Academos
Revista Eletrônica da FIA
Vol. III N. 3 Jul - Dez / 2007
ISSN 1809-3604

# MÉTODO ÁGIL XP (EXTREME PROGRAMMING)

Luciano Malaquias de Souza*[*]

**RESUMO:** Como o emprego dos métodos para desenvolvimento de software tem se tornado mais popular, existe uma grande demanda, pela industria, da introdução de práticas de desenvolvimento de métodos ágeis. Esta tendência pode ser encontrada no método *Extreme Programming (XP)* como um método popular, despertando interesse tanto na área acadêmica quanto nas comunidades de programação. Apesar disso, o XP ainda não é uma realidade na grade curricular das universidades, o que, de certa forma, impede o seu crescimento e a sua aplicação no mercado. Neste artigo, será mostrado, através de pesquisa bibliográfica, como o XP pode agilizar o processo de desenvolvimento de *software*, a teoria e aplicabilidade do mesmo na comunidade dos desenvolvedores de aplicação.

**Palavras-Chave:** *Extreme Programming* XP - metodologias- referências

**ABSTRACT:** As the job of the methods for software development if it has become more popular, exists a great demand, for the company, of the introduction of practical of development of agile methods. This trend can be found in the method Extreme Programming (XP) as a popular method, claim interest in such a way in the academic area how much in the programming communities. Although this, the XP not yet is a reality in the curricular grating of the universities, what, of certain form, it hinders its growth and its application in the market. In this article, it will be shown, through bibliographical research, as the XP can speed the process of software development, the theory and applicability of the same in the community of the members of application.

**Key-Words:** *Extreme Programming* XP- methodologies- references

[*] Analista de Sistemas Sênior e Docente da FIA

## INTRODUÇÃO

A constante necessidade de se obter resultados favoráveis na economia mundial tem obrigado a indústria a reunir esforços para dinamizar o seu processo produtivo. Para isso, muito capital tem sido empregado no desenvolvimento de soluções informatizadas que dêem agilidade a seu processo de produção e, consequentemente, gerem um saldo positivo dentro do mercado. Porém, o processo de produção de desenvolvimento de software normalmente envolve riscos, como o atraso no cronograma, mudanças nos requisitos, saída de um importante membro da equipe de desenvolvimento, alta taxa de defeitos, sistemas tornando-se obsoletos, projetos cancelados, devido à ocorrência desses fatores de risco, dentre outros. Diante deste cenário, os métodos utilizados para o desenvolvimento de software vêm ganhando importância e gerando interesse tanto na comunidade de desenvolvedores quanto na área acadêmica.

Contudo, além de os métodos tradicionais de desenvolvimento de software terem o foco voltado para a documentação, necessitam de requerimentos completos e fixos, o que os torna uma metodologia pesada e não flexível. Foi contrapondo a este cenário que surgiu o *Extreme Programming* – uma metodologia ágil, que visa um rápido desenvolvimento, atende às reais necessidades do cliente e, ainda, permite modificações, à medida que novas necessidades apareçam.

Esse foco no desenvolvimento ágil que o XP traz, é de grande interesse para a indústria, podendo ocasionar inúmeros benefícios ao seu processo produtivo. Porém, atualmente faltam desenvolvedores capacitados para trabalhar com este método, gerando uma grande demanda de profissionais com este perfil. Consequentemente, a inclusão de métodos ágeis de desenvolvimento, no currículo de referência das universidades, é de grande interesse para a indústria, fornecendo-lhe mão-de-obra qualificada e contribuindo para o desenvolvimento da metodologia.

A seguir, serão apresentados os principais conceitos do XP, mostrando seus benefícios para o mercado e aplicabilidade, a inclusão do XP no currículo referencial das universidades e metodologia teórica do XP.

**EXTREME PROGRAMMING (Aplicabilidade)**

O *Extreme Programming* é um modelo de desenvolvimento de software, criado em 1996, por Kent Bech, no Departamento de Computação da montadora de carros Daimler Crysler, ele possui muitas diferenças em relação a outros modelos, podendo ser aplicado a projetos de alto risco e com requisitos dinâmicos. O XP é um conjunto bem definido de regras, que vem ganhando um grande número de adeptos e por oferecer condições para que os desenvolvedores respondam com eficiência a mudanças no projeto, mesmo nos estágios finais do ciclo de vida do processo, devido a quatro lemas adotados por seus seguidores, que correspondem a quatro dimensões a partir das quais os projetos podem ser melhorados. São eles: Comunicação, Simplicidade, FeedBack e Coragem.

