

Desenvolvendo com agilidade: Experiências na reimplementação de um sistema de grande porte

Paulo Cheque Bernardo¹, Fabio Kon¹

¹ Departamento de Ciência da Computação
Instituto de Matemática e Estatística (IME) - Universidade de São Paulo

{paulocheque, kon}@ime.usp.br

Abstract. *We describe in this paper our experience using an agile method for the reimplementation of a large-scale corporate software system. We present the case of the Graduate Office of the University of São Paulo where we conducted a complete change in the way that software was developed, both in terms of methodology and of technologies used. Preliminary results indicate that the changes we implemented not only increased the speed of development but also greatly enhanced the quality of the software developed.*

Resumo. *Descrevemos neste artigo nossa experiência usando métodos ágeis para reimplementação de um sistema de grande porte, apresentando o caso da Pró-Reitoria de Pós-Graduação da Universidade de São Paulo onde conduzimos uma completa mudança no modo em que os softwares são desenvolvidos, tanto em termos de metodologia como de tecnologias utilizadas. Os resultados preliminares indicam que as mudanças que implantamos não apenas aumentaram a velocidade do desenvolvimento mas também melhoraram significativamente a qualidade do software desenvolvido.*

1. Introdução

A Universidade de São Paulo possui um dos maiores programas de pós-graduação do mundo. Ele conta com cerca de vinte e cinco mil alunos matriculados e outorga aproximadamente cinco mil títulos por ano. Para administrar os dados de todo o programa e controlar todos os passos da vida acadêmica dos alunos, a Pró-Reitoria de Pós Graduação (PRPG) possui sistemas informatizados. O mais antigo deles é o Fênix, que está em produção há mais de 10 anos. Um outro importante sistema é o FênixWeb, que está no ar desde 1999. Estamos agora substituindo estes sistemas antigos pelo Janus, que surgiu em 2006 da iniciativa da PRPG em modernizar suas aplicações.

Apesar de tanto o Fênix como o FênixWeb terem facilitado o processo de administração da pós-graduação da USP e, por isso, terem tido grande mérito para o seu progresso, atualmente eles não conseguem atender de maneira satisfatória às novas necessidades da universidade. A continuidade e a manutenção destes sistemas são tarefas árduas porque eles utilizam tecnologias antigas, não orientadas a objetos, mas principalmente porque não há uma metodologia de desenvolvimento que torne este processo simples e rápido.

O Janus está substituindo gradativamente estes extensos sistemas e esta transição está sendo relativamente rápida. Este novo programa utiliza tecnologias de ponta, baseadas em software livre e é desenvolvido com a ajuda de práticas de métodos ágeis de desenvolvimento, como Programação eXtrema [Beck and Andres 2004].

2. Sistemas antigos da PRPG

A manutenção e evolução dos sistemas Fênix e FênixWeb se tornou uma tarefa extremamente custosa. Eles foram construídos utilizando tecnologias hoje ultrapassadas e uma metodologia baseada no modelo em cascata iterativo, sem foco na qualidade do código produzido.

Estas aplicações possuem um código pouco coeso e muito acoplado, fazendo com que a inclusão de novas funcionalidades introduza muitas vezes, novos erros em módulos imprevisíveis. A arquitetura do código-fonte assim como a falta de componentes, leva a grande replicação de código. Temos casos onde o código da interface de usuário está intercalado com as regras de negócio e com o código de acesso ao banco de dados. Também há funções que são muito extensas (até centenas de linhas), tornando-as difícil de compreender. Ainda, os nomes utilizados para as variáveis e funções do sistema são fortemente abreviados exigindo a consulta frequente a documentação externa.

Metodologia de desenvolvimento

Algumas das falhas destas aplicações devem-se à falta de uma metodologia ágil para auxiliar o desenvolvimento. Muitas práticas que hoje são comprovadamente responsáveis pela qualidade de um software, não eram tão conhecidas na época em que estes sistemas foram desenvolvidos.

