Apresentando XP. Encante seus clientes com Extreme Programming

Giovane Roslindo Kuhn

Vitor Fernando Pamplona

Resumo: Este artigo tem por objetivo apresentar a metodologia de desenvolvimento ágil de *software* denominada *Extreme Programming*. Serão abordadas, de forma resumida, as práticas, valores, e os papéis disponíveis a cada integrante de uma equipe de XP. Alguns comparativos com outras metodologias são feitas ao decorrer do trabalho enaltecendo as propriedades que definem a *Extreme Programming*.

Palavras-chave: *Extreme Programming* , Engenharia de Software, Qualidade de Software, Metodologia de Desenvolvimento, Processos Ágeis de Desenvolvimento, XP.

Índice

1. Introdução

A muito tempo a indústria de software vem passando por grandes transformações e novos desafios, entre eles desenvolver *softwares* com qualidade, no menor tempo possível e que atendam as necessidades dos clientes.

Com estes novos desafios a indústria de software passou a dar valor a algumas áreas da informática, como a engenharia de software e qualidade de software, com intuito de atender as exigências do mercado.

A indústria começou a utilizar metodologias de desenvolvimento de software, adotou métricas e padrões para alcançar níveis aceitáveis de qualidade, prever custos e prazos em seus projetos. Porém ainda são poucos os projetos que conseguem obter pleno sucesso em seu desenvolvimento, onde prazo e orçamento estabelecidos e as necessidades do cliente sejam realmente atendidas.

Pesquisa feitas pelo *Standish Group International* em 1994, um pouco antes da adoção de metodologia de desenvolvimento pelas indústrias, apontam que apenas 16, 2 % dos projetos de software atingiam sucesso (prazo, orçamento e funcionalidades atendidas). Em 2002 esta taxa havia subido para 34 %, ou seja, um aumento de 100 % em 8 anos. Mas estas taxas ainda se encontram muito aquém do esperado pelo mercado.

As metodologias utilizadas nos projetos pesquisados eram as mais variadas, podemos citar modelo em cascata, modelo iterativo e alguns com modelo em prototipação. Neste trabalho será utilizado o termo *desenvolvimento tradicional* para os projetos que se utilizem do modelo em cascata e todos os outros que se baseiam nele.

Analisando os motivos para a baixa taxa de sucesso dos projetos desenvolvidos com modelos tradicionais, cita-se os principais motivos e bastante comuns entre eles:

- Tempo elevado entre cada fase do projeto, não acompanhando as mudanças de requisitos do projeto;
- Falta de conhecimento por parte do cliente da sua real necessidade, dando margem às especulações dos desenvolvedores;
- Forte linearidade no desenvolvimento do projeto;

Focando nas fragilidades do modelo tradicional, surgiram metodologias para desenvolvimento ágil de software. Cita-se algumas características destas metodologias:

- Foco nas pessoas, não no processo, evitando especulações dos desenvolvedores;
- Atender as reais necessidades do cliente, na velocidade e praticidade por ele desejada;
- Ausência de linearidade no desenvolvimento do projeto;
- O cliente aprender a suas reais necessidades durante o projeto e repassar esta novas necessidades a equipe de desenvolvimento;

Uma destas metodologias de desenvolvimento ágil é o *Extreme Programming* , metodologia que prima a qualidade do software desenvolvido, que atenda as reais necessidades do cliente e seja entregue dentro do prazo definido. Esta metodologia será detalhada a seguir.

2. Extreme Programming (XP)

XP é uma metodologia para desenvolvimento de software ágil, com qualidade e que atenda as necessidades do cliente. Alguns praticantes definem a XP como a prática e a perseguição da mais clara simplicidade, aplicado ao desenvolvimento de software.

Uma metodologia voltada para projetos cujos requisitos mudem com freqüência, utilizem desenvolvimento orientado a objetos, equipes de até 12 desenvolvedores e desenvolvimento incremental.

A XP Busca o máximo de valor a cada dia de trabalho da equipe para o seu cliente. Em um curto espaço de tempo o cliente terá um produto que possa ser utilizado, podendo aprender com o mesmo e reavaliar se o que foi desenvolvido é realmente o desejado.

