Programação extrema

Origem: Wikipédia, a enciclopédia livre.

Índice

Programação extrema

(do inglês *eXtreme Programming*), ou simplesmente **XP**, é uma metodologia ágil para equipes pequenas e médias e que irão desenvolver software com requisitos vagos e em constante mudança. Para isso, adota a estratégia de constante acompanhamento e realização de vários pequenos ajustes durante o desenvolvimento de software.

Os cinco valores fundamentais da metodologia **XP** são: comunicação, simplicidade, *feedback*, coragem e respeito. A partir desses valores, possui como princípios básicos: *feedback* rápido, presumir simplicidade, mudanças incrementais, abraçar mudanças e trabalho de qualidade.

Dentre as variáveis de controle em projetos (custo, tempo, qualidade e escopo), há um foco explícito em escopo. Para isso, recomenda-se a priorização de funcionalidades que representem maior valor possível para o negócio. Desta forma, caso seja necessário a diminuição de escopo, as funcionalidades menos valiosas serão adiadas ou canceladas.

A **XP** incentiva o controle da qualidade como variável do projeto, pois o pequeno ganho de curto prazo na produtividade, ao diminuir qualidade, não é compensado por perdas (ou até impedimentos) a médio e longo prazo.

Valores

- Comunicação
- Simplicidade
- *Feedback*
- Coragem
- Respeito

Princípios básicos

- *Feedback* rápido
- Presumir simplicidade
- Mudanças incrementais
- Abraçar mudanças
- Trabalho de alta qualidade.

Processo / Atividades metodológicas

Segundo Pressman (p.88, Eng. de Software) , o XP prefere uma abordagem orientada a objetos como paradigma de desenvolvimento e envolve 4 atividades metodológicas:

- Planejamento
- Projeto ("Designing")
- Codificação
- Testes

Práticas

Para aplicar os valores e princípios durante o desenvolvimento de *software*, **XP** propõe uma série de práticas. Há uma confiança muito grande na sinergia entre elas, os pontos fracos de cada uma são superados pelos pontos fortes de outras.

- **Jogo de Planejamento** (*Planning Game*): O desenvolvimento é feito em interações semanais. No início da semana, desenvolvedores e cliente reúnem-se para priorizar as funcionalidades. Essa reunião recebe o nome de Jogo do Planejamento e nelas já devem estar criadas antecipadamente pelos usuários as *User Stories* (história dos usuários). Nessa reunião, o cliente identifica prioridades e os desenvolvedores as estimam. O cliente é essencial neste processo e assim ele fica sabendo o que está acontecendo e o que vai acontecer no projeto. Como o escopo é reavaliado semanalmente, o projeto é regido por um contrato de escopo negociável, que difere significativamente das formas tradicionais de contratação de projetos de software. Ao final de cada semana, o cliente recebe novas funcionalidades, completamente testadas e prontas para serem postas em produção.
- **Fases pequenas** (*Small Releases*): A liberação de pequenas versões funcionais do projeto auxilia muito no processo de aceitação por parte do cliente, que já pode testar uma parte do sistema que está comprando. As versões chegam a ser ainda menores que as produzidas por outras metodologias incrementais, como o RUP.
- **Metáfora** (*Metaphor*): Procura facilitar a comunicação com o cliente, entendendo a realidade dele. O conceito de rápido para um cliente de um sistema jurídico é diferente para um programador experiente em controlar comunicação em sistemas em tempo real, como controle de tráfego aéreo. É preciso traduzir as palavras do cliente para o significado que ele espera dentro do projeto.
- **Design Simples** (*Simple Design*): Simplicidade é um princípio da XP. Projeto simples significa dizer que caso o cliente tenha pedido que na primeira versão apenas o usuário "teste" possa entrar no sistema com a senha "123" e assim ter acesso a todo o sistema, você vai fazer o código exato para que esta funcionalidade seja implementada, sem se preocupar com sistemas de autenticação e restrições de acesso. Um erro comum ao adotar essa prática é a confusão por parte dos programadores de código *simples* e código *fácil*. Nem sempre o código mais fácil de ser desenvolvido levará a solução mais simples por parte de projeto. Esse entendimento é fundamental para o bom andamento do XP. Código fácil deve ser identificado e substituído por código simples.
- **Testes de Aceitação** (*Customer Tests*): São testes construídos pelo cliente e conjunto de analistas e testadores, para aceitar um determinado requisito do sistema.
- **Semana de 40 horas** (*Sustainable Pace*): Trabalhar com qualidade, buscando ter ritmo de trabalho saudável (40 horas/semana, 8 horas/dia), sem horas extras. Horas extras são permitidas quando trouxerem produtividade para a execução do projeto. Outra prática que se verifica neste processo é a prática de trabalho energizado, onde se busca trabalho motivado sempre. Para isto o ambiente de trabalho e a motivação da equipe devem estar sempre em harmonia.
- **Propriedade Coletiva** (*Collective Ownership*): O código fonte não tem dono e ninguém precisa solicitar permissão para poder modificar o mesmo. O objetivo com isto é fazer a equipe conhecer todas as partes do sistema.
- **Programação Pareada** (*Pair Programming*): é a programação em par/dupla num único computador. Geralmente a dupla é formada por um iniciante na linguagem e outra pessoa funcionando como um instrutor. Como é apenas um computador, o novato é que fica à frente fazendo a codificação, e o instrutor acompanha ajudando a desenvolver suas habilidades. Desta forma o programa sempre é revisto por duas pessoas, evitando e diminuindo assim a possibilidade de defeitos. Com isto busca-se sempre a evolução da equipe, melhorando a qualidade do código fonte gerado.
- **Padronização do Codigo** (*Coding Standards*): A equipe de desenvolvimento precisa estabelecer regras para programar e todos devem seguir estas regras. Desta forma parecerá que todo o código fonte foi editado pela mesma pessoa, mesmo quando a equipe possui 10 ou 100 membros.
- **Desenvolvimento Orientado a Testes** (*Test Driven Development*): Primeiro crie os testes unitários (*unit tests*) e depois crie o código para que os testes funcionem. Esta abordagem é complexa no início, pois vai contra o processo de desenvolvimento de muitos anos. Só que os testes unitários são essenciais para que a qualidade do projeto seja mantida.
- **Refatoração** (*Refactoring*): É um processo que permite a melhoria continua da programação, com o mínimo de introdução de erros e mantendo a compatibilidade com o código já existente. Refatorar melhora a clareza (leitura) do código, divide-o em módulos mais coesos e de maior reaproveitamento, evitando a duplicação de código-fonte;
- **Integração Contínua** (*Continuous Integration*): Sempre que produzir uma nova funcionalidade, nunca esperar uma semana para integrar à versão atual do sistema. Isto só aumenta a possibilidade de conflitos e a possibilidade de erros no código fonte. Integrar de forma contínua permite saber o status real da programação.

Livros

- *Extreme Programming Explained: Embrace Change (2nd Edition)* (ISBN 0321278658)
- *Planning Extreme Programming* (ISBN 0201710919)
- *Extreme Programming Installed* (ISBN 0201708426)
- *Agile Software Development, Principles, Patterns, and Practices* (ISBN 0135974445)