Cycle

Q25. Explain about the phases involved in the extreme programming (XP) life cycle.

Answer : Model Paper-II, Q2(b)

The various XP processes include,

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

1. Planning

The planning phase initiates by listening to the activity of requirements gathering. This allows the technical members of the XP team to comprehend the business context with respect to software and also get clear idea about generated outputs, features and functionalities. Additionally, the task of listening produces set of stories generally referred to as user stories. These stories specify necessary output features and functionalities of the developed software. Every individual story is developed by customer itself and is located upon the index card. Then the customer depending upon the value of the feature or function, allots the priority value to the story. Now, the members of the XP team computes every individual story and assigns cost to it which is guaged in development weeks. But if the story demands more than three weeks then the customer is advised to divide the story into smaller stories and later assigns priority values and cost to it.

The decision of integrating the stories into next release is done by the customers and developers after collaboratively working with each other. The next release is basically a software increment which is created by XP team. After making the commitment of release, delivery date and various project matters, the XP-team suggests three ways to order the stories. They are as follows,

(i) Stories must be implemented quickly within few weeks.

(ii) Stories holding highest values must be scheduled first and implemented first.

(iii) Stories with controversy must be scheduled first and implemented first.

2. Analysis

In analysis phase, the onsite customers are involved, they are the best qualified people in determining the software work. The onsite customers may not be real customers but they work with the team and are responsible for illustrating the software requirements with their own knowledge and combine with the traditional requirements gathering methods.

Since, the customers are responsible for managing their work and identifying the software requirement they are always ready before the programmers asks the information. They also draw some general requirements and detailed requirements before the programmers estimates and implements them. Some of them may be difficult to understand but the customers control them with the help of testers by creating customer tests which could be detailed, automatically checked examples. To collaborate with the communication, programmers use a ubiquitous language to design and code. Sometimes, the teams also make use of interaction designer (i.e.,) UI for sketching on the application screen.

3. Designing

The designing phase in XP operates upon a principle called "keep it simple". Most of the developers prefer simple design representation rather than complex representation. The developers provided with guidelines for implementation of story. Also, the XP makes use of a mechanism called CRC cards (Class-Responsibility Collaborator) for the software operating in object-oriented context. This is incorporated to carryout designing.

Moreover, if any complexity arises during the designing of the story, the XP makes use of operational prototype corresponding to that part of the design called spike solution. It minimizes the risk at the time of initialization of implementation and then verifies the correct estimation for the story which holds the design problem. Another technique called refactoring is implemented which is a construction technique for design optimization.

The refactoring can be specified as a technique which makes modification in software without changing the external behaviour of the code but enhances the internal structure. Simply, it deletes the code which introduces bugs. Thus, refactoring is nothing but bringing improvements to the design.

The designing is basically a temporal artifact which requires changes as construction moves on. So, the refactoring operates upon this by making minor changes in design inorder to enrich the design.

4. Coding

The coding phase initiates by conducting unit testing consecutively. This test is applied upon every individual stories which are to be included in latest release. In general the unit test provides the developers an idea what is to be implemented inorder to validate the test.

The coding activity emphasizes upon an idea called pair programming. In essence, two people can operate upon a work station to develop code for story. Thus, exhibiting a real-time mechanism for problem solving and real-time assurance or reviewing. Once the pair-programmers are done with their work, the developed code is combined with the other works. This activity can be preformed on regular basis by the team.

5. Testing

In general, the XP supports a complex line of testing practices. The testing phase is one of the major step in the XP process where the programmers provides the basic defending in testing with the help of Test-Driven Development (TDD). it generates an automated unit and integration tests. In some cases, the programmers also establishes end-to-end tests which helps in making sure that the software operates according to the programmers planning (or) intention.

The testing phase is one of the major step in the XP-Process. The unit testing is carried out using a framework inorder to automate itself. This step leads to the strategy called regression testing which is used to change the code. Here, single unit tests are evolved as universal testing suite with which the XP-team can carryout both integration and validation regularly. This helps the XP team to monitor the progress and can raise alarms at the time of errors.

In addition to this, the XP-team also carries out an acceptance test which is nothing but a customer test. It is conducted upon customer and gives the idea about complete system features and functionalities reviewed by the customer.

Inorder to check if the testers efforts are actually giving high returns the testers provides the quality code certainty by making use of experimental testing. This testing checks the errors in software. Once the detect the errors the apply root-cause analysis activity and improvises the process to prevent the occurrence of same errors in future. They also check the software's nonfunctional features like functioning and reliability. So, that it can be easy to the customers in deciding to the customers in deciding to create the stories (or) not.

6. Deployment

XP teams ensure that their software is always ready, so that it can be installed whenever required at the end of any iteration. They deploy to the end users for the week-wise iteration demo. But for real customers it is deployed based the business needs.

The software released by the XP team is supported in following ways,

(a) It is supportive, as long as the team is operating.

(b) It support its own software depending on the organization.

(c) It is supportive, sometimes by separate team also.

In the same way, as the project is ended, the team of that ending project may hand over its project to any other team. This requires documentation which provides necessary explanation for the remaining end part of the project.

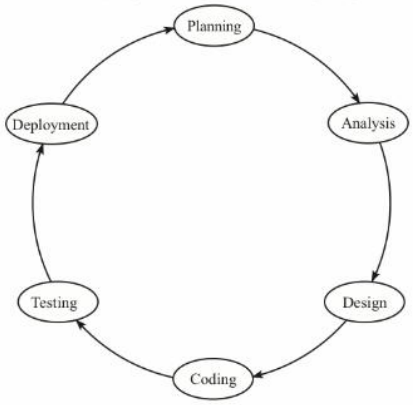


Figure: Life Cycle of XP

Q26. Discuss briefly about the working of XP.

Answer :

Working of XP

XP team handles almost all the software development activities concurrently. These activities include analysis, design, coding, testing and deployment which are performed repeatedly in a short duration.

XP concurrently performs all the phases or activities through iterations. In essence it proceeds by week-wise incrementing the work. Every week, the team has small part of release plan design, coding, testing etc., and works on the small parts of features on stories which have the customer's value and commits to deliver the stories for about four to ten per week.

Throughout the week, the team works on developing the story with all the phases and at the end of the week, the team deploy for the internal review of the software.