Por ser uma metodologia recente, a XP sofre mudanças em suas concepções e, portanto, é comum encontrar variações. A adaptação ao ambiente de desenvolvimento deve ser levada em conta, se um valor trouxer mais prejuízos do que benefícios é necessário relavaliar a utilização desta metodologia.

A XP é organizada em torno de um conjunto de práticas e valores que atuam perfeitamente para assegurar um alto retorno do investimento efetuado pelo cliente. A seguir serão apresentados os valores e em seguida as práticas.

2.1 Valores da XP

Os valores são as diretrizes da XP. Eles definirão as atitudes das equipes e as principais prioridades da metodologia.

Para uma empresa estar realmente utilizando o XP, ela deve respeitar e utilizar todos os valores e práticas listadas nos próximos capítulos e caso um destes valores ou práticas não seja utilizado pela empresa, esta empresa não está trabalhando com a metodologia XP.

2.1.1 Feedback

O cliente aprende com o sistema que utiliza e com este aprendizado consegue reavaliar o produto recebido, com isso pode realimentar a equipe de desenvolvimento com as suas reais necessidades. Com o *feedback* , o cliente conduz o desenvolvimento do seu produto, estabelece prioridades e informa aquilo que é realmente importante.

Analogamente, há o *feedback* dado pelo desenvolvedor ao cliente, onde o mesmo aponta riscos, estimativas e alternativas de *design* . Este retorno é o aprendizado do desenvolvedor sobre o que o cliente deseja.

Com este valor, o tempo de defasagem entre as fases do projeto são extremamente pequenos, o cliente está o tempo todo em contato com a equipe de desenvolvimento, muito diferente dos modelos tradicionais, onde o

cliente entrava em contato com a equipe um bom tempo depois do último *feedback* dado.

Um ponto que muitas empresas de *software* falham é não dar valor ao que o cliente realmente deseja, utilizam cegamente metodologias e acabam esquecendo o real propósito de um software: facilitar o trabalho de pessoas.

Com isto muitos sistemas acabam dificultando e burocratizando as tarefas das pessoas e como defesa as empresas alegam ter um produto genérico e que atenda as normais legais.

2.1.2 Comunicação

Para que o *feedback* entre cliente e desenvolvedor possa ser efetuado com sucesso é necessário ter uma boa comunicação entre eles. A XP prega que esta comunicação ocorra da forma mais direta e eficaz possível, oferecendo agilidade aos assuntos tratados. Recomenda-se o contato direto (face-a-face) entre cliente e desenvolvedor, para evitar qualquer tipo de especulação ou mal entendido entre as partes e para que possíveis dúvidas possam ser resolvidas de imediato.

Além de sanar as dúvidas no desenvolvimento, o cliente deverá estar disponível para a equipe, ou mesmo presente no ambiente de trabalho da empresa. Isto fará com que o cliente entenda o sistema e enriquecerá os relacionamentos pessoais, criando um elo de parceria e confiança mútua.

Algumas equipes não se adaptam bem a este valor. Este problema deve ser trabalhado em conjunto com a equipe. Enquanto não se acostumarem a falar e a trocar idéias com seus companheiros o sucesso da metodologia estará comprometido. Membros introvertidos são deseconselháveis para equipes de XP.

2.1.3 Simplicidade

Para que o cliente possa aprender durante o projeto e consiga dar o *feedback* necessário à equipe, não basta apenas uma boa comunicação, é necessário que os desenvolvedores implementem da forma mais simples possível o que o cliente deseja.

A lei é: faça a coisa mais simples que pode funcionar. Com esta filosofia, o cliente terá a funcionalidade rapidamente e da forma desejada, dando um *feedback* instantaneamente evitando especulações. O desenvolvedor deve implementar apenas o necessário para que o cliente tenha seu pedido atendido.

Ser simples não é um ato de desespero, é um ato de consciência e absoluta precisão. Muitas pessoas confundem simplicidade e facilidade. O mais simples nem sempre é o mais fácil e também não é escrever menos código. Simplicidade significa codificar o necessário para que um requisito seja atendido e entregue ao cliente.

