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

Paulo Cheque Bernardo[1] **, Fabio Kon**[1]

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

1. Introdução

A Universidade de São Paulo tem um dos maiores programas de pós-graduação do mundo. Ele possui 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. Outro 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 seus sistemas.

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].

Os quatro princípios das metodologias ágeis descritos no Manifesto Ágil [Beck et al. 2001] são compatíveis com um desenvolvimento rápido de aplicações.

2. Sistemas antigos da PRPG

A manutenção e evolução dos sistemas Fênix e FênixWeb se tornou uma tarefa extremamente custosa. Eles foram construídas utilizando tecnologias hoje ultrapassadas e uma metodologia baseada no modelo em cascata iterativo, sem foco na qualidade do código produzido. O primeiro foi criado com a ajuda de uma versão antiga de PowerBuilder[1] enquanto o segundo é uma aplicação Web, desenvolvido em Java mas utilizando apenas JSP e Servlets[2], sem a ajuda de arcabouços e bibliotecas de auxílio.

Estes sistemas possuem um código pouco coeso e muito acoplado, fazendo com que a inclusão de novas funcionalidades introduza muitas vezes, novos erros em lugares imprevisíveis do sistema. 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.

Além disso, os sistemas utilizam muitas *stored procedures* em SQL, que apesar de possuir algumas vantagens em termos de desempenho, possuem uma linguagem não orientada a objetos e inapropriada para definição de regras de negócio, tornando difícil o entendimento e a reutilização do código.

2.1. 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 freqüente 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.

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.

---

[1] PowerBuilder é um sistema de desenvolvimento de aplicações da Sybase, semelhante ao VisualBasic da Microsoft

[2] Servlets são componentes Java para acessar serviços Web. Hoje todos os arcabouços Java para Web são extensões de Servlets que possuem uma melhor abstração dos serviços.

3. O Janus

O Janus é um novo software da PRPG que visa a oferecer um sistema integrado de alto nível para a administração da pós-graduação da USP. Na mitologia romana, Janus foi o deus das portas, da transição entre o passado e o futuro. Esta aplicação está modernizando os sistemas da PRPG e integrando as funcionalidades dos sistemas antigos.

Este novo sistema é uma aplicação para Web, escrito em Java e utiliza tecnologias baseadas em software livre. Ele utiliza o arcabouço Java Server Faces (JSF) que facilita a construção de uma arquitetura MVC bem estruturada, Java Server Pages (JSP) e bibliotecas de componentes para a implementação da interface gráfica, como Tomahawk e Jenia. 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.

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 XPlanner, 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 AOP;
- **Testes:** O foco principal durante o desenvolvimento está nos testes automatizados [**?**]. 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 [?]:** 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 SubVersion. 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 possuia experiência prévia nem em métodos ágeis nem nas 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 já existentes.

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.

4. Conclusões

Até o momento, o progresso do sistema Janus está se dando de maneira satisfatória e relativamente rápida, se comparado com os outros projetos passados desenvolvidos no mesmo ambiente. Tanto os desenvolvedores como os gerentes de divisão não possuiam experiência prévia nas práticas ágeis. Aplicar estes novos conceitos em um ambiente no qual as pessoas trabalham de modo tradicional durante vários anos é uma tarefa desafiadora, pois os valores básicos são completamente diferentes.

À medida que o sistema amadurece, a aceitação destes novos princípios deve aumentar, desde que o software continue apresentando uma boa qualidade e seja desenvolvido cada vez mais 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.

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.

Beck, K. et al. (2001). Manifesto for Agile Software Development. Home page: `http://agilemanifesto.org` .

Fowler, M. (1999). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley.