Práticas em XP: Extreme Programming

Veja neste artigo o que é Extreme Programming, e quais são as suas principais características e a suas práticas.

O XP é um método de desenvolvimento de software criado em 1997 considerado leve, não prescritivo, e que procura fundamentar as suas práticas por um conjunto de valores. Essas práticas do XP serão vistas em detalhe posteriormente.

O objetivo principal do XP é levar ao extremo esse conjunto de práticas que são ditas como boas na engenharia de software. Entre elas podemos citar o teste, visto que procurar defeitos é perda de tempo, nós temos que constantemente testar, se perder tempo com código sujo é ruim, melhoraremos o nosso código sempre que uma nova mudança for feita. Portanto, como podemos notar todas as coisas práticas do XP são levadas ao extremo.

O XP já foi utilizado por grandes empresas como a Chrysler no seu sistema de folha de pagamento global que envolvia seus 90 mil empregados ao redor de todo o mundo. Esse projeto tem 2.000 classes e 30.000 métodos, e foi para produção sem nenhum atraso.

Outro exemplo foi o sistema Ford Motors Company VCAPS System que utilizava métodos tradicionais e enfrentava diversos problemas. Os desenvolvedores ficavam mais tempo consertando problemas encontrados do que desenvolvendo ou evoluindo o sistema. Após isso o projeto começou a adotar testes automatizados, integração contínua e outros valores do XP. Em menos de um ano, algo que não se conseguia há quatro anos, o sistema foi desenvolvido e evoluído constantemente sem maiores problemas.

O XP possui vários princípios em comum com o Scrum, que fazem parte dos princípios dos métodos ágeis gerais e compartilha da filosofia enxuta. O XP, no entanto, é mais dividido entre os profissionais com grandes adoradores e opositores.

O XP tem sucesso porque ele é um processo leve, foca em pessoas, disciplinas que agregam valor sempre com qualidade. O XP descarta muitas coisas pesadas como muita documentação, burocracia, processos pesados, etc. A comunicação é feita o tempo todo e a documentação do sistema é o próprio código. Dessa forma, para saber o que o software faz, olha-se o software, e através dos testes sabe-se o que está pronto e o que falta terminar. O XP também aumenta a coragem dos desenvolvedores com diversos princípios e práticas. Apenas gasta-se tempo naquilo que agrega valor ao cliente e no que vai realmente entrar em produção. Por isso o XP é considerado também bastante divertido.

A transição para o XP também é complicada, pois fazer simples é complicado e exige habilidade sendo um desafio para muitos. Além disso, é difícil admitir que não sabemos (quanto custa o software, qual o tempo que ele levará para estar pronto, etc). Por isso o XP trabalha com contratos mais vagos, pois ao longo do caminho saberemos realmente o preço, prazo e outras variáveis de um projeto. Também é difícil o pessoal do negócio aprender a colaborar com o desenvolvimento, pois os dois não conseguem se comunicar muito bem. No entanto, isto tem evoluindo bastante e o XP ajuda muito nisso. Apesar de tudo isso, a maior barreira do XP é cultural, ou seja, trocar a ideia de escopo congelado, contratos estáticos, empresas orientadas a documentos.

Portanto, o XP mudou radicalmente a forma de pensar o desenvolvimento de software, onde através da experiência de outros métodos como o desenvolvimento iterativo e incremental, e o espiral, agregou-se um conjunto de princípios técnicos.

O XP também tem muito de gerencial (o que é mais enfatizado no Scrum), mas está fortemente centrado em princípios técnicos. No entanto, o XP é muito mais difícil de ser implementado do que Scrum, por isso um plano transição é necessário como a adoção incremental de técnicas, ao invés de tentarmos colocar em prática todas as técnicas. O ideal é adotarmos as técnicas do XP aos poucos.

No próximo tópico falaremos sobre as práticas do XP, o tópico mais importante para entendermos o XP.

Práticas do Extreme Programming

O XP reúne um conjunto coerente de técnicas de engenharia, que agrega dinâmicas de grupo.

No XP temos um conjunto de quatro valores: comunicação, simplicidade, feedback e coragem, que a partir desses valores são geradas treze práticas: jogo do planejamento, programação em pares, pequenas versões, propriedade coletiva, metáforas, integração contínua, projeto simples, semana de 40 horas, testes, cliente presente, refatoração, padronização de código e reunião diária. As práticas reforçam os valores.