The diagramatic representation of working of XP (Extreme Programming) is as follows,

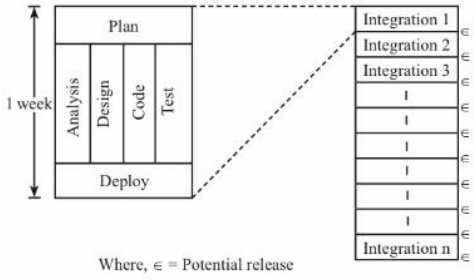


Figure: Working of XP

1.3.2 XP Team

Q27. What is meant by XP team? Why software development team is different? Explain the functions of different people involved in XP.

Answer : Model Paper-I, Q3(a)

XP Team

The XP teams are cross-functional and self-organized. They work together as a single team towards a common goal and deliver successful software. XP team has two important significances. They are as follows,

1. XP teams are responsible for their own success.
2. They involve all the necessary proficiencies to achieve the goal.

XP team consists of different people who perform various tasks during the development of software. They are,

(i) Programmers

(ii) Designers and Architects

(iii) Product manager

(iv) Domain experts

(v) Interaction designers

(vi) Testers

(vii) Project manager

(viii) Coach.

(i) Programmers

The XP team consists of 4 - 10 programmers which includes atleast one senior programmer, designer or an architect. They are responsible for determining the implicit way of delivering stories, providing effort estimates, suggesting alternatives, reducing the cost of the product and helping the customers in achieving the desired plan with the help of planning game. By means of test-driven development, programmers write tests, implements code, refactors, integrates design and design the applications. Since they are aware of the impact of technical debt on the development time and future maintenance costs. They mainly focus on the quality of software. They make sure that customers prefer to release the software at the end of any iteration.

(ii) Designers and Architects

Although Xp team employs Test-driven development (TDD) for combining the process of designing, testing and coding into single ongoing activity. But still they require expert designers and architects. They facilitate the team in their incremental design and architectural efforts. Moreover, they also help the XP team by simplifying into easier form therefore designers and architects serves as programmers.

(iii) Product Manager

The product manager of an XP team is responsible for maintaining and promoting the product vision. In other words, product managers document the vision, shares it with the stakeholders, incorporates the feedback, creates stories, set priorities, review the work progress, lead iteration demos and deal with the organizational politics. The best product manager possess good understanding of their market.

(iv) Domain Experts

Domain experts are responsible for analysing the details of the software. They spend most of the time with the team and draw the details for further stories so that they can be ready with the answers whenever the programmers questions them. They are also called as "Subject matter experts" in their field. For example - financial analysts and PHD chemists. The rules made by the are called as "Domain rules" and the knowledge of these rules is called as "Domain knowledge".

(v) Interaction Designers

The interaction designers are responsible for creating attractive and functional user interface which is the public face the product. They help in defining the UI product, understanding the needs of users by interacting the users in various forms like interviewing users, creating user personals, reviewing paper prototype with users and also observing the actual software usage. They divide their time between working with teams and users by contributing to the release planning by suggesting the team on the user priorities and needs. They iteratively refine their models with the iterative refinement of the program itself simultaneously.

(vi) Testers

Testers helps in producing the quality results by thinking critically with all their skills to help the customers in all the possible ways when the product is being visualized. They also act as technical investigator for the team in identifying the prevention of errors is successful or not by using an provisional testing. They don't test the software completely for the errors but provides the information about the software non-functional characteristics like stability, performance and scalability which includes both exploratory testing and long-running automated tests.

Testers who use the self-directed work are the best suited one in XP. They require creative thinking, flexibility and experience in defining test plans. There should be one tester for every four programmers i.e., (1 : 4) ratio.

(vii) Project Manager

The project manager is responsible for helping the team to work with rest of the organization by coaching the team with non-programming practices. Some of the functional managers also suites for playing this role.

(viii) Coaches

The leaders of XP are referred as "Coaches". They facilitate the team by organizing shared workspace and ensuring that there are right members in the team. It also assists the team in setting the conditions for the energized work and develop an informative workspace by communicating with other members of the organization. Moreover, they provide support to the team members in maintaining self-discipline, persisting on control of challenging practices like test-driven development, risk management, slack, incremental design and architecture. The main motive of coach is to make the team succeed.

Programmer Coach

A programmer coach is required by every XP team inorder to assist the other programmers with technical practices of XP. They are often known as "Senior developers" and has titles like "technical lead" or "architect". They serve as normal programmers and contribute completely in the development of a software.

Q28. Briefly explain the following terms,

(i) Whole team

(ii) Team size

(iii) Full-time team members

(iv) On-site customers

(v) Project community.

Answer :

(i) Whole Team

The whole XP team works together in an open workspace and discuss about a series of activities such as iteration demo, retrospective and iteration planning for about 2-4 hours in first iteration. They also conduct daily stand-up meetings for about 5-10 minutes. Apart from these activities each of the team members are expected to perform their own individual task also. This self-organization is considered as an hallmark of agile teams.

(ii) Team Size

The XP team sometimes includes 5-20 team members wherein there are 4-10 programmers, 6 customers, 3 testers and 1 project manager. Since, the size of the team is large it requires high level communication and reduces the individual productivity. But, this overhead can be avoided by hiring more experienced and productive member. However, XP team usually comprises of 12 members which includes 6 programmers, 4 customers, 1 tester and 1 project manager and the smallest team consists of 5 members with 4 programmers in which one programmer can also act as coach and 1 product manager acts as an project manager, domain expert and tester.

(iii) Full-time Team Members

The team members who give their full time to the project with complete attention are called as "Full-time team members". Some organizations like matrix-managed organization assigns multiple projects at a time to the team members called as "fractional assignment" which improves the productivity frequently.

(iv) On-site Customers

The on-site customers are often referred as "Customers". They are responsible for defining the software, determining the requirements of stakeholders, refining the plans and providing the programmers with their requirements. The most important activity performed by the customers is "release planning" which is the multifaceted activity. In addition, onsite customers must develop an achievable plan by coordinating with programmers and help them in knowing the requirements by creating mock-ups, reviewing work progress and developing complete customer tests so as to resolve complicated rules of business.

(v) Project Community

The project community refers to all the members of XP team who contribute in development of software. Human resource management and facilities department are the two essential members of project community. The performance reviews and compensation are handled by the human resources in the project community whereas facilities departments help the team in developing an open workspace.

Q29. Write about the following terms of XP teams,

(i) Business analysts

(ii) Technical specialists

(iii) Stakeholders

(iv) The executive sponsor

(v) Filling Roles.