Algumas práticas importantes não foram aplicadas, como o compartilhamento de código, repositório único para o sistema, testes automatizados e comunicação frequente com o cliente através de contato oral e não por documentos escritos. Estes sistemas não utilizam ferramentas para o controle de versão e muitas das funcionalidades não atendem todas as exigências dos usuários. Mas o principal problema é que não havia uma preocupação com a qualidade e a clareza do código-fonte; se o código apenas funcionasse, já era considerado satisfatório.

A junção de tudo isso, implicou na estagnação do progresso destes sistemas. O estudo do código-fonte e da documentação tornou-se um procedimento extremamente árduo. Ainda, o desenvolvimento é marcado pelo medo, pois a chance de adicionar erros é muito grande, pois não há uma bateria de testes automatizados.

3. O Janus

O Janus visa a oferecer um novo sistema integrado de alto nível para a administração da pós-graduação da USP. Ele é um sistema Web, escrito em Java e utiliza tecnologias baseadas em software livre. Utiliza o arcabouço Java Server Faces (JSF) que facilita a construção de uma arquitetura MVC bem estruturada e bibliotecas de componentes para a implementação da interface gráfica como Tomahawk¹. Ele ainda utiliza diversas outras tecnologias, arcabouços, ferramentas e técnicas que facilitam o gerenciamento do projeto e ajudam a melhorar a qualidade do código fonte.

O desenvolvimento do Janus está sendo muito rápido, mesmo tendo uma equipe de desenvolvedores pequena. Com menos de um ano de vida, o sistema já substituiu diversos módulos dos sistemas antigos que foram desenvolvidos ao longo de uma década e possui diversas funcionalidades exclusivas. Isso se deve às práticas ágeis de desenvolvimento.

¹JSF: <http://java.sun.com/javaee/jvaserverfaces>, Tomahawk: <http://myfaces.apache.org>

3.1. Desenvolvimento ágil do Janus

O desenvolvimento do Janus é feito de forma incremental e segue a filosofia dos métodos ágeis, principalmente através de algumas práticas sugeridas pela Programação eXtrema:

Planejamento e Fases pequenas: Com a ajuda da ferramenta XPlanner², organizamos pequenas fases de desenvolvimento que são desenvolvidos em iterações curtas;

Design simples: O sistema está bem modularizado e sem flexibilidade desnecessária. Muito do sucesso do encapsulamento, deve-se às técnicas utilizadas como programação orientada a objetos, reflexão e programação orientada a aspectos;

Testes: O foco principal durante o desenvolvimento está nos testes automatizados [Beck 2002]. Com auxílio de diversas ferramentas e arcabouços como o JUnit, TestNG, Selenium, Emma, Ant e CruiseControl, montamos baterias de testes de unidade, aceitação, segurança e desempenho que são executadas várias vezes por dia;

Refatoração [Fowler 1999]: Por causa da segurança que os testes automatizados nos fornecem, o sistema recebe pequenas refatorações diárias e eventualmente, grandes refatorações que envolvem grande quantidade de código. Utilizamos a IDE Eclipse que fornece boas ferramentas para refatorações;

Programação pareada: Sempre que possível, os programadores trabalham pareadamente de acordo com suas especialidades e de modo que o conhecimento do projeto se espalhe por toda a equipe;

Padronização e Propriedade coletiva do código: Com a ajuda da ferramenta CheckStyle, que permite a criação e verificação automática de padrões de estilos de código, todos os programadores seguem as mesmas regras. Isto torna viável a propriedade coletiva do código. Todos os programadores conseguem trabalhar em qualquer módulo do projeto;

Integração contínua: O Janus possui um repositório único de código auxiliado pelo Subversion³. Todos os desenvolvedores podem trabalhar em qualquer trecho do projeto e o processo de integração das mudanças é muito simples e ágil;

Cliente junto aos desenvolvedores: O *feedback* dos usuários é sempre muito rápido e acertos e refinamentos são colocados em produção diariamente. O Janus possui três tipos de usuários: alunos, docentes e funcionários das secretarias. Os desenvolvedores possuem contato direto e freqüente com todos eles, o que diminui a burocracia na coleta de requisitos.

