APLICAÇÃO DE MÉTODOS ÁGEIS EM UM PROCESSO DE DESENVOLVIMENTO DE SOFTWARE[1]

Cleo Hickmann Junior[2] <cleo.junior@gmail.com>
Anderson Yanzer[3] <anderson.yanzer@gmail.com> – Orientador

Universidade Luterana do Brasil (Ulbra) – Bacharelado em Sistemas de Informação – Campus Canoas
Av. Farroupilha, 8.001 – Bairro São Luís – CEP 92420-280 – Canoas - RS

28 de novembro de 2011

RESUMO

São diversos os motivos pelos quais estudos apontam que a maioria dos projetos de software falha, seja porque o orçamento foi extrapolado, o cronograma não foi cumprido, as funcionalidades não atendem a real necessidade do cliente ou porque estes fatores estejam em conjunto. Com base nisso, este trabalho propõe, através de um estudo de caso, aplicar técnicas e métodos ágeis em uma equipe composta por dois setores de desenvolvimento da empresa Data Cempro Informática. Os objetivos que se espera alcançar são: (a) melhorar a qualidade da codificação dos sistemas e (b) disseminar conhecimento entre a equipe. Para avaliar este projeto, foi realizado um estudo de caso durante 30 dias. Ao final, são apresentados os resultados qualitativos obtidos.

Palavras-chave: Métodos Ágeis; Desenvolvimento de Software; Programação Extrema.

ABSTRACT

Title: *"Application of agile methods on a software development process"*

There are several reasons why studies show that most software projects fail, either because the budget was extrapolated, the schedule was not met, the features do not meet the real needs of the client or because these factors are together. Based on this, this paper proposes, through a case study, to apply agile methods and techniques in a team composed of two development sectors of firm Data Cempro Informática. The objectives to be achieved are: (a) improve the quality of coding of systems and (b) spreading knowledge among the team. To evaluate this project, we performed a case study for 30 days. In the end, the qualitative results obtained are presented.

Key-words: *Agile Methods; Software Development; Extreme Programming.*

1 INTRODUÇÃO

Mesmo com a evolução dos computadores, das técnicas e ferramentas nos últimos anos, a produção de softwares confiáveis, com o mínimo de erros possíveis e entregues dentro do prazo, sem extrapolar o custo estipulado ainda é uma tarefa muito difícil. Com base nestas necessidades surgiram os métodos e práticas ágeis no desenvolvimento de software, visando melhorar o processo e simplificar o trabalho da equipe. Tais práticas sugerem uma série de mudanças que vão de encontro ao que aborda a engenharia de software tradicional, acostumada a seguir padrões de projetos como o CMMI[4], por exemplo.

Desde o ano de 1994 a empresa norte-americana *Standish Group* localizada na cidade de Boston realiza a publicação de um estudo denominado *CHAOS Report*. Esta pesquisa se trata de um levantamento contendo informações de milhares de projetos da área de TI. A versão mais recente do *CHAOS Report* (THE STANDISH GROUP INTERNATIONAL, 2001) aponta uma coleção de dados que envolvem cerca de 280 mil projetos de software nos Estados Unidos, onde os números nos mostram que:

- Os atrasos na entrega dos projetos de software representam em média 63% mais tempo do que o estimado inicialmente.
- O custo adicional médio de um projeto que não cumpriu o orçamento é de 45%.

[1] Artigo elaborado para o Trabalho de Conclusão de Curso II em Sistemas de Informação, submetida ao Curso de Bacharelado em Sistemas de Informação da Universidade Luterana do Brasil, Campus Canoas.

[2] Aluno do 8º semestre do curso de Bacharelado em Sistemas de Informação.

[3] Professor orientador do Trabalho de Conclusão de Curso II em Sistemas de Informação e coordenador do curso de Bacharelado em Sistemas de Informação da Universidade Luterana do Brasil, Campus Canoas.

[4] CMMI (Capability Maturity Model Integration) é um modelo de referência que contém práticas necessárias à maturidade em disciplinas específicas.

