9
Conclusão

Neste trabalho, foi possível observar os benefícios trazidos pela utilização de boas práticas da Engenharia de Software e do *Scrum* em um ambiente real de desenvolvimento. Apesar de algumas dificuldades terem sido encontradas no meio do caminho e de nem todos os objetivos deste trabalho terem sido completamente satisfeitos, muitas melhoras foram obtidas, podendo-se dizer que o trabalho, como um todo, foi bem sucedido.

A utilização de um gerenciamento de configuração, composto por um sistema de controle de versão e de um controle de mudanças, conseguiu diminuir drasticamente a quantidade de erros encontrados no site da organização estudada.

Esta tecnologia também possibilitou uma melhora no processo de atualização dos servidores, facilitando o trabalho dos integrantes da unidade.

Estas mudanças dependeram muito mais da estruturação do ambiente do que do próprio trabalho da equipe como um todo. Isto é, todo o ambiente de repositório de dados, instalação de programas necessários e desenvolvimento do programa para gerenciar as cópias de arquivos foram realizados por uma só pessoa.

Os outros integrantes da equipe precisaram apenas de algum treinamento relativo a como usar o controle de versão dentro da IDE utilizada e a como usar a ferramenta de cópia de arquivos criada. Ou seja, não foi necessário um investimento de tempo muito grande em treinamentos e nas respectivas curvas de aprendizado, tornando as mudanças mais viáveis e baratas, além de apresentarem muitos benefícios de uso.

Outro ponto que impactou diretamente a diminuição do número de erros encontrados foi a criação de um padrão de codificação, contendo boas práticas, acompanhado da utilização de um framework de desenvolvimento e de atividades de *code review*. A partir do padrão de codificação criado, passou-se a ter uma preocupação com boas práticas que antes eram negligenciadas pelos desenvolvedores, que acabavam gerando erros futuros nas páginas.

Sabendo-se que o código será revisado e que só será aprovado caso siga os padrões descritos, os desenvolvedores passaram a ter uma preocupação em desenvolver seu código de maneira correta (coerente com o padrão, deixando de abrir brechas que poderiam causar problemas aos usuários nas páginas posteriormente). Assim, a qualidade do código desenvolvido foi melhorada.

Em relação ao *Scrum*, foi mais difícil convencer a equipe como um todo para adotá-lo, uma vez que ele utiliza conceitos próprios, que não eram conhecidos pelos integrantes da unidade estuda. Deste modo, o *Scrum* foi utilizado apenas entre os desenvolvedores, durante o desenvolvimento de alguns projetos.

Os estudos duraram aproximadamente cinco meses e foi possível perceber uma evolução da equipe em relação aos conceitos do *Scrum* e também uma melhora em relação às estimativas de tempo e esforço.

O *Scrum* é recomendado para utilização em projetos não muito complexos ou críticos e entre equipes pequenas e multidisciplinares, o que acontece na unidade estudada. Entretanto, ele também exige uma flexibilidade grande por parte da organização – uma vez que não existe a figura do gerente e os próprios integrantes do time têm que se auto-gerenciar, auto-organizar e precisam saber responder às mudanças que ocorrem.

Também é muito importante haver integração entre os componentes da equipe, com conversas rápidas e face a face, o que muitas vezes não era o caso neste estudo, prejudicando o método. Os desenvolvedores ficam fisicamente muito distantes uns dos outros, o que prejudica a troca de informações constante.

De uma maneira geral, muitas vezes é importante tentar balancear estes fatores e até mesmo partir para uma abordagem híbrida, em que o melhor de cada método – seja ele um método ágil ou uma metodologia dirigida a plano – possa ser utilizado de acordo com o que mais se aplica às necessidades da organização.

A ferramenta Banana Scrum também teve um papel importante no desenvolvimento deste trabalho, uma vez que algumas vezes as reuniões diárias foram realizadas online, quando um desenvolvedor não podia estar presente no horário reservado. Além disto, os gerentes podiam acompanhar o andamento do projeto, visualizando gráficos e imprimindo relatórios conforme a sua necessidade.

Por último, é importante frisar que as métricas levantadas neste trabalho são de suma importância para a avaliação dos resultados obtidos. Sem as métricas, torna-se difícil avaliar os impactos das mudanças, uma vez que não há dados para comparação. Em um trabalho deste tipo, dois pontos são fundamentais: a escolha de um procedimento de criação de métricas e avaliação (aqui escolhido o Modelo GQM) e a escolha das próprias métricas: é importante ter dados históricos para comparação e sempre que possível deve-se escolher métricas objetivas e que possam ser extraídas de forma automática.

9.1.Trabalhos Futuros

Um fator que chamou a atenção neste trabalho foi a divisão espacial da equipe, na qual algumas vezes tinha seus integrantes em diferentes países, passando temporadas relativamente curtas. Este é até mesmo um fator comum na organização: ter pessoas em diferentes lugares trabalhando em conjunto, utilizando métodos de conferências de voz e emails para comunicação.

O *Scrum* é um método ágil que tem como princípio a integração da equipe, que deve estar fisicamente próxima. Porém, o que acontece se esta equipe não pode estar próxima? Quando, por exemplo, cada desenvolvedor trabalha em uma cidade ou país diferente. Este cenário tem se tornado muito comum com a globalização e a facilidade de comunicação que as novas tecnologias estão proporcionando. Assim, seria interessante verificar se o *Scrum* poderia ser utilizado em um cenário como este, talvez utilizando ferramentas como o Banana Scrum.

Outro ponto importante seria a melhoria do código legado. As melhorias aplicadas por este trabalho estão destinadas aos códigos novos, isto é, àquilo que passou a ser produzido após o início do trabalho. Assim, a utilização do padrão de codificação e do framework melhorou a qualidade do código produzido desde então, contudo, o código legado continua apresentando problemas. Portanto, seria interessante que houvesse um estudo neste sentido: melhorar o código legado de uma maneira eficiente e não tão cara.

Além disto, métodos de testes automáticos não estão sendo muito utilizados na unidade estudada. Apenas testes de carga e de estresse são realizados, porém, seria necessário criar uma série de testes automáticos que pudessem ser

executados a cada grande atualização do servidor. Outra melhora importante seria implementar o controle de mudanças dentro do próprio ambiente de desenvolvimento, como um *plugin* do Eclipse, por exemplo.

Deste modo, este trabalho teve como objetivo fazer uma primeira melhoria na infraestrutura de trabalho da unidade, uma vez que agora há gerenciamento de configuração e um método de gerência de projetos. Seria interessante, então, continuar este trabalho de modo a melhorar ainda mais a infraestrutura, padronizando os processos da unidade, passando a medir e controlar os resultados, com a preocupação de medi-los quantitativamente e também de medir o desempenho, para que métodos de melhoria contínua possam ser utilizados.