5
Organização

A organização do ambiente de trabalho é fundamental para o bom funcionamento das práticas de XP. Como a documentação é reduzida, é preciso muita comunicação para fazer com que todos saibam o que está sendo feito; como deve ser feito; e como resolver os problemas. Mais do que ciclos semanais e trimestrais de reuniões para definir as tarefas e temas que serão abordados em um determinado período, XP precisa de uma forma de trabalho que estimule a troca de informação entre os programadores.

5.1.Pair Programming

Pair Programming é uma das práticas mais conhecidas e polêmicas de *eXtreme Programming*. Esta define que todo o código que será colocado em produção deve ser feito por dois programadores em um mesmo computador. Enquanto um escreve o código, o outro revisa, e, de tempos em tempos, os pares se alternam. "*Pair Programming* é um diálogo entre duas pessoas que estão programando (analisando, arquitetando e testando) simultaneamente e tentando programar melhor" (Beck & Andres, 2004).

Programar sozinho pode causar vários problemas. Por exemplo, uma pessoa pode não estar em um bom dia. Seus testes podem estar passando, mas ela não está fazendo os *refactorings* necessários. O design pode estar funcionando, mas não ser o mais simples possível. Além disso, o provável aprendizado difundido entre o par de programadores foi evitado – apenas uma pessoa tornou-se familiarizada com o código que foi escrito (Wake, 2001).

A idéia básica por trás do *Pair Programming* é a de que "duas cabeças pensam melhor do que uma". Logo, um par de programadores trabalhando juntos vai produzir código com uma qualidade muito superior ao de um programador sozinho. O resultado será uma quantidade maior de código bem feito, um melhor entendimento do sistema e mais prazer (Jeffries et al, 2001).

A programação em pares é conhecida por produzir os seguintes benefícios: aumenta a disciplina dos programadores, torna o fluxo de trabalho mais estável, melhora o estado de espírito dos indivíduos, proporciona um conhecimento coletivo do código, cria uma coesão da equipe e diminui o número de interrupções. Um parceiro pode ajudar a garantir que os valores e as práticas que devem ser seguidos por uma equipe estão realmente sendo adotados (Wake, 2001).

Além disso, *Pair Programming* tem um impacto muito claro na taxa de defeitos de um sistema, uma vez que funciona como uma contínua revisão do código. Isso tende a diminuir a taxa de defeitos da mesma maneira que outras formas de revisão (Shull et al., 2002; Boehm & Turner, 2003). A vantagem do *Pair Programming* é que a descoberta do problema já permite refletir sobre a sua solução. Uma etapa separada de revisão de código pode detectar muitos erros, mas não garante que eles serão corrigidos. Com *Pair Programming*, os problemas podem ser detectados antes mesmo de uma funcionalidade ser implementada. Através da contínua transferência de quem comanda o teclado, o parceiro condutor está sempre preparado e o parceiro revisor pode refletir sobre como o código que está sendo escrito se ajusta a todo o design (McBreen, 2002).

De acordo com (McConnell, 2004), outros benefícios da programação em pares são:

- ***Sustenta-se melhor sob estresse do que o desenvolvimento individual***. Pares encorajam uns aos outros para manter uma elevada qualidade do código mesmo quando existe uma pressão para escrever rapidamente código desorganizado.
- ***Melhora a qualidade do código***. A legibilidade e a compreensão do código tendem a se elevar ao nível dos melhores programadores na equipe.
- ***Encurta os cronogramas***. Pares tendem a escrever código rapidamente e com menos erros. Conseqüentemente, a equipe envolvida em um projeto gasta menos tempo corrigindo defeitos na etapa final.
- ***Produz outros benefícios da construção colaborativa***. Causa a disseminação da cultura corporativa e a instrução de programadores iniciantes. Como todos estão mexendo em todo o código, também encoraja

a posse compartilhada do código e reduz o impacto da perda de um programador chave para um projeto.

Com a programação em pares, o tempo gasto com treinamento é menor. Um programador não aprende apenas por meio do sistema que está sendo feito ou do material relacionado, mas também com a experiência do seu parceiro. Duas pessoas que trabalham bem em conjunto podem ser mais produtivas que três pessoas trabalhando de maneira isolada – disse Larry Constantine em Software Development (Outubro de 1999).

Para começar a praticar *Pair Programming*, basta ter duas pessoas trabalhando juntas – em um único computador – de forma a produzir a solução. Elas fazem isso como parceiros diretos, adotando turnos de escrita e observação, provendo constante design e revisão do código (Wake, 2001).

