

# **Extreme Programming: um novo conceito em Metodologia de Desenvolvimento <sup>1</sup>**

**Cassio Valente Pires <sup>2</sup>**

## **Resumo:**

A Extreme Programming (XP) é uma nova metodologia para o desenvolvimento de software e que combina rapidez e eficiência de maneira produtiva disciplinada e sobretudo humana. Este artigo se propõe a revelar como é possível desenvolver software, com requisitos incertos e que estão sempre mudando, através da Extreme Programming, abordando sobre suas práticas e valores além dos papéis adotados por seus desenvolvedores.

## **Palavras-chave**

Extreme Programming, Engenharia de Software, Simplicidade, Comunicação, Coragem, FeedBack;

# **Extreme Programming: a new concept in Development Methodology**

## **Abstract:**

The Extreme Programming (XP) is a new methodology for the software development and that agrees rapidity and efficiency of productive form, disciplined and above all human. This article proposes to reveal as it is possible to develop software, with uncertain requisites and that are always changing, through the Carries to an Extreme Programming, boarding about their practices and values besides the papers adopted by their developers.

## **Keywords:**

Extreme Programming, Software Engineering, Simplicity, Communication, Courage, FeedBack;

---

1 Trabalho realizado pela Disciplina de Análise Orientada a Objetos com UML

2 Aluno da turma de Especialização em Orientação a Objetos e Internet do Centro Universitário Anhangüera - Unigoíás

## 1. Introdução

Os desenvolvedores de software sempre buscaram formas de tornar o processo de desenvolvimento previsível, ágil e barato. Essa preocupação fez com que as indústrias de software voltassem sua atenção para a Engenharia de Software. A partir daí, foram desenvolvidas técnicas que atendessem a níveis de qualidade aceitáveis, adotando padrões e métricas que pudessem estimar o tempo e os custos do projeto para as funcionalidades exigidas.

No entanto, o percentual de projetos que conseguem alcançar suas metas, representa um parcela insignificante do total de softwares desenvolvidos. A maioria extrapola prazos, estoura orçamentos e possui funcionalidades que nunca serão utilizadas.

Diante desse quadro desastroso, foram adotadas metodologias que tornassem o desenvolvimento de software mais próximo do ideal na visão do cliente. A adoção de documentações extensas e detalhadas foi deixada de lado, o foco passou a ser o cliente, que poderá avaliar e aprender com o projeto a medida que este avance no desenvolvimento e assim possa repassar as suas reais necessidades aos projetistas . Este método evita especulações por parte dos desenvolvedores, que estarão constantemente sendo orientados por seu cliente [WUESTEFELD et al., 2004].

Foi atendendo a esses critérios que nasceu a Extreme Programming, uma novo conceito em metodologia de desenvolvimento de software.

## 2. Extreme Programming

Segundo [BECK, 2004], seu criador, a Extreme Programming é uma metodologia ágil para equipes médias e pequenas, onde os requisitos para o desenvolvimento de software são vagos e em constante mudança.

Toda a equipe de XP é reunida com o cliente diariamente a com a ajuda de cartões é possível o ajuste das suas necessidades a uma situação próxima do ideal. Estes cartões são formulários simples, escritos pelo cliente indicando o que deve ser feito em seguida, predizendo quando será feito. Dessa forma em um curto espaço de tempo o cliente poderá fazer uso do software e dizer se está de acordo com suas necessidades.

A Extreme Programming é fundamentada em valores e práticas que garantem ao cliente versatilidade e satisfação com o produto final.

### 2.1 Valores

Esses valores definem como as equipes irão proceder durante o processo de desenvolvimento. É importante que sejam observadas para se obter o melhor resultado desta metodologia.

#### 2.1.1 FeedBack

Este é o primeiro e talvez o mais importante dos valores do XP [TELES, 2004], os demais o complementam e lhe servem de suporte, mesmo porque é o FeedBack que garante um desenvolvimento ágil e consistente. O cliente está sempre acompanhando a equipe de XP e sua evolução, sendo assim sempre que aprende algo novo no sistema, tem condições de fornecer uma realimentação aos desenvolvedores que por sua vez alteram ou corrigem o que for necessário. Essa prática faz o cliente produzir e consumir o produto indefinidamente. O mesmo ocorre com a equipe de desenvolvimento que fornece soluções técnicas ou de design ao cliente à medida que produz o sistema e à medida que consome as idéias produzidas por ele.