Evita-se suposições, o futuro é incerto e por causa disso não necessita atenção. Os requisitos evoluem gradativamente em conjunto com o sistema e a arquitetura do projeto. Algumas vezes, o que é necessário hoje será descartado amanhã, e outras vezes o que seria necessário num futuro próximo nunca será utilizado.

2.1.4 Coragem

Por ser um processo de desenvolvimento novo e baseado em diversas premissas que contrariam o modelo tradicional, o XP exige que os desenvolvedores tenham coragem para:

- Desenvolver software de forma incremental;
- Manter o sistema simples;
- Permitir que o cliente defina prioridades;
- Fazer desenvolvedores trabalharem em pares:
- Investir tempo em *refactoring* ;
- Investir tempo em testes automatizados;
- Estimar estórias na presença do cliente;
- Expor código a todos os membros da equipe;
- Integrar o sistema diversas vezes ao dia;
- Adotar ritmo sustentável de desenvolvimento;
- Abrir mão de documentos que servem como defesa;
- Propor contratos de escopo variável;
- Propor a adoção de um processo novo.
- Assumir em relação ao cliente possíveis atrasos e problemas de implementação;
- Colocar desenvolvedores e clientes frente a frente;
- Implantar uma nova versão do sistema no cliente semanalmente;
- Apostar em seus colaboradores aumentando suas responsabilidades;
- Modelar e documentar apenas quando for de extrema necessidade.

2.2 Praticas da XP

Como o nome já diz, as práticas são um conjunto de atividades que deverão ser seguidas pelas equipes que desejam utilizar a XP.

Os valores já apresentados somados a estas práticas são um conjunto ? entrelaçado ? de boas atitudes. A fraquesa de umas é compensado por outra e assim fecha-se um ciclo fortemente dependente.

2.2.1 Cliente disponível ou presente

O XP trabalha com uma visão diferente do modelo tradicional em relação ao cliente. O XP sugere que o cliente esteja no dia-a-dia do projeto, acompanhando os passos dos desenvolvedores, onde a sua ausência representa sérios riscos ao projeto.

As funcionalidades do sistema são descritas brevemente em estórias em conjunto com os testes conceituais e serão estes os indicadores para uma boa implementação. No momento que os desenvolvedores irão implementar as estórias nada mais eficaz do que dialogar com o cliente para entender a estória, fazendo-se necessária a presença do cliente no ambiente de desenvolvimento.

Ao terminar uma estória, com a presença do cliente, a mesma poderá ser validada rapidamente e a equipe receber o *feedback* necessário sobre a funcionalidade, criando ciclos rápidos e precisos.

2.2.2 Jogo de planejamento

A XP utiliza o planejamento para assegurar que a equipe de desenvolvimento esteja trabalhando naquilo que gere o máximo de valor para o cliente. Este planejamento é feito várias vezes durante o projeto. é o momento onde o cliente solicita as funcionalidades desejadas através de estórias, onde a equipe estima o custo de cada estória e planeja as *releases* e as iterações.

Todas as funcionalidades do sistema são descritas em estórias, pequenos cartões em que o cliente deve descrever o que deseja com suas palavras e da forma mais simples possível. Lembrando que a simplicidade também deve ser respeitada pelo cliente.

Após a definição das estórias é necessário estimar o tempo das mesmas para que o cliente priorize o que deve ser implementado. A XP utiliza uma unidade chamada *ponto* , que refere-se a um *dia de trabalho ideal* do desenvolvedor, onde o mesmo não precisaria atender telefonemas, participar de reuniões, ou seja, estaria preocupado apenas com a estória em questão.

Muitas vezes algumas estórias consomem semanas de trabalho, oferecendo uma certa dificuldade de serem estimadas. A XP recomenda que estas estórias sejam quebradas em tarefas menores e que as mesmas não utilizem mais que alguns pontos de um desenvolvedor, recomenda-se 4 pontos ao máximo.

Em cada estória é escrita a quantidade de *pontos* estimadas pelo desenvolvedor, o XP recomenda que as estimativas sejam efetuadas em equipe e se possível com a presença do cliente para que durante a estimativa eventuais dúvidas sejam sanadas.