Abaixo veremos melhor o que são cada uma das práticas do XP.

Versões Pequenas: Release é algo que é implantado no cliente, ou seja, está pronto para ser utilizado. As Releases normalmente são pequenas e frequentes (a cada 2-3 meses) sendo que funcionalidades prioritárias são desenvolvidas mais cedo para serem entregues mais rapidamente ao cliente, pois prioriza-se o que ele mais precisa no momento. Obviamente que algumas organizações não querem essas entregas frequentes e pequenas, pois precisam dar treinamentos antes do sistema ser entregue e isso leva um certo tempo ou então não faz parte da cultura organizacional. Nesse caso, se a organização não seguir isso não está seguindo realmente o que mandam as metodologias ágeis como o XP.

Uma Release também pode ser guiada por funcionalidade ou prazo, nesse caso deveremos analisar o que é mais importante, liberar até um certo dia, ou liberar com determinadas funcionalidades. Por exemplo, Google Docs quando lançado tinha poucas funcionalidades, mas ao longo do tempo agregou cada vez mais funcionalidades. Neste caso preferiu-se liberar poucas funcionalidades para os usuários que quisessem logo trabalhar com esse software e que aproveitassem esse serviço que não era preciso pagar e era integrado ao e-mail.

Releases são construídas ao longo de iterações. Uma iteração sempre alcança algum objetivo perceptível ao cliente, ou seja, não adianta usarmos uma iteração para projetarmos ou melhorarmos a arquitetura do

nosso software se ele não vai agregar absolutamente nenhum valor ao cliente. Nada é feito que não seja imediatamente útil e necessário para não afetar os prazos de desenvolvimento. As iterações são divididas em tarefas que são a menor quantidade de trabalho que pode ser feita até que todos os testes voltem a funcionar. As Tarefas não levam mais que um dia, pois elas devem ser integradas imediatamente assim que o programador finaliza essa tarefa, seja ao final do dia ou antes disso, para que assim os testes sejam executados e verifique-se caso algo errado foi feito. Portanto, uma vez concluídas, tarefas são integradas imediatamente.

Jogo do Planejamento: Seu objetivo é determinar rapidamente o escopo da próxima Release, combinando prioridades do negócio e estimativas técnicas. Portanto no XP planejamos o tempo todo, diferente do que alguns pensam. Com um cliente presente ele define o escopo para a próxima Release. Dessa forma, define-se as features, prioridades e o escopo da release.

Os desenvolvedores irão definir estimativas, consequências técnicas de adotar determinadas plataformas ou componentes (por exemplo, os programadores podem dizer que determinado componente dá muito problema e seria mais adequado utilizar um componente pago que economizará muito tempo), e por fim os desenvolvedores também ajudam no detalhamento de prazos para as atividades. Por fim, mantemos o plano atualizado sempre que houver um novo “toque de realidade” no projeto como algum imprevisto ou algo desejável que não contava-se que iria acontecer, ou qualquer outra coisa que possa afetar o plano atual.

Teste: A prática de teste no XP é bastante técnica, e envolve a presença do cliente no desenvolvimento e na validação de testes. O cliente compartilha com o desenvolvedor sobre o funcionamento do sistema. Os testes também se tornam as especificações da programação, visto que o teste diz o que deve estar de acordo e o que não deve estar de acordo, é como uma especificação.Os testes são escritos antes da funcionalidade, o que também é conhecido como TDD (Test-Driven Design) onde intercala-se a função de testar um pouco e codificar um pouco. Além disso, o TDD impõe o programador à saber o que deve ser verdadeiro no programa e o que não deve ser para que ele funcione corretamente, portanto, pensa-se primeiramente no problema e depois na solução. Dessa forma, os testes são automatizados, diferente de anteriormente onde o desenvolvedor fazia a implementação e entregava para alguém testar. Com os

testes automatizados podemos executá-los a qualquer momento, e dessa forma, novas funcionalidade ou alterações podem ser imediatamente testadas para ver se essas mexidas não acarretaram outros problemas. Dessa forma, o teste impõem também confiança ao sistema e dão coragem para altera-lo, pois podemos saber imediatamente se algo introduziu um bug no sistema.