Este desenvolvimento ágil e sem burocracia trabalha com muita ênfase na qualidade do código e se preocupa pouco com questões burocráticas como comunicação e desenvolvimento baseado em documentos. O projeto do sistema é feito em quadros brancos espalhados pelo ambiente de desenvolvimento ou em pequenos arquivos, quando necessário.

3.2. O Janus em números

O Janus começou seu desenvolvimento em maio de 2006 e entrou em produção em julho do mesmo ano. A equipe não possuía experiência prévia nem em métodos ágeis nem nas

²XPlanner: <http://www.xplanner.org>

³Subversion: <http://subversion.tigris.org>

tecnologias usadas e é composta por dois programadores e dois estagiários, sendo que alguns possuem maior conhecimento das tecnologias enquanto outros têm maior familiaridade com as regras de negócio e com o código dos sistemas legados.

O número de funcionalidades implementadas foi grande para o tamanho da equipe e o curto tempo de vida. Atualmente, o sistema possui cerca de 15 mil linhas de código, distribuídas em aproximadamente 100 classes Java; há ainda por volta de 50 arquivos JSP. Quanto aos testes automatizados, há cerca de 5 mil linhas de código de teste em 50 classes, cujos testes são executados várias vezes por dia.

Comparando o progresso do Janus com o dos sistemas anteriores, podemos afirmar que, até o momento, o emprego de métodos ágeis, que se adaptam ao ambiente e à equipe, está sendo muito benéfico, não apenas pela velocidade do desenvolvimento, mas também por toda a cultura de flexibilidade e qualidade de trabalho que está sendo transmitida para as outras equipes.

4. Conclusões

Até o momento, o progresso do sistema Janus está se dando de maneira satisfatória e relativamente rápida se comparada ao passado recente no mesmo ambiente. Tanto os desenvolvedores como os gerentes de divisão não possuíam experiência prévia nas práticas ágeis. Ao longo de um ano de trabalho, o líder do projeto de desenvolvimento (segundo autor) e um dos estagiários (primeiro autor) foram capazes de mostrar à equipe os valores dos métodos ágeis e os seus benefícios. Aplicar estes novos conceitos em um ambiente no qual as pessoas trabalharam de modo tradicional durante vários anos foi uma tarefa desafiadora e penosa, pois os valores básicos são muito diferentes.

À medida que o sistema amadurece, a aceitação destes novos princípios deve aumentar, desde que o software continue apresentando uma boa qualidade e continue sendo desenvolvido rapidamente. Mas a tendência é o desenvolvimento tornar-se cada vez melhor, pois a equipe, que não possuía qualquer experiência, hoje está muito mais sólida e cada vez mais produtiva. Com o bom desempenho do desenvolvimento incremental e com os indícios de que a qualidade do trabalho tende a aumentar, podemos dizer que métodos ágeis são compatíveis e eficientes com o desenvolvimento de sistemas corporativos do porte do Janus.

Quanto maior o sistema, maior sua complexidade e o número de pessoas envolvidas. Por isso, a necessidade de adaptação a mudanças e a agilidade na comunicação da equipe com os clientes é indispensável. Estas características são princípios fundamentais das metodologias ágeis, ao contrário das tradicionais que possuem processos rígidos e comunicação baseada em documentos tornando o desenvolvimento lento.

Referências

- Beck, K. (2002). *Test-Driven Development: By Example*. Addison-Wesley.
- Beck, K. and Andres, C. (2004). *Extreme Programming Explained: Embrace Change, 2nd Edition*. Addison-Wesley.
- Fowler, M. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley.