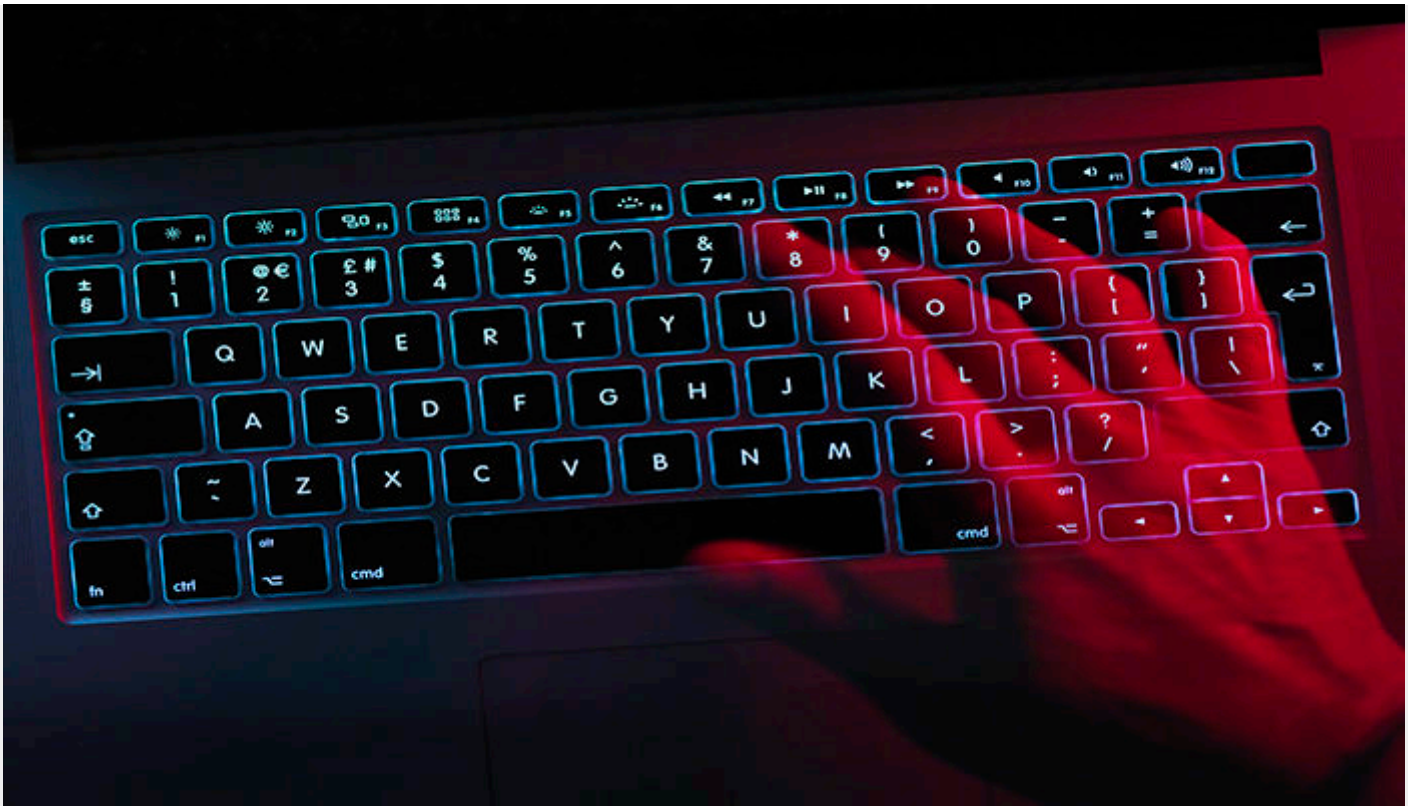
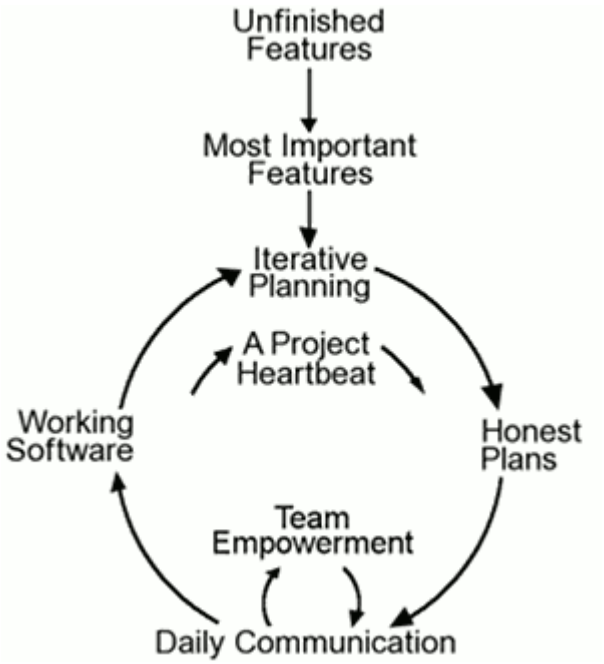
WHAT IS EXTREME PROGRAMMING (XP)?