O XP tem por objetivo gerar valor para o cliente ao longo do projeto, para isto o software é desenvolvido de forma incremental, onde a cada entrega o cliente possa utilizar o sistema e obter benefícios do mesmo. Estas entregas o XP denomina como sendo *releases* , pequenos intervalos de tempo, máximo 2 meses, onde funcionalidades que gerem valor ao cliente sejam entregues.

A divisão dos *releases* é efetuada no início do projeto, geralmente com tamanhos fixos e pequenos. Após esta divisão o cliente define as estórias que farão parte dos *releases* e tenta-se evitar um planejamento muito longo, pois na entrega de cada *release* o cliente aprenderá com o sistema e possivelmente irá alterar as estórias para o próximo *release* .

Mesmo os *releases* sendo efetuados em curto espaço de tempo, continua representando um tempo muito longo para o *feedback* do cliente. Por esta razão os *releases* são divididos em espaços menores, chamados de *iterações* .

Uma iteração contêm um conjunto de estórias a serem implementadas, com duração entre uma a três semanas, onde ao final da mesma o cliente possa validar as implementações efetuadas e fornecer o *feedback* necessário à equipe.

2.2.3 Stand up meeting

O dia de trabalho de uma equipe XP sempre começa com um *stand up meeting* . é uma reunião rápida pela manhã, com aproximadamente 20 minutos de duração e onde seus integrantes permaneçam preferencialmente em pé.

Segundo estudos uma reunião em pé é mais rápida, evita bate-papos paralelos e faz os integrantes irem diretamente ao assunto. Mais uma vez, ágil e simples.

A reunião tem por objetivo atualizar a equipe sobre o que foi implementado no dia anterior e trocar experiências das dificuldades enfrentadas. Neste momento também são decididas as estórias que serão implementadas no dia e em conjunto definir os responsáveis por cada uma delas.

2.2.4 Programação em par

O XP **exige** que todo e qualquer código implementado no projeto seja efetuado em dupla, chamada de programação em par. é uma técnica onde dois desenvolvedores trabalham no mesmo problema, ao mesmo tempo e no mesmo computador. Um deles é responsável pela digitação (condutor) e outro acompanhando o trabalho do parceiro (navegador).

Esta prática agrega diversas técnicas de programação, enquanto o condutor codifica o problema o navegador permanentemente revisa o código digitado, onde pequenos erros de programação que demoraria algumas horas para serem depurados, facilmente são percebidos pelo navegador da dupla.

Um dos grandes benefícios da programação em par é a troca de experiências e idéias entre os desenvolvedores. A solução para um problema geralmente é a soma das idéias da dupla, tornando uma solução e codificação muito mais simples e eficaz.

é com esta prática que o XP uniformiza o nível dos desenvolvedores da equipe, através da troca de experiências. Após alguns meses trabalhando em duplas todos os desenvolvedores conhecerão todas rotinas do sistema, prontos para qualquer modificação ou para auxiliar algum iniciante.

Um grande questionamento sobre esta prática é com questão a produtividade dos desenvolvedores. Porém, é um erro pensar que somente uma pessoa estará codificando enquanto o outro apenas observa. O membro que não está codificando não apenas observa, mas também troca idéias, gera soluções e evita praticamente todos erros de codificação além de cobrar padrões de desenvolvimento da equipe.

Estudos indicam que a produtividade de uma equipe que utiliza *pair programming* e de equipes que tenham desenvolvedores sozinhos é praticamente a mesma, porém a qualidade do código gera facilidade de manutenção e outros ganhos a médio e longo prazo.

2.2.5 Refactoring

Um desenvolvedor ao deparar com um código mal escrito ou pouco legível mas que esteja funcionando, nos modelos tradicionais de desenvolvimento, dificilmente efetuaria alterações neste código, mesmo que tivesse que implementar novas funcionalidades.

O XP prega que todo desenvolvedor ao encontrar um código duplicado, pouco legível, mal codificado, sem

padronização, lento, com código legado ou uso incorreto de outras implementações, tem por obrigação alterar este código deixando-o mais legível e simples, porém esta alteração não pode mudar o comportamento do código em questão.

Esta prática anda de mãos dadas com o código coletivo, já que todo desenvolvedor tem a possibilidade de melhorar qualquer código do sistema.