Segundo Jeffries et al. (2001), geralmente é melhor que o papel de condutor seja realizado pelo programador que tem menos certeza do que tem que ser feito. É fácil alguém ficar aborrecido quando não compreendeu realmente o conceito do que tinha que ser feito e, então, perde-se a vantagem de trabalhar em pares.

A prática de *Pair Programming* também pode apresentar pontos negativos: programadores experientes podem achar entediante ensinar um desenvolvedor menos experiente; muitos profissionais preferem trabalhar sozinhos; diferenças no estilo de código dos programadores podem resultar em conflitos; problemas de horário podem ocorrer porque nem todas as pessoas seguem a mesma agenda; com a difusão do trabalho à distância, muitas pessoas trabalham de casa, o que torna a programação em pares muito difícil ou simplesmente impossível de ser praticada.

Programadores de diferentes categorias trabalhando juntos também podem ser um problema. De acordo com (Stephens & Rosenberg, 2003), os seguintes cenários podem acontecer:

- ***Experiente x Novato***: O experiente pode se achar o responsável por todo o desenvolvimento, não se preocupando em ensinar nada para o programador mais inexperiente. Além disso, o mais veterano pode simplesmente ignorar as opiniões do desenvolvedor novato.
- ***Novato x Novato***: A falta de experiência pode levar muitas vezes os dois programadores a não saber o que fazer ou tentar soluções pouco apropriadas.

- ***Experiente x Experiente***: Discussões que estão mais ligadas ao ego de cada um dos programadores podem desvirtuar o sentido real do trabalho em pares.

Pair Programming é uma técnica que possui pouco tempo de estudos dando suporte à sua efetividade, ao contrário do que acontece, por exemplo, com inspeções formais. Mas os dados iniciais sugerem que esta prática apresenta benefícios similares aos da inspeção. Até mesmo os relatórios que não são baseados em fatos apresentam dados positivos (McConnell, 2004).

O custo de desenvolvimento para se obter esses benefícios não é de 100%, como é de se esperar. Ou seja, dois programadores juntos não demoram o dobro do tempo para fazer o que dois programadores separados fariam. O estudo usualmente citado sobre *Pair Programming* considera que a programação em pares aumenta a qualidade do código em 15%, mas demora, em média, 15% mais tempo (Cockburn & Williams, 2000).

Um estudo recente, feito sob a forma de um rigoroso experimento científico, verificou que pares compostos por novatos são muito mais produtivos do que programadores inexperientes trabalhando sozinhos. Programadores experientes trabalhando em conjunto, no entanto, não apresentaram ganhos tão significativos em relação ao trabalho desacompanhado. (Lui & Chan, 2006). Foram definidos dois princípios, então:

Princípio A: Quando um problema de programação é novo e demanda muito esforço para produzir o design, o algoritmo e o código, um par de programadores é muito mais produtivo e pode produzir uma solução muito melhor do que dois deles trabalhando individualmente (Lui & Chan, 2006).

Princípio B: Dois programadores não precisariam trabalhar juntos para resolver um problema conhecido, pois eles seriam mais produtivos trabalhando individualmente (Lui et al, 2006).

Com relação à quantidade de erros encontrados em um sistema, *Pair Programming* é uma prática complementar ao *Test Driven Development*. Como a programação em pares está relacionada com a constante revisão e troca de informações, esta ajuda a resolver problemas que não seriam resolvidos apenas com os testes. Além disso, fazer testes nem sempre é uma tarefa fácil. Compartilhar com outra pessoa a criação deles e da implementação que os faz funcionar pode ser muito útil.

De maneira geral, *Pair Programming* dá suporte e habilita a maioria das outras práticas de *eXtreme Programming*. Esta é também a prática que tem maior influência em como os programadores trabalham, pois requer que todo o código que vai para produção seja escrito com um parceiro. *Pair Programming* é o mecanismo que XP usa para garantir obediência ao resto das práticas de programação, porque com a constante troca de pares, a equipe inteira pode ver quem não está realmente adotando as regras (McBreen, 2002).

5.2.Infra-estrutura

A prática de *Pair Programming* não depende de ferramentas para que seja bem sucedida. Porém, uma infra-estrutura adequada deve ser montada. Como duas pessoas podem gostar de trabalhar se o espaço para elas for apertado, incômodo e desconfortável? Como é possível escrever e revisar o código se o monitor for pequeno, a visão dificultada pelo reflexo do sol e etc.?