Answer :

(i) Business Analysts

The business analyst serve as intermediary between the customers and developers by analyzing and improving the customer requirements wtihin a functional requirements specification. They facilitate as a medium in order to support the other member of onsite customers. They assists the programmers in expressing the technical trade-offs in business terms.

(ii) Technical Specialists

The technical specialist has their own area of expertise which help them to develop technical solutions that support business requirements. However, though each and every member in the team have a specific role, all the member must come into play for development of an effective software.

(iii) Stakeholders

The stakeholders are group of people (end-users, purchases, managers, executives) who forms a large subset of project community by showing active interest in success. Eventhough, they don't play any role in the routine development, but they are involved in iterations demo. The needs of stakeholders are handled by the onsite customers.

(iv) The Executive Sponsor

The executive sponsor is an ultimate customer who requires regular demo's of the project and ensures that the project is running accordingly.

(v) Filling Roles

In XP, it is not always necessary to hire a specific member to play a specific role in the project development. Sometimes, multiple roles can be played by a single team member. For instance, a product manager who is generally a domain expert can also handle tasks performed by a project manager and similarly most of the programmers who are generalists can understand various technologies. Therefore, in the absence of any members, both programmers and testers should pick up the speed.

1.3.3 XP Concepts

Q30. Write about extreme programming (XP) concepts. Explain the common ideas followed.

Answer : Model Paper-I, Q3(b)

XP Concepts

The following are the concepts of extreme programming (XP).

(i) Refactoring

(ii) Technical debt

(iii) Timeboxing

(iv) The last responsible moment

(v) Stories

(vi) Iterations

(vii) Velocity

(viii) Theory of constraints

(ix) Mindfulness.

(i) Refactoring

Refactoring is a process of improving the existing software without altering its internal mechanisms. Hence, when a given software is said to be refactored it means that, all the loopholes existing in it such as inappropriate data structures or inefficient algorithms or certain redundant data etc. are corrected and what remains is the code with improved capabilities.

(ii) Technical Debt

Technical debt refers to the cost that has incurred due to the additional rework performed by selecting of an easy solution rather than using a better approach which would consume more time. This method is generally adopted when the customer detects any bug in the code. It is emerged during the life of a project. It is not easy to estimate roughly the time that may be required to fix the bug. Since, it may consume either minutes to hour or may even extend to half-day.

XP considers technical debt as an efficient approach to communicate about the requirements for refactoring and improvement in task corresponding to the source code and its architecture.

(iii) Timeboxing

The specific period of time allotted for accomplishing a tasks is referred as timeboxing. It is both difficult and valuable. It helps the team to work consistently towards achievement of their goal. However, it is not easy to recognize the point where information is enough, but it is an essential time management skill which can be developed through timeboxing approach.

(iv) The Last Responsible Moment

In XP, every potential change is regarded as an opportunity to learn something significant and so the XP team put off their commitment till the last responsible moment. According to Poppendieck and Poppendieck, it is not a good choice to delay the commitments beyond the last responsible moment. Delaying the decisions till the last point may increase the decision accuracy, reduces the workload and decreases affect of changes. It also gives time to increase the amount of data gathered so as to make a presiding decision. As a result, this reduces the workload and possible amount of rework that may occur from an incorrect decision. Hence, the changes becomes simple as they are probably less likely to invalidate decisions or cause additional rework.

(v) Stories

Stories are represented as independent and unique elements of the project. They are generally used to describe features of a product. They are written from the perspective of customers and are recorded on index cards. The stories may be written by different stakeholders or development team member depending on the project.

(vi) Iterations

An iteration is a complete cycle of designing, coding, verifying, and releasing procedures. Each iteration consumes a time period of one to three weeks. It is initiated with the end user stories and ends with development of an efficient software.

The staring point of every iteration indicates a point where the customers can modify the direction of project. Small iterations allows more frequent adjustments whereas the fixed-size iteration provides well timed rhythm of development. Although small and frequent iteration includes more planning overhead, the amount of planning is proportional to the length of iteration.

(vii) Velocity

The process of estimating the amount of work done in an iteration is known as "velocity". In other words, it is mapping of estimation to an calendar and has no relation to the productivity. The units of velocity measured are deliberately non-specific. It allows the team to make commitments and predict the release dates of iteration.

(viii) Theory of Constraints

In XP, every system contains one constraint for determining the overall performance of the system. The programmers are the constraints of XP team. Regardless of the work done by the customers and testers, the software team can complete their work only as fast as the programmers can do. Therefore, if the other members of the team develop more quickly than the programmers then the work falls out of date and requires reworking thereby slowing the programmers work further. Thus, until the programmers are constraints the customers will have sufficient time to complete their work before the programmers requires it.

(ix) Mindfulness

The ability of a team member to react effectively to change is referred as mindfulness. Sometimes the changes maybe difficult to analyze but XP provides several opportunities to collect feedback from the code, co-workers and from every activity performed in the process. So, team member must utilize these opportunities and respond to changes efficiently.

1.4 ADOPTING XP – KNOWING WHETHER XP IS SUITABLE

Q31. What are the pre-requisites and recommendations for adopting XP?

Answer :

Prequisites of XP

The following are the prerequisites for adopting XP. They are,

1. Management support
2. Team agreement
3. Collocated team
4. On-site customers
5. The right team size
6. Use all XP practices.

4. An Experienced Programmer Coach

A person filling this role is the one to mentor a team. Every programmer coach must possess skills in system's architecture, design, development and programming in object oriented, relational database and client-server platforms. In addition to these skills, the team members expect their coach to guide them in making right decisions. It is essential to have a good programmer coach so that they help the team to practice XP consistently and rigorously.

5. A Friendly and Cohesive Team

The work which is done in teams provides many benefits. If there is cohesiveness in the team then it helps the team members to achieve their goals in a friendly team environment. On the other hand, lack of cohesion affects the performance of the team due to unnecessary stress among the members.

1.4.1 Implementing XP

Q32. Write about the process of applying XP to a Brand New project.