A padronização oferece facilidades aos desenvolvedores no momento de implementar novas funcionalidades ou efetuar qualquer tipo de manutenção, uma vez que o código se encontra simples e claro.

Uma questão importante é que a prática de *refactoring* esta apoiada pelos testes automatizados, pois facilmente o desenvolvedor terá um *feedback* se a alteração por ele efetuada irá gerar qualquer tipo de comportamento anormal no sistema, sofrendo o aprendizado sobre a alteração por ele efetuada.

2.2.6 Desenvolvimento guiado por testes

Esta atividade é considerada extremamente chata e dispendiosa por muitos desenvolvedores na modelagem tradicional, porém para os desenvolvedores de uma equipe XP esta atividade deve ser encarada com extrema naturalidade. Todo código implementando deve coexistir com um teste automatizado, assim garantindo o pleno funcionamento da nova funcionalidade.

é com base nestes testes automatizados que qualquer desenvolvedor terá coragem para alterar uma parte do código que não tenha sido implementada por ele, já que executando os testes automatizados poderá verificar instantaneamente se a modificação efetuada alterou o comportamento do sistema.

Com a implementação de testes o desenvolvedor poderá amadurecer o entendimento sobre o problema que irá solucionar, já que os testes são codificados antes da nova implementação.

No XP existem dois tipos de testes, os testes de unidade e de aceitação. O teste de unidade tem por objetivo verificar se os resultados gerados por cada classe estão corretos, já o teste de aceitação tem por objetivo verificar se a interação entre as classes que implementam uma funcionalidade (estória) atendem a necessidade de forma correta. Os testes de aceitação são descritos pelo cliente e implementados pelos desenvolvedores, facilitando ainda mais a interação entre as partes.

2.2.7 Código coletivo

No modelo tradicional de desenvolvimento, é comum dividir o projeto em partes e colocar responsáveis por cada uma destas partes. Porém apenas uma pessoa torna-se conhecedora daquela parte.

Este tipo de divisão traz sérios problemas ao projeto, uma vez que se aquela parte necessitar inúmeras alterações, apenas uma pessoa estará capacitada para alterá-la. Com estas inúmeras alterações, esta pessoa pode se tornar um gargalo no projeto.

O XP trava uma batalha contra este tipo de divisão, já que não existe uma pessoa responsável por uma parte do código. Cada desenvolvedor tem acesso a qualquer parte do sistema e tem liberdade para alterá-la a qualquer momento e sem qualquer tipo de aviso.

Esta prática tem como conseqüência um código revisado por diversas pessoas e caso algo não esteja claro, com

certeza será alterado por alguma pessoa (*refactoring*) para que o mesmo torne-se legível.

2.2.8 Padrões de desenvolvimento

Um dos valores do XP é a comunicação, e o código também é uma forma da equipe se comunicar. Uma forma clara de se comunicar através do código, é manter um padrão no projeto para que qualquer um possa rapidamente entender o código lido.

O XP recomenda a adoção de um padrão desde o início do projeto, preferencialmente padrões utilizados pela comunidade da linguagem de desenvolvimento. Havendo a necessidade de modificações e adaptações durante o projeto, que visam agilizar o desenvolvimento, as mesmas devem ser efetuadas.

2.2.9 *Design* simples

Nota-se que todas as práticas do XP focam que o maior valor possível seja gerado para o cliente, para tal premissa ser verdadeira o XP prega um *design* do sistema da forma mais simples possível para que atenda a necessidade do cliente.

Umas das premissas do desenvolvimento tradicional é que o custo de uma alteração no sistema cresce exponencialmente ao longo do projeto, com isto tenta-se criar soluções genéricas preparando o sistema para futuras alterações.

Este tipo de pensamento dá margens para especulações e implementações que na maioria dos casos não terá utilidade para o cliente. O XP parte do princípio que o custo de uma alteração tende a crescer lentamente e se estabilizar ao longo do projeto, esta premissa é dita em função dos avanços nas linguagens e práticas de programação, novos ambientes e ferramentas de desenvolvimento, utilização de orientação a objetos no desenvolvimento e em conjunto com estes novos avanços existe o fruto das outras práticas XP, deixando o código simples, legível e passível de alteração a qualquer momento.