Entre os tipos de testes temos o Teste Unitário que automatiza o teste da funcionalidade e tipicamente testa uma classe, ou pequeno grupo de classes. Nesse caso usamos frameworks de testes como Junit para Java e Nunit para DotNet. Se um bug é descoberto, acrescenta-se imediatamente um caso de teste para ele. Assim garantimos que ele não se repetirá.Outro tipo de teste é o Teste de Aceitação (Teste funcional) que é especificado pelo cliente para testar que o sistema funciona conforme especificado por ele. Quando todos os seus testes de aceitação passam, a história é considerada completa.

Teste automatizado é a prática que sustenta e viabiliza muitas outras práticas como Integração contínua, Projeto Simples, Versão Pequena, Refatoração, e Propriedade Coletiva.

Programação em pares: Trata-se de duas pessoas trabalhando com UMA máquina onde um codifica, e o outro critica ou dá sugestões. Os pares trocam de lugar periodicamente. Essa prática é excelente e favorece comunicação e aprendizado. Com essa troca constante de ideias temos como resultado um Projeto de maior qualidade e como estudos comprovam temos maior produtividade e maior qualidade (com padrão de codificação e entendimento do código e partes do código que não eram entendidos). Essa prática também facilita aprendizado dos novos integrantes.

Evidente que essa prática requer ambiente de trabalho apropriado como, por exemplo, um ambiente que provê mobilidade, e pessoas que devem concordar com o barulho que será causado.

Projeto Simples: Projetos flexíveis são considerados uma defesa contra mudanças imprevistas no software, porém, projetos flexíveis também têm custos, tempo para desenvolvimento e manutenção, além de um código mais complexo e que muitas vezes nunca será utilizado.

O XP preconiza que a mudança é barata, pois ele utiliza ciclos curtos, projetos simples, refatorações, por isso mantém-se o projeto o mais simples possível, modificando-o quando for necessário suportar mais

funcionalidade. Portanto, o XP diz que o melhor e mais simples projeto é aquele que passa em todos os testes, não contém duplicação de funcionalidade, salienta as decisões de projeto importantes e tem o menor número possível de classes e métodos.

Refatoração: A refatoração significa melhorar o código sem alterar sua funcionalidade. Antes de uma mudança sempre refatoramos o código para facilitar a realização de mudanças. Ou seja, se após a refatoração o código continua funcionando como anteriormente, incluímos as novas mudanças. A refatoração contínua possibilita manter um bom projeto, apesar das mudanças frequentes. O Projeto é uma atividade diária, de responsabilidade de todos.

A refatoração provê um aumento contínuo de qualidade do código.

Propriedade Coletiva: Todos podem modificar o código a qualquer momento. Códigos não pertencem a apenas uma pessoa. A melhor forma de evitarmos problemas como trocas de pessoas da equipe e com isso a perda de conhecimento é a propriedade coletiva, onde todos mexem em todas as partes do programa e conhecem de tudo um pouco.

Integração Contínua: Todo código deve ser integrado diariamente e todos testes devem passar antes e depois da integração. Se algum problema é encontra do ele deve ser corrigido imediatamente.

Cliente presente: Clientes devem estar presentes para escrevem testes de aceitação, definirem prioridades e histórias para as futuras iterações.

Semana de 40 horas: O XP preconiza que não se pode trabalhar horas extras por mais de uma semana, pois trabalho extra é sintoma de que algo está errado. Devemos manter um ritmo sustentável.

Padrões de Codificação: Todos mexem em todos os códigos, todos refatoram e todos trabalham em pares. Assim é interessante mantermos um padrão para termos algo solidificado. Por isso a melhor forma é a equipe definir um padrão de codificação sempre no inicio dos projetos.

Metáfora: é uma linguagem comum que todos devem possuir. Por exemplo, ao invés de descrevermos como uma certa arquitetura funciona apenas comunicamos o seu nome e todos entendem o que um

programador quis dizer.

Reunião diária: é uma prática vinda do SCRUM em que todos fazem uma rápida reunião de pé para discutir o que foi feito no dia anterior, o que será feito no dia atual e se existe algum impedimento.

Neste artigo, vimos o que é o XP, quais são as suas práticas e o que são cada uma delss. Também vimos o que o XP preconiza como sendo realmente importante e fizemos algumas comparações com o SCRUM que também é bastante utilizado hoje, porém mais como um framework gerencial, diferente do XP que é mais técnico.