- Apenas 67% das funcionalidades prometidas para o cliente na etapa inicial do projeto foram entregues com sucesso, de uma forma geral.

No ano de 2000, o *Standish Group* chegou ao seguinte resultado final com a pesquisa realizada analisando todos os projetos, conforme pode ser visto na Figura 1.

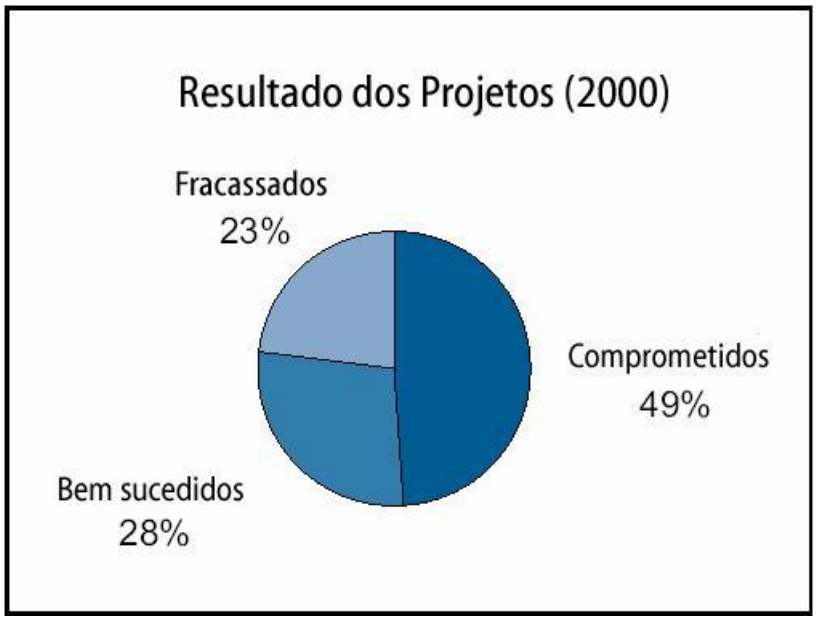

Figura 1 – Estatística sobre os projetos pesquisados pelo *CHAOS Report* (THE STANDISH GROUP INTERNATIONAL, 2001)

Já no ano de 2002 durante a Terceira Conferência Internacional sobre *Extreme Programming*, o presidente da *Standish Group* Jim Johnson apresentou um novo estudo onde foram revelados números sobre a utilização das funcionalidades nos projetos de software (JOHNSON, 2002). O resultado está exposto na Figura 2, que apontou números alarmantes para a comunidade de desenvolvimento de software.

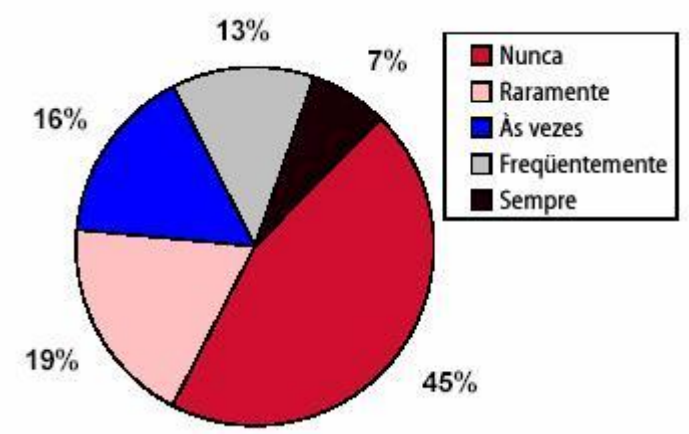

Figura 2 – Gráfico estatístico sobre utilização de funcionalidades (JOHNSON, 2002)

Ao analisar todos estes números e os fatores que levaram esta gama enorme de projetos de software ao fracasso e poucos a serem bem sucedidos, é de extrema relevância avaliar a questão do desenvolvimento de software no sentido de buscar a melhor forma de fazê-lo. E é justamente esta busca pela melhoria contínua que estimulou o autor desta dissertação a propor e sugerir a adoção de práticas e métodos que visam melhorar em um todo o processo de desenvolvimento de software pela sua equipe de trabalho.