### **2.1.2 Simplicidade**

Tudo em XP é feito da forma mais simples possível, o que não necessariamente, significa fazer de qualquer jeito. Simplicidade quer dizer que se deve produzir código simples de maneira funcional. O que também não significa digitar menos código e sim ir de encontro à funcionalidade que o cliente deseja de modo que o FeedBack ocorra.

Essa simplicidade é exigida para que não se produza trabalho extra visualizando necessidades que o cliente ainda nem descobriu. Além disso sempre que há dúvida sobre determinada funcionalidade a tendência de um desenvolvedor é produzir um código muito generalizado e que provavelmente nunca será utilizado. Trata-se então de desperdício de tempo e dinheiro do cliente que não solicitou tal implementação. Diante de uma dúvida a única certeza que se pode obter é perguntando ao cliente.

### **2.1.3 Comunicação**

Durante a fase de desenvolvimento é necessário que a equipe de desenvolvimento troque informações com seu cliente para dar continuidade ao projeto, o que é fundamental para o FeedBack. Essa comunicação em geral é feita de forma escrita como documentação. O XP ao contrário, prega que essa comunicação seja feita face-a-face.

Existem muitas formas de se comunicar seja por telefone, email, mensagem instantânea, carta ou outros meios quaisquer. Quando estamos diante de alguém percebemos gestos, as tonalidades da fala, expressões faciais e tudo o mais que enriquece uma comunicação, mas quando usamos algum dos meios descritos, essa comunicação perde essa riqueza, e pior, compromete o entendimento.

Se estamos falando queremos ser compreendidos, do contrário esperamos que o interlocutor nos informe sobre o que não foi entendido [TELES, 2004]. O que significa dizer que uma comunicação face-a-face torna o entendimento muito mais claro, prática essa que deve ser aplicada continuamente pela equipe de XP.

### **2.1.4 Coragem**

Sendo o XP um novo processo de desenvolvimento, suas premissas básicas contrariam os métodos de desenvolvimento atuais. Por isso é necessário coragem para:

*Desenvolver software de forma incremental* – Exige que os desenvolvedores reavaliar as partes já prontas para adicionar novas funcionalidades de acordo com o FeedBack do cliente.

*Manter o sistema simples* – É necessário coragem para não se deixar levar por preocupações futuras, investindo em soluções para problemas que não existem, e que caso apareçam sejam capazes de alterá-los.

*Permitir que o cliente priorize as funcionalidades* – Geralmente os desenvolvedores definem qual a ordem segundo um critério técnico de dependência dos módulos, o que não vai de encontro às necessidades do cliente. É necessário coragem para deixar para este cliente definir as prioridades.

*Fazer os desenvolvedores trabalharem em par* – Esse tipo de programação diminui o tempo de implementação e a ocorrência de erros.

*Investir tempo em refactoring* – Significa melhorar o código já existente sem alterar sua funcionalidade de modo que evolua e permita que alterações futuras sejam efetuadas mais rapidamente.

*Investir tempo em testes automatizados* – O tempo aqui não é encarado como desperdício, mas como investimento, dada a grande redução de erros no sistema.

*Estimar as histórias na presença do cliente* – É comum que gerentes de projeto temam que o cliente perceba insegurança em sua equipe, mas a XP é um processo de desenvolvimento que se baseia na honestidade e na comunicação aberta.

*Expor o código a todos os membros da equipe* – Como o código escrito por um desenvolvedor é exposto a todos os integrantes da equipe é temido que surjam críticas por parte dos demais.

*Integrar o sistema diversas vezes ao dia* – Ao integrar partes do projeto, há o risco de que algumas delas deixem de funcionar, antes que isso ocorra deve-se testá-las sistematicamente.

*Adotar um ritmo sustentável* – Para que haja qualidade no código produzido é importante que a equipe esteja sempre disposta, é necessário coragem para abandonar a crença que horas excessivas de trabalho adiantarão o projeto.

*Abrir mão de documentação que servem como defesa* – geralmente em projetos fracassados, as desconfianças tomam forma e geram atritos entre clientes e desenvolvedores, que por sua vez tomam a documentação como forma de defesa. Em XP essa prática é inadmissível pois desrespeita a premissa da honestidade.