No entanto, o XP não deve ser aplicado a qualquer tipo de projeto. O grupo de desenvolvedores deve formar uma equipe de 2 a 10 integrantes, que devem estar por dentro de todas as fases do desenvolvimento. É necessário realizar vários testes, às vezes, alterar o projeto em decorrência destes. A equipe tem de ser bastante interessada e pró-ativa, para assegurar a alta produtividade, e o cliente deve estar sempre disponível para tirar dúvidas e tomar decisões em relação ao projeto.

Seguindo os requisitos expostos, anteriormente, o XP poderá trazer inúmeros benefícios ao mercado, de forma que o processo de desenvolvimento se torne mais ágil e flexível. Um desses benefícios é a agilidade no Planejamento *(Planning Games)* de não definir uma especificação completa e formal dos requisitos, ao contrário das metodologias tradicionais. Outro é a produção de sistemas simples que atendam aos atuais requisitos, não tentando antecipar o futuro e permitindo atualizações freqüentes em ciclos bastante curtos.

A comunicação com o cliente, no XP, mostra-se mais intensa que nos métodos tradicionais, devendo o cliente estar sempre disponível para tirar as dúvidas, rever requisitos e atribuir prioridades, utilizando-se de sistemas de nomes, em vez de termos técnicos para facilitar a comunicação. As equipes devem manter-se integradas com os projetos, melhorando a comunicação e a produtividade. Uma outra característica importante do XP é que o código é sempre escrito em duplas, visando a melhorar a qualidade do código por um custo muito baixo, às vezes menor

do que o desenvolvimento individual. O código deve estar padronizado, para que todos na equipe possam entender o que está sendo escrito e possa ser validado durante todo o desenvolvimento, tanto pelos desenvolvedores quanto pelos clientes, a fim de se saber se os requisitos estão ou não sendo atendidos.

## EXTREME PROGRAMMING, INDÚSTRIA E UNIVERSIDADE

A inclusão de metodologias de desenvolvimento ágil nos currículos de referência das universidades é de grande valia para a indústria, que busca maior produtividade com a utilização do XP, mas que esbarra na falta de desenvolvedores qualificados para aplicar tais metodologias. A seguir, serão apresentado os conceitos do XP, descritos anteriormente, que podem ser aplicados nas universidades e, consequentemente, fornecer mão-de-obra necessária à indústria.

## EXTREME PROGRAMMING: CONCEITOS

### Valores

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

### Princípios Básicos

- *Feedback* rápido
- Simplicidade

- Mudanças incrementais
- Abraçar mudanças
- Trabalho de qualidade

**Práticas**

Para aplicar os valores e princípios durante o desenvolvimento de *software*, o **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 Planeamento** (*Planning Game*): O desenvolvimento é feito em iterações semanais. No início da semana, desenvolvedores e cliente reúnem-se para priorizar as funcionalidades. Essa reunião recebe o nome de Jogo do Planeamento. Nela, 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 projecto 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 colocadas em produção.

**Pequenas Versões** (*Small Releases*): A liberação de pequenas versões funcionais do projecto auxilia muito no processo de aceitação por parte do cliente, que já pode testar uma parte do sistema que está comprando. As versoes chegam a ser ainda menores que as produzidas por outras metodologias incrementais, como o RUP.

**Metáfora** (*Metaphor*): Procurar 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 de tempo real, como controle de tráfego aéreo. É preciso traduzir as palavras do cliente para o significado que ele espera dentro do projeto.

**Projecto Simples** (*Simple Design*): Simplicidade é um princípio do 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 projecto. Esse entendimento é fundamental para o bom andamento do XP. Código fácil deve ser identificado e substituído por código simples.

**Time Coeso** (*Whole Team*): A equipe de desenvolvimento é formada pelo cliente e pela equipe de desenvolvimento.

**Testes de Aceitação** (*Customer Tests*): São testes construídos pelo cliente em conjunto com analistas e testadores, para aceitar um determinado requisito do sistema.