Answer : Model Paper-II, Q3(a)

Implementing XP in Brand-New Project

When XP is applied to a Brand-New project, then the development team requires atleast three or four weeks to gain complete information about XP. Because, during the first stage of development, the on-site customers will just work on the release plan and the programmers focuses on establishment of technical infrastructure, while other member in the team learns how they can collaborate with the team. As a result, more amount of time will be consumed by the team to complete the project. This issue can be overcomed, if the developers work on planning and technical infrastructure before initiating the first iteration. Planning is the very first activity of first iteration. During this iteration, the team selects one specific feature that must be part of their first release. If an application includes an user-interaction, then create a story to display web page. Similarly, if it includes reporting then create a story for bare-bones report, and if it requires installation, then create a story for installer also.

These stories will provide ideas for creating more stories which will fill the missing details. The selection of stories must be made depending on the users capability to deal with it.

This reduces the customer's time in answering questions of programmers and so they can focus on creating the release plan.

The iteration planning is quite more complex than the first iteration as velocity to not established. Work on one or two stories at the same time and check the progress every day which helps in completing the stories and delivering them even if the initial help is not correct. After planning, an integration machine must be arranged for the programmers to start establishing technical infrastructure. After setting, the working on stories begins, again.

During the first iteration, the programmers works on few stories and a projector is arranged where the whole team thinks while one person operates same as pair-programming which reduces the disordering and helps in establishing initial practices like project structure, filenames and namespaces.

After a specific time period, the basic should be established well and the project should be large enough for team to work on separate parts without interfering one another. At this stage, it is good to schedule discussion on first coding standards.

When the programmers are working on the stories, the customers and testers should operate on the vision and release plan. The creation of product vision must be dealt with the stakeholders initially and then formalize it. While the vision is being finalized, brainstorm the stories and think about any other feature that could be included and select a date for the first release.

Q33. Explain about the process of applying XP to an existing project.

Answer :

Implementing XP in an Existing Project

All the XP practices can be adopted by the greenfield projects at a time. If the existing codebase is legacy project, then one must adopt XP incrementally as it may consume more time to attain the output. The biggest challenge in implementing XP to an existing project is to set enough time to pay down technical debt. The occurrence of new technical debt must be stopped to improve the productivity and to reduce the bug production. Therefore, as the technical debt decreases, velocity will increase. Setting a slack is a painful decision, so, if the collecting technical debt wont be stopped then velocity will be decreased resulting in delay of the product.

However, the product managers avoids it by resolving it smartly as follows,

Ordering

As the legacy projects have a disordered approach to planning, the first task of the XP team is to structure the project in an order. After starting with XP's structural practices, the team should be moved to an workspace and must implement pair programming for conducting iteration planning, retrospectives etc., and then apply the following practices,

(i) Apply all the "Thinking" practices

(ii) Apply all the "Collaborating" practices

(iii) Apply all the "Planning" practices

(iv) Apply version control, collective code ownership and customer reviews.

1. Management Support

It is not easy to work with XP without managements support. Product manager, onsite customers, integrated testers, common workspace with pairing stations and team members are all necessary to work with XP. Therefore, to acquire these, management play a virtual role. In addition, it is also responsible for providing the following features,

(i) It enables every member of the team to control the entire development process, database schema, builds and version control.

(ii) It supports compensation and review practices that can work successfully with the team-based effort.

(iii) It accepts new ways of demonstrating progress and showing results.

(iv) It shows patience with lowered productivity while the team is learning.

2. Team Agreement

The agreement of team members for using XP is as essential as management's support. If the team members does not prefer using XP then its certainly not going to work. But, XP assumes that the team members are willing to adopt it. However, is not a good practice to force the process on the team member who is resisting it.

3. A Collocated Team

Generally, XP depends on high-frequency and high-speed communication for most of its activities. This type of communication can be achieved when all members of the team gather at a specific place. The seating arrangement of the team is made according to the "caves and commons" approach. The primary motive of this approach is to enable team members to interact together and work individually in the same room.

4. On-site Customers

The on-site customers contribute to a large extent in the success of an XP team. The product manager is the most important among on site customers, who determines which features will be developed by the team. The decisions of on-site customers impact the value of the software. They answer the queries immediately and help the software team in knowing the feedback on-time. This facilitates both the customers and developers in improving their understanding of the software that is being developed. Domain experts and interaction designers also play vital role by helping the team in planning about upfront requirements phase and provide details about upfront more features are required by the software.

5. The Right Team Size

In XP, the teams that include only 2 to 12 programmers are considered as most-effective. This is because small teams are more flexible and can quickly adapt to changes when compared to the large teams containing 50 to 100 programmers. Moreover, the teams with less than 4 programmers might not have the required intellectual diversity. So, they may also find difficult to use pair programming. Although, large team with experienced members can handle such complexities efficiently but novice XP team may experience difficulty in the beginning.

6. Use all the Practices

XP makes use of all the resources of the project in an efficient manner. It ensures that every practice contributes directly to the development of valuable software. For instance, in XP pair programming supports collective code ownership which is necessary for refactoring. As refactoring supports incremental design and architecture and these enables the customer driver planning and frequent release thereby increasing the software value and delivering successful software.

Recommendations of XP

The recommendations for adopting extreme programming (XP) are as follows,

1. A brand-new codebase
2. Strong design skills
3. A language that's easy to refactor
4. An experienced programmer-coach
5. A friendly and cohesive team.

1. A Brand-New Codebase

In XP, it is essential to create a code that can be changed easily when required. If the code is difficult to change then it may impact on all the planning activities performed during development process. An XP teams works hard so as to keep their code simple and easy to change. If the code base is brand new then making changes becomes more easier than the existing code. Moreover, a brand new codebase facilitates the team, to deliver more number of features after every iteration with least risk.

2. Strong Design Skills

Development of simple design is one of the practices of XP that can be problematic. If the design is simple then changes can be made easily without modify in the complex algorithms and programming complex designs may increase the expenses spent on that project and also consumes more time. Therefore, it is recommended to have simple design so that programmer can plan for any future changes.

3. A Language that's Easy to Refactor

XP mostly depends on refactoring concept for improvising the existing designs consistently. So, XP supports languages tht does not make refactoring difficult. At present, object-oriented and dynamic languages with garbage collection are the easiest to refactor whereas C and C++ are most difficult languages to refactor.

Pay-down Technical Debt

The major issue in legacy projects is excessive technical debt which should be prevented. Firstly, a ten-minute build should be created along with continuous integration, then introduce a test-driven development. Meanwhile, reduce the existing technical debt by introducing additional slack into the iterations. This practice helps in improving the quality and productivity greatly over specific time period.