Neste contexto de mudança de paradigma, migrando das metodologias pesadas ou tradicionais para a aplicação de processos ágeis de desenvolvimento de software existe uma questão de mudança cultural, o que

muitas vezes dificulta a adoção das metodologias ágeis pela equipe de desenvolvimento. Há muitos exemplos que comprovam que em equipes grandes este processo de mudança é muito mais complicado, muitas vezes até mesmo inviável (YOSHIMA, 2011).

Este artigo tem o objetivo de relatar o resultado de um estudo de caso da aplicação de técnicas e métodos ágeis em uma equipe de desenvolvimento de software cujo processo atual é baseado no modelo tradicional. O desafio da mudança de cultura empresarial em prol de resultados melhores em termos de produtividade e qualidade do software serviu de motivação para o desenvolvimento deste trabalho. Para facilitar o entendimento do leitor, as informações referentes ao cenário, bem como descrição dos processos e da equipe utilizada no estudo de caso serão abordados no decorrer deste artigo.

Na **Seção 2** deste documento consta o referencial teórico utilizado como embasamento e fonte de pesquisa para o tema proposto, enquanto na **Seção 3** será falado sobre a empresa e o cenário utilizado como estudo de caso desta pesquisa. Já a **Seção 4** apresenta os métodos e práticas propostas, juntamente com o instrumento de pesquisa da população afetada e a análise destes dados. O estudo de caso contendo o relato de como foram aplicada as práticas ágeis na **Seção 5**. Na **Seção 6** serão apresentados os resultados obtidos. As considerações finais serão expostas na **Seção 7** e, por fim, o artigo é encerrado com os agradecimentos e as referências utilizadas para elaboração do mesmo.

2 METODOLOGIAS ÁGEIS

O termo "metodologias ágeis" se popularizou em 2001 quando um pequeno grupo composto por dezessete especialistas em processos de desenvolvimento de software se reuniu na cidade norte-americana de *Utah* e estabeleceram princípios comuns compartilhados por todos esses métodos. Com isso, houve a criação da Aliança Ágil, que posteriormente estabeleceu o "Manifesto Ágil" (*Agile Manifesto*). Os conceitos chave do Manifesto Ágil são:

- **Indivíduos e interações** mais que processos e ferramentas;
- **Software em funcionamento** mais que documentação abrangente;
- **Colaboração com o cliente** mais que negociação de contratos;
- **Responder a mudanças** mais que seguir um plano.

Além dos conceitos, o Manifesto Ágil conta com doze princípios para aqueles que pretendem alcançar a agilidade. São eles:

- Nossa maior prioridade é satisfazer o cliente através da entrega contínua e adiantada de software com valor agregado.
- Mudanças nos requisitos são bem-vindas, mesmo tardiamente no desenvolvimento. Processos ágeis tiram vantagem das mudanças visando vantagem competitiva para o cliente.
- Entregar frequentemente software funcionando, de poucas semanas a poucos meses, com preferência à menor escala de tempo.
- Pessoas de negócio e desenvolvedores devem trabalhar diariamente em conjunto por todo o projeto.
- Construa projetos em torno de indivíduos motivados. Dê a eles o ambiente e o suporte necessário e confie neles para fazer o trabalho.
- O método mais eficiente e eficaz de transmitir informações para e entre uma equipe de desenvolvimento é através de conversa face a face.
- Software funcionando é a medida primária de progresso.
- Os processos ágeis promovem desenvolvimento sustentável. Os patrocinadores, desenvolvedores e usuários devem ser capazes de manter um ritmo constante indefinidamente.
- Contínua atenção a excelência técnica e bom design aumentam a agilidade.
- Simplicidade - a arte de maximizar a quantidade de trabalho não realizado - é essencial.