*Propor contratos de escopo variável* – A maioria dos contratos são confeccionados de forma a impedir que o cliente faça alterações no projeto. Como XP visa o aprendizado do cliente, é necessário que os contratos sejam revistos para dar ao cliente, novas chances de fazer alterações que julgue necessárias.

*Propor a adoção de uma processo novo* – Mudanças de paradigmas são sempre difíceis de encara, como o XP rompe com uma abordagem de desenvolvimento bastante tradicional, é necessário coragem para colocar suas premissas em prática.

## 2.2 Práticas

### **2.2.1 O cliente está sempre presente**

Normalmente para o desenvolvimento de software, são levantados todos os requisitos com o clientes para confecção do projeto, depois disso o próximo contato com o cliente só acontecerá na fase de implementação meses depois.

O XP, ao contrário, propõe que o cliente esteja presente no dia-a-dia do projeto. Sua presença torna o processo de desenvolvimento muito mais simples, conduzindo o desenvolvimento com pequenos ajustes ao longo do projeto.

As funcionalidades do sistema são descritas em cartões, que recebem o nome de histórias. Quando se decide implementar uma dessas histórias o desenvolvedor dialoga com o cliente para compreendê-lo melhor.

À medida que se aperfeiçoam as funcionalidades, as dúvidas emergentes podem ser esclarecidas pelo cliente, o que garante a agilidade ao processo. Ao terminar de codificar o desenvolvedor pode demonstrá-las ao cliente que fará o FeedBack instantaneamente.

### **2.2.2 Jogo de Planejamento**

Em XP o cliente é o responsável pelas decisões de negócio enquanto que os desenvolvedores, pelas decisões técnicas. As funcionalidades do sistema são descritas pelas histórias que o cliente escreve à mão em pequenos cartões.

Quando o cliente escreve uma história, ele pensa melhor sobre o que deseja, e passa uma idéia bem resolvida do funcionalidade que deseja. Com essa prática o cliente se sente responsável por tudo o que está escrevendo nos cartões [TELES, 2004]. Como cada um gera um custo de desenvolvimento, o cliente pensa melhor antes de pedir novas funcionalidades.

Muitas vezes as histórias produzem sistemas simples que podem ser implementadas rapidamente. No entanto há histórias que podem consumir muito esforço de desenvolvimento, demorando dias ou até semanas. Para esses casos as histórias são reescritas em tarefas, as quais são divididas entre os desenvolvedores.

### **2.2.3 Stand up meeting**

São reuniões realizadas todos os dias pela equipe de XP. Essas reuniões são realizadas em até 10 minutos e as pessoas devem permanecer de pé. Dessa forma são induzidas a uma reunião rápida e objetiva.

Um dos propósitos dessa reunião é atualizar os membros da equipe sobre as últimas tarefas realizadas no dia anterior. É uma oportunidade que se tem de compartilhar problemas e soluções encontradas, disseminando conhecimento à equipe como um todo. Além disso, com a troca de informações, temos a chance de descobrir pontos fracos no sistema, aperfeiçoando-o a cada dia.

A reunião também trata do que será feito no dia que está se iniciando, e levanta as prioridades de cada um. Com todos os propósitos sendo compartilhados, temos uma noção geral de todas as fases do projeto.

Embora estar de pé não seja absolutamente necessário é importante que todos estejam concentrados e participem ativamente da reunião. É preciso que todos tomem conhecimento do que está acontecendo e principalmente que seja realizada todos os dias.

#### **2.2.4 Programação em duplas**

Ao contrário de outras metodologias, em XP a codificação não é feita individualmente, e sim em pares. Enquanto um cuida da digitação o outro cuida de acompanhá-lo ajudando na estratégia e identificando possíveis problemas. Ao identificá-los poupamos tempo com erros insignificantes. Quando modelamos um código, usamos todo o conhecimento adquirido em nossas vidas, quando modelamos com o auxílio de outra pessoa há uma soma de conhecimento e produzindo código de melhor qualidade.

Embora pareça estranho uma pessoa digitar e outra ficar parada, ocorreremos em um grande engano, pois ambas estão programando. O que acontece é que o tempo de programação cai quase pela metade quando é praticada em pares [BECK, 2004].

#### **2.2.5 Refactoring**

Trata da recodificação de códigos mal escritos, mesmo que estejam funcionando. É um risco em potencial permitir que um código assim permaneça inalterado sem saber exatamente o que ele está fazendo no sistema. Qualquer tentativa de manutenção futura estará comprometida, criando sérios problemas para quem necessitar modificá-lo ou compreendê-lo.

Ao deparar com esta situação o desenvolvedor deve reescrevê-lo de forma legível, segundo o padrão adotado pela equipe sem necessitar pedir permissão a ninguém.. Sua funcionalidade permanecerá a mesma, contudo estará mais amigável.

Em XP o código é de domínio de todos, dessa forma é um dever do desenvolvedor fazer o refactoring de qualquer parte do sistema que estiver ilegível.

#### **2.2.6 Desenvolver com auxílio de testes**

Testes são essenciais para o bom funcionamento do software, quanto mais cedo as funcionalidades são testadas menos problemas serão encontrados no futuro. A quantidade de detalhes a situações que devemos imaginar para produzir os testes é enorme, mesmo assim não devemos encará-lo como uma coisa chata e desagradável, e sim como algo intrínseco ao ato de programar [TELES, 2004]. Os testes devem ser simples e não devem consumir muito tempo.

### **2.2.7 Código coletivo**

É natural em projetos que partes do sistema sejam distribuídos pelos integrantes da equipe. No entanto é um erro permitir que esses integrantes se especializem em determinadas áreas do projeto a qual estão envolvidos. Naturalmente isso cria uma enorme dependência, sendo muito difícil substituí-las sem um grande esforço de aprendizado por parte dos outros membros. Por isso no XP cada desenvolvedor tem total liberdade de atuar em todas as partes do código, alterando o que for necessário e praticando o refactoring.

### **2.2.8 Padrão de desenvolvimento**

Para melhorar a comunicação através do código e torná-lo legível é importante que se adotem determinados padrões. Detalhes simples como a forma de indentação podem fazer grande diferença. Deve-se procurar evitar muitos comentários, e dentro da racionalidade podem-se usar nomes longos que comuniquem a idéia que se quer expressar.

### **2.2.9 Design Simples**

O XP trabalha para que o desenvolvimento gere valor para o cliente. Sendo assim o design deve ser o mais simples possível de modo que atenda a todas as necessidades do sistema. O XP parte do princípio que as futuras alterações crescem lentamente e tendem a se estabilizar [BECK, 2004]. Dessa forma pode-se adiar as grandes tomadas de decisão tanto quanto possível aumentando as chances de acertá-las.

### **2.2.10 Metáforas**

Muitas vezes quando queremos explicar determinado assunto para alguém, sentimos uma determinada dificuldade em fazer com que o interlocutor nos compreenda, então quando comparamos o assunto com outro de compreensão ampla, temos um resultado muito mais positivo. Por serem tão eficientes as metáforas passaram a ser utilizadas pelo XP.

### **2.2.11 Ritmo Sustentável**

Sistemas importantes sempre consomem muito tempo para serem concluídos e nem sempre o cronograma acompanha a velocidade dos negócios. A consequência disto está no excesso de horas trabalhadas, fins de semana e feriados que são sacrificados incondicionalmente. O trabalho de desenvolvimento exige concentração e criatividade, o que não é algo fácil de se encontrar em pessoas fatigadas.

O XP faz uma proposta diferente, em respeito à individualidade de cada um e considerando os limites humanos, ela propõe jornadas diárias de 8 horas ou 40 horas semanais. Ao contrário do que a maioria dos gerentes de projeto acreditam, essa prática aumenta a agilidade de desenvolvimento do projeto [BECK, 2004].

### **2.2.12 Integração Contínua**

Tradicionalmente para o desenvolvimento de software, dividimos o sistema em partes estratégicas e as distribuimos entre os programadores. Definimos interfaces convencionais para que essas partes se comuniquem e formem um sistema como um todo. Mas o que se tem observado são inúmeros problemas de integração, os mais frequentes decorrem da falta de compreensão da interface alheia e por ignorar padrões pré-estabelecidos para as interfaces.

De outra forma, o XP propõe que as partes do sistema seja, integradas e testadas diversas vezes ao dia, o que é conhecido como integração contínua. O tempo exigido para o build deve ser pequeno e para isso precisamos diminuir a interdependência entre arquivos e compilar somente códigos editados.