**Ritmo Sustentável** (*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.

**Reuniões em pé** (*Stand-up Meeting*): Reuniões em pé para não se perder o foco nos assuntos de modo a efetuar reuniões rápidas, apenas abordando tarefas realizadas e tarefas a realizar pela equipe.

**Posse Coletiva** (*Collective Ownership*): O código fonte não tem dono e ninguém precisa ter permissão concedida para poder modificar o mesmo. O objetivo com isto é fazer a equipe conhecer todas as partes do sistema.

**Programação em Pares** (*Pair Programming*): é a programação em par/dupla num único computador. Geralmente a dupla é criada com alguém sendo iniciado na liguagem e a 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. Dessa forma o programa sempre é revisto por duas pessoas, evitando e diminuindo assim a possibilidade de erros (bugs).Com isto, procura-se sempre a evolução da equipe, melhorando a qualidade do código fonte gerado.

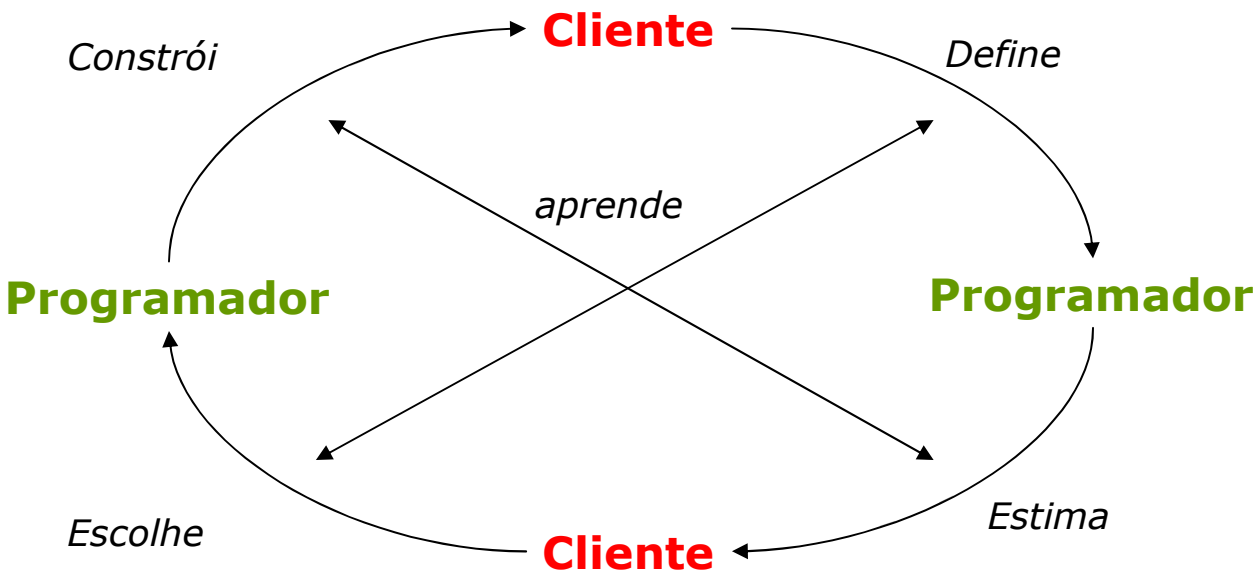
**Padrões de Codificação** (*Coding Standards*): A equipe de desenvolvimento precisa estabelecer regras para programar e todos devem seguir estas regras. Dessa forma parecerá que todo o código fonte foi editado pela mesma pessoa, mesmo quando a equipe possui 10 ou 100 membros.