Para que a programação em pares funcione, o ambiente de trabalho deve estar preparado. As máquinas devem ser dispostas de forma que duas pessoas possam sentar lado a lado. Deve haver espaço para que o teclado e o mouse possam ser manuseados por qualquer um dos desenvolvedores de maneira confortável.

O ambiente deve ser propício e isso significa que deve ser espaçoso. Os assentos devem ser confortáveis. O monitor dever ser grande e possuir uma resolução favorável. O formato e a disposição da mesa onde o trabalho é realizado são críticos para o *Pair Programming*. O computador colocado em um canto fechado não funciona. Duas cadeiras lado a lado direcionadas para o monitor, este é o caminho certo (Jeffries et al., 2001).

Ao mesmo tempo em que o espaço físico deve ser apropriado para o trabalho em conjunto, a privacidade de cada um deve ser respeitada. É muito chato, por exemplo, ler e-mails pessoais na frente de um colega de trabalho. Por isso, espaços reservados devem ser feitos para que as pessoas possam ler um livro ou artigo, consultar seus e-mails e acessar a Internet.

5.3.Precauções

Para que *Pair Programming* dê certo, o primeiro passo é não ter pressa para que todos estejam adotando esta técnica à risca. Comece com uma parceria para revisar o código e para encontrar boas soluções para problemas não triviais. Com o tempo, os programadores ganharão maturidade com o uso da prática, ao ponto de fazê-la como é esperado.

Alguns problemas com *Pair Programming* podem surgir por causa de fatores distintos, como a dinâmica social, a falta de privacidade, a ausência de um tempo para pensar com mais calma e complicações ergonômicas. Outra anomalia que costuma acontecer é o fato de que um dos programadores toma posse do teclado e não o compartilha. É possível que um programador passe dias ou até semanas sem digitar uma palavra.

Por isso, apesar de ser um conceito básico, o seu uso pode ser beneficiado por alguns princípios:

- ***Existe algum padrão de código de forma que os programadores não percam tempo filosofando sobre o estilo de código?***

  Utilize padrões de código. A programação em pares não será efetiva se as duas pessoas do par desperdiçarem seu tempo argumentando sobre estilo de código. Dessa forma, os programadores podem focar no problema essencial a ser resolvido.

- ***Os dois programadores de cada par estão participando ativamente?***

  Não deixe a programação em pares se tornar observação. A pessoa que não possui o teclado deve ser um participante ativo na programação. Esta pessoa deve analisar o código, pensar sobre qual o próximo passo a ser implementado, avaliar o design e planejar como testar o código.

- ***É feita uma seleção das tarefas que serão realmente beneficiadas pelo uso de Pair Programming?***

  Nunca force *Pair Programming* para tarefas simples. Se editar arquivos HTML, por exemplo, for uma tarefa de domínio geral dos programadores, não faça com que eles trabalhem em conjunto para escrever uma página Web simples.

- ***A determinação de tarefas e a organização dos pares são variadas?***

  Varie os pares e as tarefas designadas regularmente. Na programação em pares os benefícios aparecem porque diferentes programadores estão aprendendo diferentes partes do sistema.

- ***Algum programador não está conseguindo acompanhar o desenvolvimento quando feito de forma colaborativa?***

  Encoraje os pares a se adaptarem ao ritmo de ambos. Um parceiro indo muito rápido limita os benefícios de se ter outro parceiro. O parceiro mais rápido precisa diminuir a velocidade ou o par deve ser desfeito e uma nova arrumação deve ser arranjada.

- ***Alguém está se esforçando demais para entender o que está escrito?***

  Garanta que os dois desenvolvedores possam ver o monitor. Até mesmo detalhes aparentemente simples, como não ter uma boa visibilidade do monitor ou usar fontes muito pequenas, podem causar problemas.

- ***Os pares estão formados de acordo com a cadência e a personalidade de cada um?***

  Não force pessoas que não gostam uma da outra a formarem um par. Algumas vezes, conflitos de personalidade impedem que pessoas façam *Pair Programming* de forma efetiva. Não faz sentido forçar uma pessoa que não se entende com o seu par a fazê-lo. É preciso ter sensibilidade com relação à compatibilidade de personalidades.

- ***Existe um responsável por organizar os pares e gerenciar o contato com pessoas externas ao projeto?***

  Determine um líder para a equipe. Se toda a equipe quiser fazer 100% da programação em pares, ainda assim é necessário que uma pessoa coordene o desígnio de tarefas, dê conta dos resultados e atue como um ponto de contato para pessoas fora do projeto.

"Trabalhar em parceria não é algo que ocorre naturalmente para todo mundo, mas as pessoas tendem a gostar disso depois que elas tentam" (Jeffries et al., 2001). Algumas técnicas, no entanto, podem ser utilizadas para dar início ao uso da prática:

- ***Peça ajuda por meio de parceria***. Esta é uma abordagem poderosa para resolver muitos problemas. Além disso, permite que o parceiro colaborador se sinta mais útil e importante.
- ***Ofereça ajuda por meio de parceria***. Quando alguém perguntar como alguma coisa funciona, ou como fazer alguma coisa, tente dizer "Eu tenho alguns minutos. Vamos dar uma olhada nisso juntos." Vá para frente da máquina do seu parceiro, deixe o teclado com ele e ofereça ajuda dessa maneira.
- ***Envie ajuda por meio de parceria***. Depois que as pessoas se acostumarem a fazer parceria com você, estimule-as a fazer parcerias com outras pessoas também.
- ***Apareça sem avisar***. Simplesmente sente-se ao lado uma pessoa que esteja parecendo confusa e pergunte: "O que você está fazendo?"

Uma conseqüência do uso de programação em pares é a difusão do conhecimento, em detrimento da formação de especialistas. Se a rotatividade dos programadores for razoável, todos os programadores terão grandes chances de conhecer quase todo o código existente. Todos os programadores possuirão noções de como trabalhar com várias tecnologias e resolver problemas diversos. Porém, dificilmente existirão pessoas com um domínio muito grande de partes específicas.

A empresa deve analisar o que acha mais vantajoso. Possuir especialistas também é saudável e o ser humano tem a tendência de se especializar naquilo que faz. Uma solução para esse problema é nunca deixar que partes críticas tenham apenas um especialista. Assim, as informações cruciais estarão sempre difundidas.

Em empresas pequenas, normalmente há dois possíveis cenários: muitos projetos para poucos programadores (por exemplo, 4 projetos para 5 programadores) ou poucos projetos para muitos programadores (por exemplo, 2 projetos para 10 programadores).

No primeiro cenário, torna-se difícil fazer *Pair Programming* se todos os programadores tiverem que trabalhar em todos os projetos. Nestes casos, é melhor ter uma rotatividade menor de pares (um par para cada dois projetos). Dessa forma a informação estará distribuída, mas sem o problema de "em qual projeto eu estou trabalhando agora?" ou "o que nós temos que fazer mesmo?".

Quando a quantidade de projetos é menor, é mais fácil fazer com que todos estejam integrados para resolver os problemas. Todos sabem com muito mais facilidade o que deve ser feito, o porquê, para quem e até quando.

No contexto atual de desenvolvimento de software, a tecnologia já permite que uma pessoa trabalhe de casa ou fora da empresa sem qualquer perda no seu desempenho. Como fazer *Pair Programming* se o restante da equipe estiver em outra cidade ou do outro lado do mundo? Já existem ferramentas que apresentam soluções para interação multi-usuário em uma mesma IDE. Mais sobre o assunto pode ser lido em (Baheti et al., 2002a, b).

Pesquisas mostram que as pessoas gostam mais de trabalhar em pares do que sozinhas (Cockburn & Williams, 2000). Isso não significa que não existam pessoas que não gostem de trabalhar sozinhas. Por isso, obrigar uma pessoa fazer o seu trabalho da forma que ela não gosta, nunca será mais produtivo, independente das vantagens que qualquer prática possa trazer.

Pair Programming e inspeções formais produzem resultados similares em termos de qualidade, custo e planejamento. Por esse motivo, a escolha entre elas torna-se um problema de estilo pessoal ao invés de um problema de caráter técnico. Algumas pessoas preferem trabalhar sozinhas com algumas interrupções para reuniões de inspeção. Outras preferem gastar mais do seu tempo trabalhando diretamente com outras pessoas (McConnell, 2004).

Pair Programming não é uma prática decisiva para que *Test Driven Development* funcione. Porém, é muito melhor ter alguém para ajudar a pensar nos testes que devem ser escritos, principalmente quando ainda não se tem o domínio dessa técnica. Não há problemas em pessoas trabalhando aos pares se elas realmente quiserem fazer isso. Por isso, não faz sentido coibir a programação em pares. Tornar essa prática obrigatória é muito diferente, no entanto.