Extreme Programming (XP)

O XP é uma metodologia ágil, assim como o SCRUM, e seu objetivo principal é levar ao extremo um conjunto de práticas que são ditas como boas na engenharia de software.

O XP diz que:

1. Já que testar é bom, que todos testem o tempo todo;
2. Já que revisão é bom, que se revise o tempo todo;
3. Se projetar é bom, então refatorar o tempo todo;
4. Se teste de integração é bom, então que se integre o tempo todo;
5. Se simplicidade é bom, desenvolva uma solução não apenas que funcione, mas que seja a mais simples possível;
6. Se iterações curtas é bom, então mantenha-as realmente curtas;

Levando assim todas essas práticas ao extremo.

O XP possui um conjunto de princípios e valores que juntos formam a base para as práticas do processo. São eles:

Princípios

- Feedback rápido;
- Presumir simplicidade;
- Abraçar mudanças;
- Trabalho de alta qualidade;
- Pequenos passos;
- Melhoria;
- Diversidade;
- Reflexão.

Valores

- Comunicação;
- Simplicidade;
- Feedback;
- Coragem;
- Respeito.

Práticas

- **Cliente presente:** O cliente deve participar ativamente do processo de desenvolvimento. Qualquer alteração, progresso do sistema, planos futuros entre outras ações devem ser comunicadas a ele. Sempre o escute, para que saiba qual é o problema a ser resolvido.
- **Planejamento:** O desenvolvimento utilizando o XP é feito em iterações. Uma iteração é um período curto de tempo (1 ou 2 semanas) onde a equipe desenvolve um conjunto de funcionalidades. Sendo assim, no início da semana,

desenvolvedores e clientes se reúnem para priorizar as funcionalidades, reuniões que podem ser comparadas às Plannings no SCRUM. Essa reunião chama-se jogo de planejamento e nela já devem estar criadas as estórias. Se uma estória for muito grande, ela deve ser dividida em tarefas com duração máxima de alguns dias. Essas estórias devem ser escritas pelo cliente, pois assim ele consegue pensar melhor em cada funcionalidade.

- **Stand Up Meeting:** Reuniões que podem ser comparadas as Dailys no SCRUM. São reuniões feitas em pé e de curta duração para que o time se mantenha alinhado, para saber o que cada um está fazendo exatamente, em que ponto está o projeto e se alguém está tendo problemas para executar suas tarefas.
- **Programação em par:** É uma programação em par (dupla) em um único computador. Como é apenas um computador, o software sempre é revisto por duas pessoas, diminuindo assim a possibilidade de falhas. Busca-se sempre a evolução da equipe melhorando a qualidade do código fonte.
- **Testes constantes:** É utilizado o Desenvolvimento Orientado a Testes (Test Driven Development), o conhecido TDD. Primeiro crie os testes unitários e depois crie o código para que o teste funcione, essa abordagem é complexa no início, mas os testes unitários são essenciais para que a qualidade do projeto seja mantida.
- **Refatoração:** É um processo que consiste na melhoria contínua da programação por meio de revisões, sempre procurando o mínimo de introdução de erros e mantendo compatibilidade com o código já existente.
- **Código coletivo:** Diz que o código fonte não tem dono e ninguém precisa solicitar permissão para poder modificar o mesmo. O objetivo é fazer a equipe conhecer todas as partes do sistema.
- **Padronização do código:** Como todo mundo trabalha no desenvolvimento do mesmo software, a equipe de desenvolvimento precisa estabelecer regras para programar e todos devem seguir essas regras, assim parecerá que todo código fonte foi digitado pela mesma pessoa.
- **Design simples:** Essa prática se encaixa no princípio da simplicidade e é basicamente seguir o que o usuário está pedindo, por conta disso o design do software acaba sendo mais simplista. Além disso, o software acaba ficando mais fácil de ser alterado. Com essa metodologia você consegue alterar o código quando precisar, sem comprometer a qualidade.
- **Metáfora:** Procura facilitar a comunicação com o cliente, entendendo qual a realidade dele. É preciso traduzir as palavras do cliente para o significado que ele espera dentro do projeto.
- **Ritmo sustentável:** Manter um ritmo de trabalho com qualidade, onde eles estejam atentos e dispostos.
- **Semana de 40 horas:** É trabalhar com qualidade buscando ter um ritmo de trabalho saudável, 40h por semana, 8h por dia, sem horas extras.
- **Integração contínua:** Sempre que for produzida uma nova funcionalidade você nunca deve esperar uma semana para integrar com a versão atual do sistema. Isso só aumenta a possibilidade de conflitos e a possibilidade de erros no código fonte.
- **Releases curtas:** As liberações de pequenas versões funcionais do projeto auxiliam muito no processo de aceitação por parte do cliente que já pode testar uma parte do sistema. As versões chegam ainda ser menores que as produzidas em outras metodologias incrementais, como o RUP. Os releases são pequenos pedaços do

produto que são entregues ao cliente antes do tempo, assim o cliente não precisa esperar o produto todo ficar pronto para ver.