### **2.2.13 Releases curtos**

O processo comum de desenvolvimento de software é muito demorado, da análise à implantação são necessários meses e até anos, correndo ainda o risco de nunca ser implantado.

O XP trabalha com a noção de releases curtos. Releases são um conjunto de funcionalidades que geram valor ao cliente. Devem ser tão curtas quanto possível, assim o cliente obterá o máximo de valor com o mínimo de esforço, implementando novas releases sempre que surgirem.

### **2.2.14 A organização do ambiente de trabalho**

Para melhorar a comunicação e a programação em par, a XP recomenda que todos os membros da equipe possam trabalhar próximos uns dos outros, de modo que se possa escutar qualquer assunto relativo ao desenvolvimento do projeto. Mesas são preferíveis a baias, devem ser grandes o suficiente para que duas pessoas possam se acomodar confortavelmente para exercer a programação em par.

## **2.3 A Equipe XP**

### **2.3.1 Gerente de Projeto**

Trata das negociações e acordos com o cliente, repassando a ele como está o andamento do projeto, além disso deve agendar as visitas que o cliente irá fazer e insistir sempre que compareça. É responsável por todos os aspectos técnicos necessários ao andamento do projeto. Deve evitar que a equipe se envolva com os aspectos burocráticos e provê-la de todo o equipamento e software necessário.

### **2.3.2 Coach**

É o supervisor da equipe, ele garante que os valores e premissas da XP sejam respeitadas e postas em prática. Deve conhecer profundamente as práticas de desenvolvimento e orientar a equipe, alertando para eventuais falhas no processo. O ideal é que esteja presente diariamente com a equipe.

### **2.3.3 Analista de teste**

Garante a qualidade do sistema com a aplicação de testes. Auxilia o cliente a elaborar os testes de aceitação e fazer com que o sistema processe todos corretamente. Os testes devem ser aplicados ao final de cada iteração, para garantir a estabilidade e segurança do sistema. E depois da instalação da última release o próprio cliente pode seguir os roteiros e continuar a validação.

### **2.3.4 Redator Técnico**

É especialmente importante ter uma pessoa dedicada a tratar da documentação do sistema, de forma que esteja sempre atualizada, permitindo aos desenvolvedores que se dediquem essencialmente ao trabalho de codificação. Nada impede que o analista de testes desempenhe essa função, o importante é que esteja em sintonia com os desenvolvedores e com o cliente.

### **2.3.5 Desenvolvedor**

É aquele que analisa, projeta e codifica o programa sistema. Na verdade é ele quem o constrói. A XP não trabalha com hierarquias, cada desenvolvedor exerce o papel de design, programador e analista em diversos pontos do sistema, o que é garantido pela programação em par.

## **3. Conclusão**

A Extreme Programming devido a suas técnicas tem atraído cada vez mais desenvolvedores inclusive grandes empresas como Ford, Borland, BMW e Symentec entre outras. Sua metodologia aproxima muito mais o cliente da equipe de desenvolvimento garantindo uma integração absoluta na construção do software. Até mesmo pela programação em duplas a XP consegue disseminar conhecimento e garante uma inter-

relação muito maior na equipe. Por todos os seu benefícios que humanizam a produção de software a XP sai com vantagem sobre as demais metodologias.

#### **4 Referências Bibliográficas**

[TELES, 2004] TELES, Vinícius Manhães. “*Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade*”. São Paulo – SP: Novatec Editora, 2004.

[BECK, 2004] BECK, Kent. “*Programação extrema (XP) explicada: acolha as mudanças*”. Porto Alegre – RS: Bookman, 2004.

[AMBROSI, 2004] AMBROSI, Cleison Vander; GRAHL, Everaldo Artur. “*Extreme Programming: Guia Prático*”. Rio de Janeiro – RJ: Campus, 2002

[POHREN, 2004] POHREN, Daniel. “*XP Manager: Uma Ferramenta de Gerência de Projetos Baseados em Extreme Programmin*”. Novo Hamburgo – RS: Centro Universitário Feevale, 2004.

[WUESTEFELD et al., 2004] WUESTEFELD, Klaus. Xispê: Extreme Programming. Disponível em <http://www.xispe.com.br/index.html>. Acessado em: 23/11/2004