Organizing Backlog

The team must determine the best approach to review, clean up and categorize new entries. The customers and testers must remove the duplicates and the unimportant issues from the database.

Important Bugs Fixation

As the bug database becomes a reliable bug repository, then it may become difficult to fix decision. Then, the team must involve product manager to analyze the cost of fixing the bug. The bugs that are under don't fix must be left undone and then the remaining bugs that are fixed must be turned into stories.

Forwarding Testers

Testers have more workload on the manual regression testing. But, when the programmers focuses on the test-driven development, then an automated regression suite can be created which reduces the pressure of testers.

The programmers uses the iteration slack when there is less need to pay down technical debt which automates the remaining manual regression tests and creates end-to-end tests at beginning, then refactors end-to-end tests by focusing on unit and integration tests.

While the team generates new bugs, testers have time to work on other tasks. Then, integrate the testers with the team and start testing process.

Emerging from Darkness

The emerging from darkness is a process with allows to reduce technical debt, increase code quality and remove the defects. This increases the productivity as well. At the beginning, the progress may be low depending on the technical debt. It may take time to progress but when there is good progress then there will be more estimations and the programming will be enjoyable.

Q34. Explain in detail about the process of applying XP in a phase-based organization.

Answer :

Implementing XP in a Phase-based Organization

XP being an iterative process cannot be employed in a phase based environment easily. If any organization wants to implement XP within the existing phase-based structure, then that organization must seek permission from their higher authorities for implementing simultaneous phases of XP. If it works, then developers can embed XP into phase-based structure. The following are the various suggestions that can be utilized for implementing XP in a phase based organization,

Mandatory Planning Phase/Planning Gate

During this phase, the organization expects a detailed plan from the developers. This phase continues for about a month or four iterations. A combination of these iterations provides the customer with the complete functional product.

Analysis Phase

If an organization organizes any forward analysis phase then the requirements document may be received as "Fait accomplish" which are decomposed into stories. The starting point of each sentence in the story should include the words "must", "shall" and "should". However, the requirements documents does not serve as replacement for product manager (or) on-site customers, as they are essential to fill the missing details in the requirements document.

Design Phase

This phase is initiated only on completing the analysis of the program. At this stage, programmers often share their responsibilities. Design and architecture is created incrementally. It is mandatory to ensure simplicity of the design so as to minimize the amount of time and costs required during development process.

Coding Phase

It is the most important phase in XP. This phase is divided into one week iterations XP mainly focuses on coding phase so as it ensure it delivers a valuable product to the customers at the end of the development cycle.

Testing Phase

Testing is performed in each iteration due to which it consumes more time than coding phase. This phase is integrated with the development phase.

Deployment Phase

The developers can employ the XP's wrap up activities at the end of iteration for deploying the product at the end of cycle.

Q35. What are the extremities used for applying bits and pieces of XP?

Answer :

Extremities Used for Applying Bits and Pieces of XP

The extremities used for applying bits and pieces of XP for the existing method are as follow,

Iterations

If the interventions are recurring then the XP team, must adopt day-long iterations. During this iterations joint planning session is conducted at the beginning of each day by using planning game and measure of the work that can be tackled by the team. It is ensured that programmers measure their individual tasks. But, if the day-long iterations also does not facilitate well then go with the weekly iterations wherein standard up meetings and weekly iteration demos take place thereby enabling the team to plan and create charts for showing the future tasks.

Retrospectives

The regular retrospective meeting serves as an excellent way of improving the development process. It helps in creating transparency and trust among the team members. If the team has the authority of making further improvements, then they must schedule weekly retrospectives (or) bi-weekly retrospectives.

Ten-minute Build

A ten minute build is a practice where the developer designs the codebase to be built automatically. The quality of life can be improved with speed and automated build. It helps in reducing the amount of undone work. This reduces the time and team members can hit more targets.

Continuous Integration

The continuous integration not only minimizes the problems of integrating but also allows the team to detect the problems at early stage. Additionally, it also improvises the tests and build process.

Test-driven Development

The test-driven development is a powerful activity which forms the basis for reducing bugs, improving the ability of refactoring, increasing the speed of development and decreasing technical debt. However, as it cannot be understood easily unlike other practices, it consumes time to master.

1.4.2 Assessing Agility

Q36. Illustrate about assessing Agility with an example.

Answer :

Assessing Agility

The agile assessment is a process which is used to make investment decisions and to make alterations to the process. It focuses mainly on five important aspects of agile development which are used to explore results instead of specific practices. In addition to XP, developers can also employ quiz for assessing the underlying approach. This quiz determines the typical sources of risk. Any score with less than the minimum denotes risk and it provides opportunity to improve. However, the experienced XP team can grasp it well but if the score is shown less on any aspect, then it should be improved.

The five important aspects of the assessment quiz of agile development are as follows,

(i) Thinking

(ii) Collaborating

(iii) Releasing

(iv) Planning

(v) Developing.

The following example illustrates the concept of conducting self assessment quiz.

Example

The team members enters the scores on a photocopy of blank radar diagram. As, partial credit is not accepted, if the answer is not sure then they must give zero points.

Thinking

Question	Yes	No	XP Practices
Do programmers critiques all production code with at least one/other programmers?	5	0	Pair Programming
Do all team members consistently, thoughtfully, and rigorously apply all the practices that the team has agreed to use?	75	0	Pair Programming, Root-Cause Analysis, Retrospectives
Are team members generally focused and engaged at work?	5	0	Energized Work
Are nearly all team members aware of their progress toward meeting team goals?	4	0	Informative Workspace
Do any problems recur more than once per quarter?	0	5	Root-Cause Analysis: Retrospectives
Does the team improve its process in some way at least once per month?	5	0	Retrospectives

Collaborating

Question	Yes	No	XP Practices
Do programmers ever make guesses rather than getting answer to questions?	0	75	The XP Team
Are programmers usually able to start getting information (as opposed to sending a request and waiting for a response) as soon as they discover their need for it?	4	0	Sit Together
Do nearly all the team members trust each other?	4	0	The XP team; sit together
Do team members generally known what other team members are working on?	1	0	Stand-Up Meetings
Does the team demonstrate its progress to stakeholders at least once per month?	4	0	Iteration Demo, Reporting
Does the team provide a working installation of its software for stakeholders to try at least once per month?	1	0	Iteration Demo
Are all important stakeholders currently happy with the team's progress?	3	0	Reporting, Iteration Demo, Real Customer Involvement
Do all important stakeholders currently trust the team's ability to deliver?	3	0	Trust, Reporting