2.2.10 Metáfora

Muitas vezes pessoas tentam explicar um assunto ou problema a outras pessoas por um período sem obter o êxito necessário na explicação dada, simplesmente as outras pessoas não conseguem entender a mensagem que está se tentando repassar.

Ao criar comparações e analogias com o assunto que está em questão as pessoas passarão a entender deste assunto de uma forma muito mais rápida e possivelmente não a esquecerão mais. Este tipo de artifício é chamado de *metáfora* no XP, e deve ser utilizado com intensidade durante o projeto, uma vez que facilita a comunicação e fixação dos assuntos entre as partes.

Esta prática anda em conjunto com o ritmo sustentável, já que para o desenvolvedor criar boas metáforas exige criatividade e desenvolvimento de idéias, o que torna necessário uma boa condição física e bem estar por parte do desenvolvedor.

2.2.11 Ritmo sustentável

Uma grande problema nos projetos de software é a falta de tempo para encerrar o mesmo, e uma das técnicas

mais adotadas para compensar a falta de tempo é submeter os desenvolvedores (humanos) a trabalharem até mais tarde e muitas vezes sacrificarem seus finais de semana.

Nos primeiros momentos este tipo de atitude tem efeitos positivos, porém passado alguns dias o rendimento da equipe cai drasticamente, dando margens a erros pela falta de concentração no desenvolvimento, justamente pelo cansaço físico do desenvolvedor.

O XP proíbe que os desenvolvedores trabalhem até mais tarde. O XP sugere um ritmo sustentável de 40 horas semanais, respeitando assim a individualidade e o físico de cada desenvolvedor. Desta forma a equipe estará sempre concentrada e muito menos propensa a pequenas falhas na implementação.

2.2.12 Integração contínua

No desenvolvimento tradicional geralmente as equipes são organizadas de modo que uma parte (módulo) fiquei sob responsabilidade de um desenvolvedor, cabe a esta pessoa efetuar testes e codificação que dizem respeito a sua parte. Esta estratégia reduz a complexidade e as preocupações de um desenvolvedor.

Interfaces de integração são convencionadas para que futuramente estas partes possam se comunicar, este tipo de desenvolvimento esta propenso a sérios erros de integração, uma vez que os períodos em que as partes são integradas e testadas são extremamente longos.

O XP propõe uma integração contínua, se possível deve ser efetuada diversas vezes ao dia para que toda a equipe tenha conhecimento do código recém desenvolvido. Com esta prática o *feedback* sobre a alteração efetuada será retornado em um menor espaço de tempo.

2.2.13 *Releases* curtos

No modelo tradicional o projeto é dividido em fases, de um modo que o cliente começará a ter benefício com o sistema muito próximo de o mesmo estar finalizado. A maior parte do investimento do projeto é feita antes mesmo de estar concluído, sem ter gerado qualquer tipo de valor econômico ao cliente.

O XP recomenda que pequenos investimento sejam efetuados de forma gradual e que busque grandes recompensas o mais rapidamente possível. Com a prática de *releases* curtos, o XP pretende dar o máximo de valor econômico ao cliente em um curto espaço de tempo.

Release é um conjunto de funcionalidade bem definidas e que representam algum valor para o cliente. Um projeto XP pode ter um ou mais *releases* no seu ciclo de desenvolvimento.

2.3 Equipe XP

Em uma equipe de XP existem papéis a serem desempenhados por um ou mais desenvolvedores. Estes papéis serão listados a seguir.

2.3.1 Gerente de projeto

Pessoa responsável pelos assuntos administrativos do projeto, mantendo um forte relacionamento com o cliente para que o mesmo participe das atividades do desenvolvimento.

O gerente de projeto é responsável por filtrar assuntos não relevantes aos desenvolvedores e aspectos que só devam ser implementados em iterações futuras.

Para um bom exercício de gerente de projeto é necessário que a pessoa conheça e acredite nos valores e práticas do XP para que possa cobrar isto da sua equipe.

2.3.2 Coach

Pessoa responsável pelas questões técnicas do projeto, recomenda-se que esta pessoa seja a com maior conhecimento do processo de desenvolvimento, dos valores e práticas do XP. é de responsabilidade do *coach* verificar o comportamento da equipe frente o processo XP, sinalizando os eventuais erros cometidos pela equipe.