- As melhores arquiteturas, requisitos e designs emergem de equipes auto-organizáveis.
- Em intervalos regulares, a equipe reflete sobre como se tornar mais eficaz e então refina e ajusta seu comportamento de acordo.

Mesmo que o Manifesto Ágil tenha sido assinado em 2001, as ideias subjacentes que guiam o desenvolvimento ágil não são tão recentes. No entanto, estas ideias começaram a ser materializadas somente na década de 1990. Vale ressaltar que em sua essência, os métodos ágeis foram desenvolvidos em um esforço para vencer as fraquezas percebidas e reais da engenharia de software convencional (PRESSMAN, 2006).

Roger Pressman (2006) ainda afirma que o que diferencia as metodologias ágeis das metodologias tradicionais são o enfoque e os valores, ou seja, para uma metodologia ser considerada ágil deve enfatizar pessoas ao invés de processos e algoritmos. Além disso, também é enfatizado pelas metodologias ágeis o gasto de tempo que deve ser menor com documentação e maior com implementação.

Conforme dito por Fowler (2005), a maioria dos projetos de software conta com três suposições-chave, e é justamente o que o processo ágil de desenvolvimento visa atender:

- É difícil prever antecipadamente quais requisitos de software vão persistir e quais serão modificados. Da mesma forma que é difícil prever como as prioridades do cliente serão modificadas durante o desenvolvimento do projeto.
- Para muitos tipos de software, o projeto e a construção são intercalados, ou seja, as duas atividades devem ser realizadas juntas de modo que os modelos de projeto sejam comprovados à medida que são criados. É difícil prever o quanto de projeto é necessário antes que a construção seja usada para comprovar o projeto.
- Análise, projeto, construção e testes não são tão previsíveis (do ponto de vista do planejamento) como gostaríamos.

Segundo consta na *Wikipédia*, o objetivo da maioria dos métodos ágeis é tentar minimizar o risco no processo de desenvolvimento de software. Métodos ágeis priorizam comunicação em tempo real a documentos escritos, ou seja, enfatizam trabalho no software como uma medida primária do progresso. Para Highsmith (2002), a definição de agilidade é a seguinte:

> "Agilidade é a habilidade de criar e responder a mudanças com respeito ao resultado financeiro do projeto em um turbulento ambiente de negócios. Agilidade é a habilidade de balancear flexibilidade com estabilidade."

A visão de Ivar Jacobson (2002) sobre agilidade fornece uma discussão bastante útil, pois ele contextualiza o trabalho de engenharia de software:

> "Agilidade tornou-se atualmente uma palavra mágica quando se descreve um processo de desenvolvimento de software. Tudo é ágil. Uma equipe ágil é uma equipe esperta, capaz de responder adequadamente a modificações. Modificação é aquilo para o qual o desenvolvimento de software está principalmente focado. Modificações no software que está sendo construído, modificações nos membros da equipe, modificações por causa de novas tecnologias, modificações de todas as espécies que podem ter impacto no produto que eles constroem ou no projeto que cria o produto. O apoio para modificações deveria ser incorporado em tudo que fazemos em software, algo que se adota porque está no coração e na alma do software. Uma equipe ágil reconhece que o software é desenvolvido por indivíduos trabalhando em equipes e que as especialidades dessas pessoas e sua capacidade de colaborar estão no âmago do sucesso do projeto."

O Quadro 1 (DYBÅ e DINGSØYR, 2008) apresenta algumas das principais diferenças entre o desenvolvimento tradicional e o desenvolvimento ágil:

