**What is Extreme Programming (XP)?**

Extreme programming (XP) is one of the most important software development frameworks of Agile models. It is used to improve software quality and responsiveness to customer requirements. The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.

**Good practices need to be practiced in extreme programming:** Some of the good practices that have been recognized in the extreme programming model and suggested to maximize their use are given below:

* **Code Review:** Code review detects and corrects errors efficiently. It suggests pair programming as coding and reviewing of written code carried out by a pair of programmers who switch their work between them every hour.
* **Testing:** Testing code helps to remove errors and improves its reliability. XP suggests test-driven development (TDD) to continually write and execute test cases. In the TDD approach, test cases are written even before any code is written.
* **Incremental development:**Incremental development is very good because customer feedback is gained and based on this development team comes up with new increments every few days after each iteration.
* **Simplicity:**Simplicity makes it easier to develop good-quality code as well as to test and debug it.
* **Design:**Good quality design is important to develop good quality software. So, everybody should design daily.
* **Integration testing:** It helps to identify bugs at the interfaces of different functionalities. Extreme programming suggests that the developers should achieve continuous integration by building and performing integration testing several times a day.

**Basic principles of Extreme programming:** XP is based on the frequent iteration through which the developers implement User Stories. User stories are simple and informal statements of the customer about the functionalities needed. A User Story is a conventional description by the user of a feature of the required system. It does not mention finer details such as the different scenarios that can occur. Based on User stories, the project team proposes Metaphors. Metaphors are a common vision of how the system would work. The development team may decide to build a Spike for some features. A Spike is a very simple program that is constructed to explore the suitability of a solution being proposed. It can be considered similar to a prototype. Some of the basic activities that are followed during software development by using the XP model are given below:

* **Coding:** The concept of coding which is used in the XP model is slightly different from traditional coding. Here, the coding activity includes drawing diagrams (modeling) that will be transformed into code, scripting a web-based system, and choosing among several alternative solutions.
* **Testing:** The XP model gives high importance to testing and considers it to be the primary factor in developing fault-free software.
* **Listening:** The developers need to carefully listen to the customers if they have to develop good quality software. Sometimes programmers may not have the depth knowledge of the system to be developed. So, the programmers should understand properly the functionality of the system and they have to listen to the customers.
* **Designing:** Without a proper design, a system implementation becomes too complex, and very difficult to understand the solution, thus making maintenance expensive. A good design results elimination of complex dependencies within a system. So, effective use of suitable design is emphasized.
* **Feedback:** One of the most important aspects of the XP model is to gain feedback to understand the exact customer needs. Frequent contact with the customer makes the development effective.
* **Simplicity:** The main principle of the XP model is to develop a simple system that will work efficiently in the present time, rather than trying to build something that would take time and may never be used. It focuses on some specific features that are immediately needed, rather than engaging time and effort on speculations of future requirements.
* **Pair Programming:**XP encourages pair programming where two developers work together at the same workstation. This approach helps in knowledge sharing, reduces errors, and improves code quality.
* **Continuous Integration:**In XP, developers integrate their code into a shared repository several times a day. This helps to detect and resolve integration issues early on in the development process.
* **Refactoring:**XP encourages refactoring, which is the process of restructuring existing code to make it more efficient and maintainable. Refactoring helps to keep the codebase clean, organized, and easy to understand.
* **Collective Code Ownership:** In XP, there is no individual ownership of code. Instead, the entire team is responsible for the codebase. This approach ensures that all team members have a sense of ownership and responsibility towards the code.
* **Planning Game:** XP follows a planning game, where the customer and the development team collaborate to prioritize and plan development tasks. This approach helps to ensure that the team is working on the most important features and delivers value to the customer.
* **On-site Customer:** XP requires an on-site customer who works closely with the development team throughout the project. This approach helps to ensure that the customer’s needs are understood and met, and also facilitates communication and feedback.