2.3.3 Analista de teste

Pessoa responsável em garantir a qualidade do sistema através dos testes escritos. Ele deve ajudar o cliente a escrever os casos de testes e no final de cada iteração verificar se o software atende todos os casos de testes.

Recomenda-se que esta pessoa não seja um desenvolvedor, para evitar uma visão tendenciosa já que não conhece o código desenvolvido. O analista de teste deve ter uma visão muito parecida com a do cliente e em muitos projetos esta pessoa acaba exercendo o papel de redator técnico.

2.3.4 Redator técnico

Pessoa responsável em documentar o sistema, evitando um forte trabalho dos desenvolvedores neste tipo de atividade, permitindo uma maior dedicação ao trabalho de codificação.

Esta pessoa deve estar em plena sintonia com os desenvolvedores e cliente para que a documentação reflita o código escrito e as regras de negócio atendidas pelo sistema.

2.3.5 Desenvolvedor

Pessoa responsável em analisar, projetar e codificar o sistema. No XP não existe diferença entre analista, projetista e programador uma vez que em vários momentos do projeto o desenvolvedor estará exercendo alguma destas atividades.

Naturalmente existe níveis distintos de desenvolvedores dentro de uma equipe, mas com as práticas do XP, como *pair programming* , a tendência é a equipe se tornar uniforme em suas habilidades.

3 Conclusões

A metodologia de desenvolvimento *Extreme Programming* pode ser considerada extremamente nova, porém vem acompanhando as necessidades **humanas** dos desenvolvedores pelo mundo.

Gurus da tecnologia da informação vem aprimorando as concepções desta metodologia para atender as necessidades do mercado e principalmente das pessoas.

Um empresa ao utilizar este processo por completo, só estará agregando valor aos seus negócios e melhorando o ambiente de seus colaboradores e clientes, tratando-os como pessoas e parceiros. Está é chave no mundo dos negócios, o bem estar de seus colaboradores e a parceria entre o fornecedor e seus clientes, criando um laço de confiança ou até mesmo um sentimento de amizade.

Entender as necessidades do cliente não é ciência, é arte. Dar incentivo a ela é o mínimo que podemos fazer.

4 Referências

AMBROSI, Cleison Vander; GRAHL, Everaldo Artur. **Extreme Programming: Um modelo de processo para o desenvolvimento de software.** Blumenau ? SC: Instituto Catarinense de Pós-Graduação.

ASTELS, David; MILLER, Granville; NOVAK, Miroslav. **Extreme Programming: Guia prático.** Rio de Janeiro ? RJ: Campus, 2002.

BECK, Kent. **Programação extrema (XP) explicada: acolha as mudanças.** Porto Alegre ? RS: Bookman, 2004.

POHREN, Daniel. **XP Manager: Uma Ferramenta de Gerência de Projetos Baseados em Extreme Programming.** Novo Hamburgo ? RS: Centro Universitário Feevale, 2004.

TELES, Vinícius Manhães. **Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade.** São Paulo - SP: Novatec Editora Ltda, 2004.

WUESTEFELD, Klaus. **Xispê: Extreme Programming.** Disponível em http://www.xispe.com.br/index.html . Acesso em: 23 / 07 / 2004.

5 Autores

Giovane Roslindo Kuhn está cursando o 4 º ano de graduação em Ciências da Computação da Universidade Regional de Blumenau . Atuando profissionalmente no Projeto Jakare da Senior Sistemas , que consiste em um framework para desenvolvimento de aplicações J2EE e é um dos responsáveis do projeto Baba XP . Incentivador do uso de metodologias ágeis de desenvolvimento, especialmente XP e desing patterns.

Vitor Fernando Pamplona está cursando o 4 º ano de graduação em Ciências da Computação da Universidade Regional de Blumenau . É entusiasta do Prevayler, da XP e do Software Livre. Participa profissionalmente em um projeto com Swing, J2EE e Hibernate, é um dos responsáveis pelo projeto Baba XP , articulista do iMasters , um dos coordenadores do JavaFree Blumenau JUG e um dos administradores do portal sobre java e software livre, JavaFree.org