Extreme Programming is an application of the Agile software development framework. Its goal is to make better software and improve the quality of life for the team members.



How does Extreme Programming work?

Extreme programming (XP) is an adaptive programming style designed to listen to feedback from both the customer and engineers to deliver a good, and possible, product on time. It is adaptive within the Agile methodology, particularly as XP in Agile is responsive to:

- Customer demand
- The possibilities available to your company and your industry

XP is not meant to be used in all software development projects. A particular distinction is that XP is a development framework for fixed-time projects. This is very different from something like Continuous Integration and Development (CI/CD).

Some industries have a pulse to them. Apple puts out a new phone every year, a publisher may push a book every couple months, a software team may release a new version of their software once a year. Extreme Programming has proven beneficial in these environments where you have to meet regular deadlines.

XP programming and team size

Extreme programming consists of small teams of 2 to 12 people. It is possible for an XP team's size to grow, but the root purpose—adaptability—must still be possible.

Sometimes, more team members come at the cost of adaptability.

Team roles in the XP model

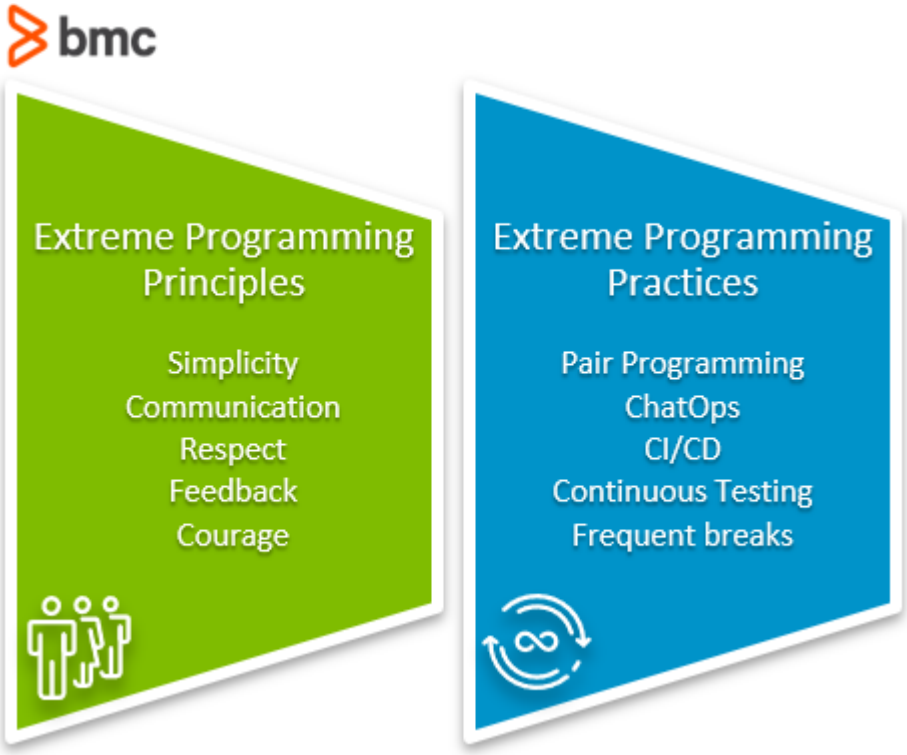
It is common to see XP teams include many different roles, not only developers. Like other agile frameworks, an XP team will consist of:

- Developers
- Product/project owners and managers
- Users

In the XP model, you can also have two new roles: The Tracker and the Coach.