Releasing

Question	Yes	No	XP Practices
Can any programmer on the team currently build and test the software and get an unambiguous success/fail result, using a single command?	25	0	Ten Minute Build
Can any programmer on the team currently build a tested, deployable release using a single command?	5	0	Ten Minute Build
Do all team members use version control for all project-related artifacts that aren't automatically generated?	25	0	Version Control
Can any programmer build and test the software on any development workstation with nothing but a clean check-out from version control?	25	0	Version Control
When a programmer gets the latest code, is he nearly always confident that it will build successfully and pass all its tests?	5	0	Continuous Integration
Do all programmers integrate their work with the main body of code at least once per day?	4	0	Continuous Integration

Do any programmers optimize code without conducting performance test first?	0	3	Performance Optimization
Do programmers ever spend more than an hour optimizing code without customer's approval?	0	3	Performance Optimization
Are on-site customers rarely surprised by the behavior of the software at the end of an iteration?	4	0	Incremental Requirements
Is there more than one bug per month in the business logic of completed stories?	0	3	Customer Tests
Are any team members unsure about the quality of the software team is producing?	0	1	Exploratory Testing; Iteration Demo; Real Customer Involvement

The score will be identified as follows,

(i) If the score is 75 points (or) less, then it represents immediate improvement is required which to denoted in red colour.

(ii) If the score is between 75 to 96 points, then it represents as improvement necessary and its denoted in yellow colour.

(iii) If the score is 97 or 98 or 99, then it represents that improvements can be made. It is denoted in green colour.

(iv) If the score is 100, then it represents that no further improvements are required.

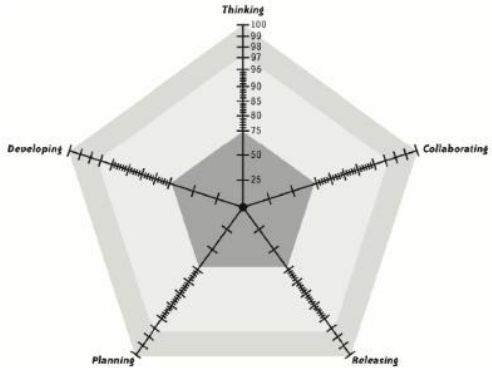


Figure: Assessment Chart

1.5 PRACTICING XP

1.5.1 Thinking

Q37. Explain in detail about the five practices that help the developers in building an efficient product.

Answer :

Thinking

The XP requires an habit of mindfulness rather than experts. Here, mindfulness refers to being completely focused on who you are with and what you are doing without distraction. This increases the frequency of developer's ability to be mindful in various situations.

In XP, there are five practices that helps the mindful developers to excel. They are as follows,

(i) Pair-Programming

For answer refer Unit-I, Q38.

(ii) Energized Work

For answer refer Unit-I, Q40.

(iii) Informative Workspace

For answer refer Unit-I, Q41.

(iv) Root-cause Analysis

For answer refer Unit-I, Q42.

(v) Retrospectives

For answer refer Unit-I, Q43.

The following are the steps to be followed for improving mindfulness,

Step 1

Initiate the process by forming heterogeneous pairs wherein a programmer collaborates with a customer, customer collaborates with tester and tester collaborates with some other developer and so on.

Step 2

The team must select any practice of their choice and timebox the discussion for about 15 minutes. This discussion may include any of the following sets of questions,

If the practice is not used,

1. What is easy to do about the practice? What is hard? What sounds ridiculous or silly?
2. How does the practice differs from the previous?
3. What must be done to use the practice exactly?

If the practice is being used,

1. What would be the aspects you do differently for the practice?
2. If you follow the practice exactly as written, what happens?
3. What experimental change could you try which gives new insights to the practices?

Step 3

Select, any three paid teams for leading the discussion such that every member gets chance to lead equally.

1.5.2 Pair Programming

Q38. Explain in detail about the concept of pair programming and why it is used.

Answer : Model Paper-II, Q3(b)

Pair-Programming

The pair-programming is a technique used in XP where two developers (or) programmers work together on a single computer for designing, coding and testing the user stories. Generally, the members in the paired team possess equal skills and spend equal time at the keyboard.

Pair-programming is the first practice which is used to increase the brain power. The pair consists of two programmers where one person writes the code called as "driver" and the other person is called "navigator" who takes the responsibility of performing the next tasks to be done. When the driver writes the code, the navigator reviews each line of code as it is being typed on keyboard. Both, the programmers keep switching their roles frequently. This arrangement makes the driver to work without any restraint on strategic challenges of creating accurate and syntactical correct code. It enables navigator to work on the strategic issues without getting distracted by the details of coding. This facilitates both driver and navigator to develop a good quality of work within short time period.

Pairing also helps in building good programming habits. The dependency of XP on continuous testing and design refining involves more self-discipline. The pair programmers spreads coding knowledge and tips through out the team. Moreover, pair programming helps the developers in getting rid of the roadblocks more easily.

The paired programmers produces the code through the conversation. As they drive or navigate they think of making the work best and takes small design steps rapidly such as test-driven development and discusses the assumptions. Short term goals, general direction and any relevant history of the features (or) project.

The pairing may occur frequently and members can be paired with anyone depending on the requirement. Sometimes, the pairing may be shifting al through the day which improves the team bonding and spreads the design skills and knowledge among the team.

Driving and Navigating

When the process of pairing is initiated the drivers may get more ideas and detects the problems more faster. But, the same things happen when driver and navigator switch their roles again. While navigating, the navigator may want to take the keyboard away form the driver. But it is essential to provide the driver with some time to rectify his is mistakes. Moreover, the navigator must be more productive, should think about the next step and must be prepared with the suggestions, and must write them on an index card rather than interrupting the driver. When there is break, then present these suggestions.