Quadro 1 - Comparativo entre Desenvolvimento Tradicional e Desenvolvimento Ágil

	Desenvolvimento Tradicional	Desenvolvimento Ágil
Pressuposto Fundamental	Os sistemas são plenamente determináveis, previsíveis e são	Software de alta qualidade adaptativa é desenvolvido por pequenas equipes

	construídos através de um planejamento meticuloso e extenso.	que usam o princípio do design contínuo de aperfeiçoamento e testes com base no *feedback* rápido e mudanças.
Estilo de Gestão	Comando e controle	Liderança e colaboração
Gestão do Conhecimento	Explícito	Tácito
Comunicação	Formal	Informal
Modelo de Desenvolvimento	Ciclo de vida do modelo (cascata, espiral ou alguma variação).	Modelo de entrega evolutiva.
Desejado Organizacional Forma/Estrutura	Mecanicistas (burocrática com a alta formalização), destinada a grandes organizações.	Orgânico (flexível e participativa incentivando a ação cooperativa social), destinada a pequenas e médias organizações.
Controle de Qualidade	Planejamento pesado e controle rigoroso. Testes pesados.	Controle contínuo dos requisitos de design e soluções. Teste contínuo.

De acordo com Cockburn (2002), a ideia dominante no desenvolvimento ágil é que a equipe de software pode ser mais eficaz na resposta à mudança se puder:

- Reduzir o custo de movimentação de informação entre as pessoas. Para fazer isso, a equipe ágil deve:
  - Colocar as pessoas fisicamente mais próximas;
  - Substituir documentos, falando em pessoa usando a lousa;
  - Melhorar a amabilidade da equipe, seu senso de comunidade e de moral, de modo que as pessoas estão mais propensas a fornecer informações valiosas rapidamente.
- Reduzir o tempo decorrido entre as tomadas de decisão para ver as consequências dessa decisão. Para fazer isso, a equipe ágil deve:
  - Colocar usuários especialistas a disposição da equipe;
  - Trabalhar de forma incremental.

Porém, Ambler (2004) destaca que existem várias implicações interessantes no desenvolvimento ágil de software para profissionais de qualidade:

1. Maior qualidade implica uma menor necessidade de atividades de garantia de qualidade: Código de melhor qualidade implica uma menor necessidade de atividades de garantia de qualidade, tais como revisões e inspeções, em outras etapas do ciclo de vida do *software*. Agilistas são infectados de qualidade.

2. Acostume-se a artefatos "incompletos": Modelos, documentos e código-fonte evoluem ao longo do tempo. Na única vez que agilistas pode dizer que alguma coisa está feita é quando eles estão prontos para liberá-la em produção.

Uma equipe de desenvolvimento ágil normalmente tem comunicação em tempo real entre os integrantes da equipe quanto entre a equipe e as pessoas que conhecem a definição do programa – por exemplo, os clientes. Para tornar o processo mais ágil esta comunicação deve ser preferencialmente feita frente a frente ao invés de usar documentos.

Dentre as metodologias ágeis, podemos citar para fins de exemplificação as mais usadas atualmente: *Extreme Programming* (XP*)*, *Scrum*, *Teste Driven Development* (TDD), *Adaptive Software Development*

(ASD), *Feature Driven Development* (FDD), *Dynamic Systems Development Method* (DSDM), *Lean Software Development* e *Crystal Methods*.

3 CENÁRIO ATUAL

O cenário atual do processo de desenvolvimento de software na empresa onde foi realizado o estudo de caso possui a vantagem de o desenvolvimento ágil de software não ser uma novidade no time. Atualmente a equipe de desenvolvedores faz uso da TDD através de um *framework* próprio que foi desenvolvido derivado da *DUnit*, do Delphi (disponível a partir da versão 2005). A TDD é uma técnica de desenvolvimento de software dirigido por testes, ou seja, o desenvolvedor escreve um caso de teste automatizado que define uma melhoria desejada ou uma nova funcionalidade. Então é produzido código que possa ser validado pelo teste para posteriormente o código ser refatorado para um código sob padrões aceitáveis. Segundo Beck (2003), TDD encoraja designs de código simples e inspira confiança.

Assim como qualquer metodologia de desenvolvimento ágil, a TDD também é composta por pequenos ciclos de desenvolvimentos (STAVARENGO, 2011). Estes ciclos são formados por cinco passos:

1. Criar um teste;
2. Executar o novo teste;
3. Criar a nova função;
4. Executar os testes novamente;
5. Refatorar o código.