- **The Tracker.** A role a developer generally fills to track particular key metrics of the team's performance, such as software quality. The role is not necessary. You'll create it on a need-only basis.
- **The Coach.** An outside party who can watch and monitor the team's performance; this person is often a consultant or a mentor.

(Learn more about Agile team roles.)



XP practices

Extreme Programming is all about communication, simplicity, feedback, courage, and respect. Common extreme programming practices are:

- Pair Programming
- ChatOps
- Continuous Integration
- Continuous Testing
- Frequent breaks! Focus is important for good programming; overworking can hurt your ability to focus.

Get started with the XP methodology

First, you need to determine if the XP methodology is right for you. XP won't work if:

1. The project is concurrent middleware development. Too many usage scenarios doesn't work with XP, and reliable unit testing is impossible.
2. The project is OS Kernel and device drivers.
3. Change is too costly or if you must closely monitor change to preserve safety.

For more specifics, refer to this article.

Second, start a new project. Projects begin with the user. What are their stories? What are their use cases? What is the solution?

- **Develop the communication lines early**. If it's done late in the project, what's the purpose of doing this methodology in the first place? Rule number one is to communicate so the team can adapt. Start clear communication early. Make it a standard for any project at the outset.
- **Pay attention to the progress of code development.** You can paint broad strokes first to get a

picture of the finished idea without coding, and then fill in the details later. (Abide by the law of diminishing returns.)
- **Test your application regularly with the user**. You do not want to stray too far from the use case. If you do, see if it can be changed to align more with the user's vision, and don't be afraid to scrap and start anew. The purpose of agile development is to avoid spending a lot of resources to, in the end, find you need to fit a square peg into a round hole. In agile development, it should be known well-ahead of time the whole is round, and to develop the peg accordingly.

Once these items are in place, XP programming should not feel like a system to strive to achieve, but a natural working order that keeps everyone on track, focused, and relaxed.

The regularity of the product release cycle inherent with Extreme Programming can increase everyone's sense of time, allowing the excitement to begin when a new project starts, and the happiness, and relief, to follow when a project comes to its deadline.

Benefits of Extreme Programming

This agile software development approach benefits your company, your customers, and your team. Specific benefits include:

- **Better quality.** Software produced with a test-driven development approach suffers fewer bugs and design deficiencies, making it easier to have clean and efficient code.
- **Flexible and adaptable processes.** Incremental changes are less disruptive and plans can adjust dynamically, based on needs.
- **Better communication and knowledge sharing.** Developers working in pairs can share knowledge, play devil's advocate, and stay in-the-know about problems and progress.
- **Higher productivity.** Clear goals and priorities, along with a clear plan, work with other extreme programming concepts to boost quality output.
- **Timely delivery.** Frequent iterations make it possible to progress faster and collect early feedback. The result is better on-time delivery of finished software.
- **Increased innovation.** In an environment that encourages trial and error, with learning from mistakes, developers are free to explore and test new ideas.
- **More cost efficiencies.** Faster production of quality code lessens the need to fix problems and do work over again. With less waste in using resources, teams are more efficient.
- **Lower risks.** When fixes are faster and cheaper, problems are found early. With regular user involvement, the chance of solutions being fit for purpose and delivered on time is much higher.
- **Better stakeholder management.** With all voices having a place at the table, diverse perspectives are heard and accommodated, improving overall buy-in.

Related Reading

- BMC DevOps Blog
- Scrum vs Kanban: A Comparison of Agile Methodologies
- The Scaled Agile Framework (SAFe): What To Know and How To Start
- DevOps Metrics & KPIs
- Asynchronous Programming: A Beginner's Guide

- Python vs Java: What's The Difference?