**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.

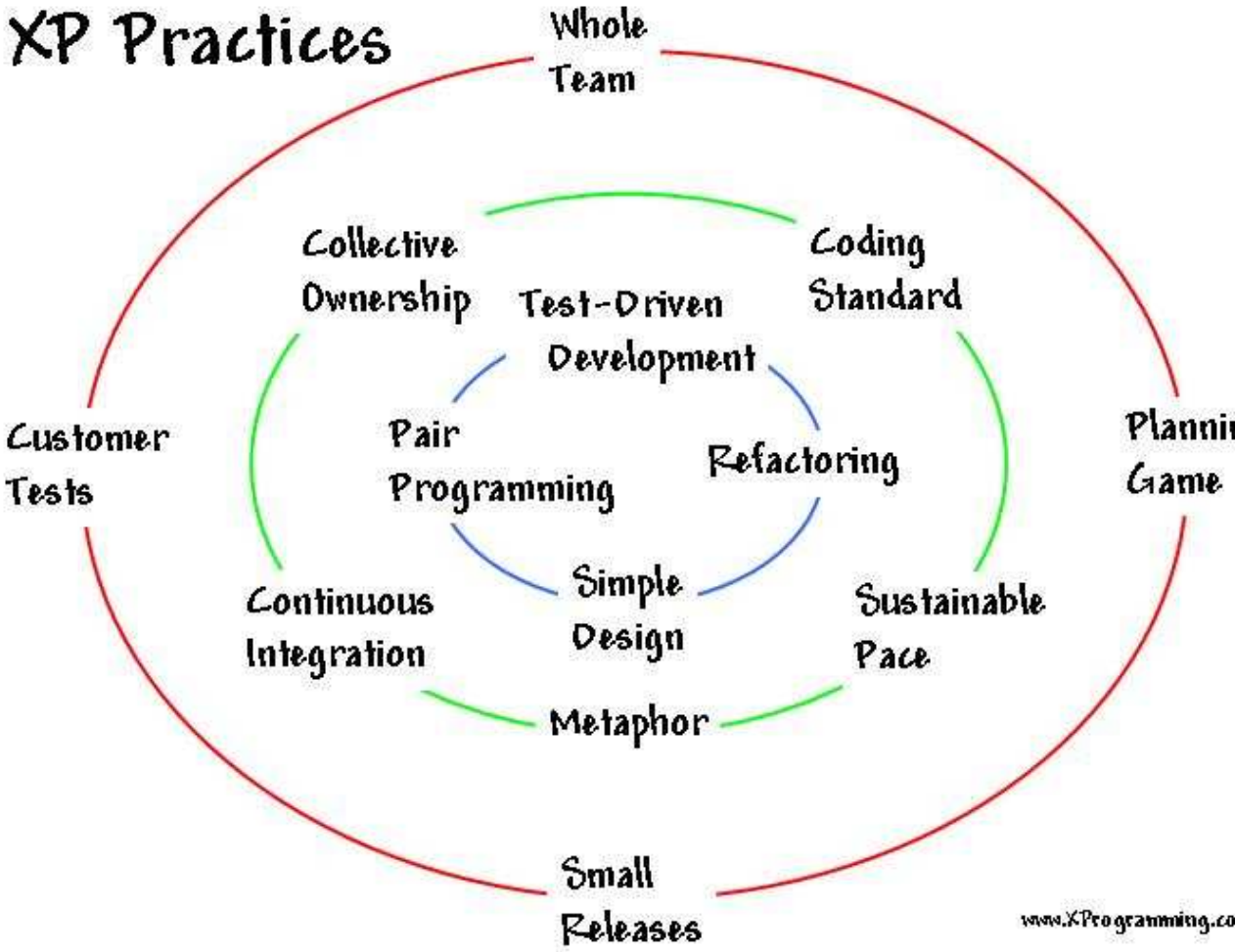
**Refatoração** (*Refactoring*): É um processo que permite a melhoria contínua da programação, com o mínimo de introdução de erros e mantendo a compatibilidade com o código já existente. Refactorizar melhora a clareza (leitura) do código, divide-o em módulos mais coesos e de maior reaproveitamento, evitando a duplicaçao de código-fonte.

**Integração Contínua** (*Continuous Integration*): Sempre que realizar uma nova funcionalidade, nunca esperar uma semana para integrar na versão actual 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.

**CICLO DE VIDA**



## PRÁTICAS




## CONCLUSÃO

A partir do que foi discutido neste artigo, e estando ciente de que este tema tratado é amplo, pode-se concluir que a inclusão do *Extreme Programming* no dia a dia do desenvolvimento de software enriquece a comunidade de programação independente do segmento das empresas nos quais os profissionais desempenham suas atividades, garantindo a evolução dos negócios e garantindo dinamismo na economia atual.

## REFERÊNCIAS

BECK, K. – Extreme Programming Explained: enbrance change. Boston : Addison Wesley /Longman, 1999.