3.1 Sobre a empresa

A Data Cempro é uma empresa que há mais de vinte anos atende o mercado de sistemas contábeis com produtos de fácil usabilidade e com a melhor relação custo x benefício para o uso das tecnologias modernas. Desde 1986, a Data Cempro Informática vem acompanhando as contínuas mudanças tecnológicas exigidas pelo mercado. E é através da agilidade neste processo de mudança que seus clientes hoje podem atestar a segurança que a Data Cempro proporciona através dos seus sistemas e serviços, o que também lhe garantiu a conquista do prêmio Top Informática Rio Grande do Sul e o sucesso em todas as feiras e eventos das quais participou.

Situada na cidade de Cachoeirinha, na região da Grande Porto Alegre, atualmente a empresa conta com mais de 60 colaboradores e possui em sua equipe de desenvolvimento cinco núcleos compostos por um gerente, um ou dois técnicos em qualidade de software, em média dois técnicos de suporte e uma quantidade de desenvolvedores que varia de acordo com a necessidade e complexidade dos sistemas. Os núcleos estão divididos em:

- Núcleo Comercial
- Núcleo Departamento Pessoal
- Núcleo Fiscal
- Núcleo Contábil
- Núcleo Web

Cada núcleo é responsável pelos seus sistemas, no entanto o programador que for dar manutenção nos mesmos deve ter o cuidado para garantir que suas alterações não influenciem no funcionamento dos demais sistemas. Lembrando que todos os sistemas são integrados, com os sistemas acessando uma base de dados centralizada, além de muitos arquivos fontes em comum, visando um trabalho com maior conforto e amigabilidade, tanto para os clientes quanto para a equipe de desenvolvimento.

Optou-se por realizar o estudo de caso na Data Cempro Informática por se tratar de uma empresa moderna que, no papel de seus gestores, encoraja os programadores a melhorar o processo de desenvolvimento agregando novas ideias, novas tecnologias e novas metodologias. Este trabalho é só o início de um projeto que tende a se estender pelos demais setores da empresa, de acordo com os resultados obtidos e a aceitação da equipe.

O estudo de caso foi feito a partir de uma equipe que compreende dois núcleos de desenvolvimento (contábil e fiscal). O grupo foi composto por um time de 10 pessoas, contendo 1 gerente de desenvolvimento que também faz o papel de analista, 2 técnicos em qualidade de software e 7 programadores. Optou-se por não utilizar os técnicos em suporte pelo fato de eles não agregarem valor ao software desenvolvido em si, pois o papel deles se restringe ao atendimento a clientes e abertura de novas requisições.

3.2 Sobre os produtos

De acordo com o item 3.1, foi visto que o estudo de caso ocorreu compreendendo o núcleo contábil e fiscal da empresa. Para fins de conhecimento, esta seção faz um breve resumo sobre os dois sistemas de responsabilidade destes núcleos:

3.2.1 WinLivros

O *WinLivros*[5] possui tecnologia de ponta, atualização na legislação do RS e federal, versatilidade ao compor rapidamente todos os relatórios de livros fiscais e até mesmo integrações com SINTEGRA[6], SPED[7] Fiscal, DACON[8], DCTF[9], GIS[10], GIA[11] mensal, GMB[12], contabilidade e importações. Esses são os pontos fortes deste sistema de escrituração fiscal.

O programa para escrituração fiscal da Data Cempro, possui rotina inteligente de lançamentos com sugestão de valores, validação do CFOP[13], acesso à tabela de cadastros de emitentes e destinatários e plano de contas, permite a recuperação da digitação anterior, além de possuir opções para filtros e pesquisas por qualquer campo.

O *WinLivros* está preparado para a geração dos livros fiscais e demais documentos de empresas nas modalidades Geral, Empresa de Pequeno Porte e Microempresa.

3.2.2 ContabMillenium

O *ContabMillenium*[14] foi desenvolvido com o objetivo de atender as necessidades de escritórios de contabilidade e/ou departamentos contábeis, através de um sistema contábil moderno, ágil, flexível, integrado, fácil e com incrível capacidade de mostrar resultados da maneira que você desejar.

