1
Introdução

Muitas micro e pequenas empresas brasileiras de informática gostariam de melhorar a forma como desenvolvem software. Isso não decorre de uma necessidade externa como, por exemplo, a exigência dos clientes, mas de uma vontade interna de aprimorar a forma como se trabalha. E uma das formas de aperfeiçoar essa qualidade baseia-se no uso de um processo de desenvolvimento de software.

Esse objetivo pode ser atingido com a adoção de um processo de controle de qualidade rígido, como o *Capability Maturity Model* (CMM), ou seus derivados. No entanto, Orr (2002) afirma que "certificações CMM estão mais relacionadas com a gerência do processo (estimativas, cronograma e controle) e não tanto com a qualidade do software que é produzido". Além disso, processos rígidos apresentam alto custo de implantação e envolvem excessiva burocracia. Por isso, o seu uso é desencorajado nas pequenas empresas.

Novas metodologias, chamadas ágeis, estão surgindo em um ambiente de desenvolvimento de software para equipes pequenas. Essas metodologias têm como característica definir processos de controle de qualidade, mas sem a complexidade de processos robustos como o CMM. O destaque entre elas é o *eXtreme Programming* (XP), ágil e composto por um conjunto de boas práticas que, como o seu título sugere, são levadas ao extremo. Uma dessas práticas, chamada *Test Driven Development* (TDD), tem como objetivo garantir a criação de código limpo e funcional.

TDD começou a ser divulgado no início do ano 2000, como uma das práticas do *eXtreme Programming* (XP). "Recentemente, vem atraindo um interesse maior na sua própria direção" (Newkirk & Vorontsov, 2004). Esta é uma técnica de programação que envolve a criação, em primeiro lugar, de um caso de teste e, depois, da implementação do código necessário para fazer com que o teste passe. A prática deve ser efetuada repetidamente, até que todas as funcionalidades

desejadas tenham sido implementadas. O objetivo do *Test Driven Development* é obter feedback rapidamente.

Para analisar os ganhos efetivos dessa prática, foi proposto e avaliado um ambiente propício ao desenvolvimento de software de forma incremental e baseada em testes.

Algumas perguntas que devem ser respondidas são:

- Quais as ferramentas necessárias para seguir essa prática com eficiência?
- Quais os pontos fortes e fracos dessa abordagem?
- Que cuidados devem ser tomados?
- Quais as restrições que inviabilizam o uso dessa prática?

Partindo do pressuposto de que esta técnica, quando adotada por equipes pequenas, proporciona melhoria no desenvolvimento de software, é feita uma análise utilizando como base o desenvolvimento de sistemas reais. Isto torna a pesquisa bastante prática. Com essa avaliação, foi possível julgar a real eficácia da técnica, além de apontar os pontos fortes e fracos da abordagem.

1.1.Motivação

Este trabalho foi fruto do envolvimento com uma empresa de desenvolvimento de software que busca formas de melhorar a qualidade dos produtos que desenvolve.

"Qual é a atividade ou fase que melhora a qualidade de uma aplicação? A resposta é fácil: testar e testar muito" (Murphy, 2005). Por isso, *Test Driven Development* foi escolhido como a técnica a ser empregada para garantir maior qualidade do código que é escrito. Como nenhum teste era feito anteriormente, qualquer teste seria melhor do que nenhum.

Mas por que abordar outras práticas além do TDD? Apesar de poder ser aplicada isoladamente, esta técnica faz parte de um conjunto completo de práticas do processo de desenvolvimento ágil chamado *eXtreme Programming*. Muitas destas práticas são interdependentes. Por isso, para utilizar TDD de forma produtiva, percebeu-se que era necessário organizar uma infra-estrutura apropriada.

Pequenas empresas freqüentemente trabalham com prazos apertados e não têm condições de parar por muito tempo a produção de software para dar treinamento aos seus funcionários. Por isso, a introdução de novas técnicas e o aprendizado de novas metodologias deve interferir minimamente na produção dos seus programadores.

Além disso, uma característica interessante de *eXtreme Programming* em ambientes como o descrito acima é a idéia de melhoria baseada na eliminação do desperdício de tempo. Ou seja, não é preciso desenvolver mais rápido, basta deixar de fazer aquilo que não é necessário. Evitar o desperdício de tempo permite uma dedicação maior às tarefas mais importantes.

Levando tudo isso em consideração, este trabalho tem por objetivo expor aos interessados no assunto como eles podem se beneficiar das lições aprendidas e como podem acertar mais na hora de implantar essa nova técnica de desenvolvimento de software na sua rotina de trabalho.

1.2.Estrutura da Dissertação

O *eXtreme Programming* define diversas práticas com a finalidade de melhorar a qualidade do software que é desenvolvido. Muitas são interdependentes e, apesar de poderem ser usadas isoladamente, elas apresentam ganhos mais efetivos quando utilizadas em conjunto.

O foco deste trabalho é o *Test Driven Development*, porém outras práticas, indispensáveis ao bom funcionamento desta técnica, também são abordadas. As práticas estão divididas em cinco capítulos: 2 Medição e Acompanhamento, 3 Gerenciamento, 4 Integração, 5 Organização e 6 *Test Driven Development*.

Os capítulos 2, 3, 4, 5 abordam essas outras práticas de XP que são complementares ao uso de *Test Driven Development*. Estes capítulos apresentam o estado da arte, seguido de um tópico com sugestões de ferramentas úteis para o funcionamento de cada prática. Por fim, uma seção com recomendações para prevenir que as práticas sejam utilizadas de maneira indevida e para evitar que possíveis problemas ocorram. O capítulo 6 – *Test Driven Development* – segue a mesma estrutura, porém apresenta uma abordagem mais completa.

Uma avaliação empírica das práticas é feita no capítulo 7, acerca dos seguintes temas:

- Planejamento do uso das práticas no ambiente da empresa.
- Preparação do ambiente de desenvolvimento.
- Treinamento.
- Dificuldades encontradas durante o processo.
- Uma ponderação crítica sobre os pontos fortes e fracos das práticas e do ambiente preparado.

O capítulo 8 apresenta a conclusão deste trabalho. Os trabalhos futuros são abordados no último capítulo.

As práticas são conceitos que, nesta dissertação, foram aplicados com base na linguagem de programação Java. Além disso, as ferramentas sugeridas e todo o ambiente preparado para a experiência também utilizam esta linguagem de programação. Isso não significa, contudo, que as práticas não podem ser aplicadas utilizando-se outras linguagens de programação.