Pairing Stations

A good pairing station is crucial for the pair-programming which requires huge space for both the people to sit side by side and feel comfortable to work. The pairing station must

Does the integration build currently complete in fewer than 10 minutes?	4	0	Ten-Minute Build
Do nearly all programmers share a joint aesthetic for the code?	1	0	Coding Standards
Do programmers usually improve the code when they see opportunities, regardless of who originally wrote it?	4	0	Collective Code Ownership, Refactoring
Are fewer than five bugs per month discovered in the team's finished work?	1	0	No Bugs

Planning

Question	Yes	No	XP Practices
Do nearly all team members understand what they are building, why they are building it, and what stakeholders consider success?	25	0	Vision
Do all important stakeholders agree on what the team is building, why and what the stakeholders jointly consider success?	25	0	Vision
Does the team have a plan for achieving success?	4	0	Release Planning
Does the team regularly seek out new information and use it to improve its plan for success?	3	0	Release Planning
Does the team's plan incorporate the expertise of business people as well as programmers, and do nearly all involved agree the plan is achievable?	3	0	The Planning Game
Area nearly all the line items in the team's plan customer-centric, results-oriented, and order-independent?	4	0	Stories
Does the team compare its progress to the plan at predefined, timeboxed interval, no longer than one month apart, and revise its plan accordingly?	4	0	Iterations
Does the team make delivery commitments prior to each timeboxed interval, then nearly always deliver on those commitments?	4	0	Iterations, "Done Done", Slack, Estimating
After a line item in the plan is marked "complete", do team members later perform unexpected additional work, such as bug fixes or release polish, to finish it?	0	25	"Done Done"
Doe the team nearly always deliver on its release commitments?	3	0	Risk Management

Developing

Question	Yes	No	XP Practices
Are programmers nearly always confident that the code they' ve written recently does what they intended it to?	25	0	Test-Driven Development
Are all programmers comfortable making changes to the code?	25	0	Test-Driven Development
Do programmers have more than one debug session per week that exceeds 10 minutes?	0	3	Test-Driven Development
Do all programmers agree that the code is at least slightly better each week than it was the week before?	25	0	Refactoring; Incremental Design and Architecture
Does the team deliver customer-valued stories every iteration?	3	0	Iterations; Incremental Design and Architecture
Do unexpected design changes require difficult or costly changes to existing code?	0	3	Simple Design
Do programmers use working code to give them information about technical problems?	1	0	Spike Solution

include simple folding tables which are six feet long, so that two people can sit comfortably side-by-side and four feet deep. Each table needs high-powered development workstation (i.e., monitor, keyboard and mouse)

Advantages

Pairing is a powerful tool which reduces defects, shares knowledge, improves design quality, supports self-discipline and reduces distractions. It is done constantly and provides feedback more faster than the scheduled inspections.

Q39. What are the challenges faced in the 'pair programming'? Explain them.

Answer :

Challenges of Pair-programming

The process of pairing can be uncomfortable for the project team initially because it requires more team collaboration than before. However, the feeling of hesitation among the team members will be reduced after a month or some specific time period but the development team has to face the following challenges,

(i) Comfort

The team members paired should be comfortable with their position and equipment. They must clear their desk and ensure that two people can sit comfortably side by side. Before any issue arises, every member of the team must know how to communicate with co-workers of challenging personal habits.

(ii) Mismatched Skills

As pairing is a collaboration among the peers, sometimes a senior developer might be paired with a junior developer. Then, in such situation the paired team should maintain balance between them by taking it as an opportunity to learn from one another rather than conflicting the ideas proposed.

(iii) Communication Style

Communication skills are essential for team and team member's success. Few members in the paired team may find uncomfortable to involve their partners in communication. Ping-pong pairing makes the practice of communication easier where in when one person writes a test, another person makes it pass and while the second writes a new test and the first person makes it pass, in this way it goes on.

Therefore, every paired team should adopt an attitude of co-operative problem solving, which enables them to effectively communicate and contribute to problem solving processes.

(iv) Tools and Keybindings

Sometimes, one may find few members of the team mainly emphasizing on selection of the toolset, which might make the other co-worker angry. So, they must try to standardize on a specific toolset which is best for the project.

1.5.3 Energized Work

Q40. Explain about XP's practice of 'Energized work'.

Answer :

Energized Work

Energized work is one of the XP's practice which recognizes that the developers can provide best productive work when they are motivated and energized, when compared to the work that they provide under difficult circumstances.

The simple way to be energized are as the follows,

1. Taking care of yourself
2. Going home on time every day
3. Spending time with family and friends
4. Engaging in activities which takes the mind off the work.
5. Eating healthy food.
6. Exercising and sleeping more.

As all these things makes a person busy, then brain will provide new development ideas the next morning.

If quality time-off is the yin of energized work then the focused work is the yang. One must pay full attention to the work by disabling the interventions like email and instant messaging and also the phones should be placed in silent mode. They must ensure that they are restrained from unnecessary meetings and organizational politics by the project manager.

The energized work needs an home life along with supportive workplace. However, it is an individuals choice to be energized or not so as to remove the roadblocks.

If anyone in the team is feeling sick, they must be allowed to stay at home which prevents others from getting infected and also prevents poor work. The pair programming is another better way which encourages the energized work, by pairing the developers. The healthy food provided in the workplace is also a good way which supports the energized work.

If the number of error occurrence is more than the progress, then it is better to take a break and relax for sometime so, that the mind becomes fresh by the time again the work starts. If it is already an end of the day, going home for that day and starting freshly the next day is good idea.

1.5.4 Informative Workspace

Q41. Give a detailed explanation about informative workspace.

Answer : Model Paper-III, Q3(a)

Informative Workspace

Informative, workspace is one of the practice of XP that is developed for constructing feedback mechanisms about agile team that help them in their routine activities. These feedbacks are provided by means of visual displays. It enables people to sense the state of project by simply walking into the room. It conveys the status of team members without distributing the team members and helps in improving the trust of stakeholders. A typical informative workspace includes the following,

Subtle Cues

Information is essence of informative workspace. The feel of the room is the subtle cue that communicates useful information quickly and subconsciously.

An informative workspace also provides many other cues like white boards, stacks of index cards, a collaborative design on white board and so on. All these clues enable the team to communicate more quickly and effectively than a half-hour powerpoint presentation.

Big Visible Charts

The big visible chart is an important aspect of an information workspace which is used to display the information so clearly that it communicates across the room also. The iteration and release planning boards are the ubiquitous examples of big visible charts. The useful status chart (a large plastic perpetual calendar) is a team calendar which represents important dates, iteration numbers and contact information etc.

Hand-Drawn Charts

Hand drawn charts are hand drawn slides which are used to ensure that information is constantly visible and easily modified.

Process Improvement Charts

Process improvement charts are used for visualizing a process variable over time. These charts are use by the XP team to improve the following,

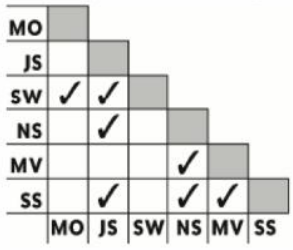
(i) To improve amount of paring by the tracking the percentage of time spent on pairing and percentage of time spent on flying individually.

(ii) To improve pair switching by tracking the number of pairing combinations made in each iteration.

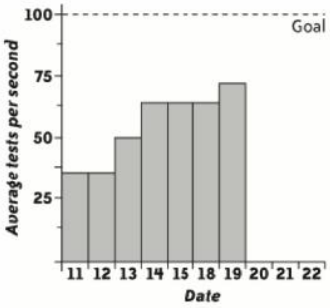
(iii) To improve build performance by tracking number of tests executed per second.

(iv) To improve support responsiveness by tracking age of oldest support request.

(v) To improve unnecessary interventions by tracking the number of hours spent on every iterations, non-story work.



(a) Pair Combinations



(b) Tests Per Second

Figure: Representation of Process Improvement Charts

Gaming

The gaming is a process which occurs when the people tries to improve the number at an expense of overall progress, i.e., when the programmer focus more on improvising the number of tests in the system, rather than maintenance. As a result, maintaining the software becomes more difficult as the system may include redundant or unnecessary test also.

1.5.5 Root Cause Analysis

Q42. What is Root cause analysis? Explain in detail how agile teams can use the 5-whys approach to identify the root cause analysis.

Answer : Model Paper-III, Q3(b)

Root-Cause Analysis

A root-cause analysis is a simple technique that is performed with the objective of determining the main reasons for the problem occurrence. This analysis is generally conducted whenever a bug is detected during development process. It can be used for analysing both major and minor problems.

If the cause of the problem is due to inefficiency of the team then it can be solved by gathering the team members and asking them for possible solutions to resolve the issue. But, if the problem cannot be handled by the team, then they may required the help of large organization to fix it.

Inorder to evaluate the why problems that occur during the production process, 5-whys approach was introduced by Sakichi Toyoda. This approach is referred as why analysis. The concept of why analysis can be understood from the following real world example.

When a project team starts working on a new task then they consume more time for bringing the code into working state, why?

Now the root-cause analysis approach of asking "why?" five times will work as the follows,

1. Why? The answer of this 'why is that the build is broken in the source control frequently.
2. Why? The answer is without running the tests, the team develops the code.
3. Why don't they run tests before checking in? The answer is – Because, the tests may consume more time to run than the specified time sometimes.

4. Why does tests take long? The answer is because testing is performed for long time period to setup and teardown.

5. Why testing consumes more time in database? The answer is it is difficult to test the business logic of design without testing the database.

Therefore, the five way approach helps you in determining what one should do for correcting the situation. it also facilitates the team in selecting tools that must be used and help them in deciding whether to automate regression testing activities or not.

1.5.6 Retrospectives

Q43. What are Retrospectives? Explain in detail about the iteration retrospective.

Answer :

Retrospectives

Retrospectives are the meetings that are conducted at the end of every iteration in software development. During this meeting the members of team determines all the actions that are necessary for improvement and move forward to the next iteration.

Types of Retrospectives

The following are the different types of retrospective used in XP. They are,

(i) Iteration retrospective

(ii) Release retrospective

(iii) Project retrospective

(iv) Surprise retrospective.

The iteration retrospective is the most common retrospective which occurs at the end of every iteration and release retrospective, project retrospectives and surprise retrospectives are conducted during occurrence of an unexpected event. These kind of retrospectives are conducted by neutral third parties i.e., experienced retrospective facilitator.

Process of Conducting an Iteration Retrospective

An iteration retrospective can be conducted by any member in the team. It is mostly preferred to start the retrospectives with an experienced neutral facilitator.

Every member in the team must get the opportunity to convey their ideas in retrospectives about the improvements that can be made to enhance the quality.

Normally, retrospectives are timeboxed between one to three hours. The team members must not feel shy to extend to extra half-hour, and the they can smoothly wind it up by moving to next step.

The following are the steps involved in process of conducting an iteration retrospective,

1. Norm kerth's prime directive
2. Brain storming
3. Mute mapping
4. Retrospective Objective.

Step 1: The Prime Directive

Retrospectives must be utilized as a platform to learn and find solutions to improve the way of working rather than blaming (or) attacking one another's skills and abilities.

It is the facilitator's job to grip the destructive behaviour. So, the team must always keep in mind, the Norm kerth's Prime directive that "Regardless of what we discover today we understand and truly believe that every member in the team has provided their best that".

Step 2: Brainstorming

Effective Brainstorming serves as a foundation to proceed further with the discussion. After agreeing to the prime directive, lay out your ideas on a white board. For instance,

(a) Enjoyable

(b) Frustrating

(c) Puzzling

(d) Same

(e) More

(f) Less

Here, the enjoyable, frustrating and puzzling are the events of thinking.

The team members can provide 'n' number of ideas without restricting them to a specific category. Let each member write his ideas on an index card. Collect all the cards and stick them on the board under their headings. Once, all the cards are gathered then evaluate these ideas constructively so as to ensure success.

Step 3 : Mute Mapping

The mute mapping is a variant way of affinity mapping where no one speaks and many ideas can be sorted quickly.

It needs more space where every member of the team go to the whiteboard and glide cards around by following three rules which are as follows,

(i) The related cards should be placed closed to one another

(ii) The unrelated cards should be placed far apart

(iii) There should be no talking.

If any two people are disagreeing to place a card, they should compromise without talking only. This activity takes around 10 minutes depending on the size of the team.

After completing mute mapping, the cards should be placed into groups on the whiteboard. Then, draw a circle around each group with a marker, which represents a category. After circling, read sampling of cards from every circle and name the category of card and ensure that you should not move cards between the categories. Finally, choose the categories that must be improved in the next iteration.

Step 4 : Retrospective Objective

When the categories are selected, then perform suitable approach on them so as to make improvements on it. Then discard the cards that are not selected. If there are many ideas, take the best suggestion and if there is no proper response, go with voting.

Therefore, the retrospectives provides the team with two advantages, they are,

(i) It helps the team to share the ideas which gives opportunity to grow more.

(ii) It helps to come up with specific solution to make necessary improvements.

(iii) In addition, retrospectives brings complete team more closer and each member learns to respect and be honest and discuss their successes and failures without any hesitation.