O sistema contábil da Data Cempro possui plano de contas totalmente configurável. É fornecido um modelo padrão que pode ser modificado ou excluído pelo cliente, caso este deseje a criação do seu plano de contas do zero, sendo possível também, definir máscaras e níveis a cada grupo criado.

O programa possui rotina inteligente de lançamentos, permitindo o acesso direto ao plano de contas, recuperação da digitação anterior, históricos sem limite de tamanho e conciliação de contas. Possui também opção para filtros e pesquisas por qualquer campo, possibilitando, de forma rápida e fácil, a visualização de lançamentos anteriores e totalizações através da própria tela de lançamentos do sistema.

3.3 Processo atual de desenvolvimento de software

Conforme foi descrito na seção anterior, a Data Cempro Informática conta com cinco núcleos de desenvolvimento onde o trabalho da equipe funciona de forma integrada. Porém, o processo de desenvolvimento de software atualmente segue uma linha tradicional, baseado no modelo cascata (SOMMERVILLE, 2000) e se estabelece resumidamente da seguinte forma:

- Cada núcleo é responsável por um ou mais sistemas de acordo com a sua área de atuação, portanto cabe ao técnico em qualidade de software definir quais requisições farão parte da

[5] Fonte: http://www.datacempro.com.br/winlivros.

[6] Sistema Integrado de Informações sobre Operações Interestaduais com Mercadorias e Serviços.

[7] Sistema Público de Escrituração Digital.

[8] Demonstrativo de Apuração de Contribuições Sociais.

[9] Declaração de Débitos e Créditos Tributários Federais.

[10] Guia de Informação e Apuração do ICMS.

[11] Guia de Informação e Apuração do ICMS Simplificada.

[12] Guia Modelo B.

[13] Código Fiscal de Operações e Prestações.

[14] Fonte: http://www.datacempro.com.br/contabmillenium

próxima liberação de versão (*release*). Esta lista de requisições pode ser alterada de acordo com o andamento do processo de desenvolvimento. No caso de os programadores estarem com suas tarefas adiantadas, alguma requisição que ficou de fora da lista pode ser incluída. Para isso, o programador responsável pela mesma deve avaliar a complexidade da requisição a ser incluída na lista de prioridades. No entanto, se o tempo estiver escasso, alguma tarefa tomar mais tempo do que deveria, e a previsão é de que a fila de requisições não seja concluída, pode acontecer de serem retiradas da lista as requisições com menor prioridade, que normalmente são colocadas no final da fila.

- As requisições são armazenadas em um sistema de suporte desenvolvido internamente, com acesso comum entre todos os membros da equipe. Nele podem ser realizados diversos processos como incluir, modificar e excluir requisições, consultar e emitir relatórios, consultar dados de clientes, etc.
- Conforme citado anteriormente, os testadores organizam as requisições por prioridades, definindo prazos para a liberação das mesmas. Além disso, são eles que encaminham as requisições para os programadores realizarem as implementações solicitadas. Esta lista de prioridades é armazenada em uma planilha do MS Project, localizada em um local comum a todos os membros do núcleo. À medida que as requisições forem concluídas esta lista deve ser atualizada manualmente, podendo ser feito tanto pelo testador como pelo desenvolvedor.
- Após as requisições serem passadas para o programador, este só passará a mesma para a fase de testes posterior à conclusão das alterações solicitadas pelo técnico em qualidade de software. Cabe ao testador avaliar se a necessidade da requisição implementada foi atendida.
- Sempre que necessário, o programador consulta o técnico em qualidade ou até mesmo de suporte para tirar as dúvidas que possam estar travando o processo de desenvolvimento. O contato do programador diretamente com o cliente não deve ser realizado, salvo algumas exceções. Normalmente este contato é feito pelo técnico em suporte que repassa para o testador que por sua vez repassa ao programador.
- Com os devidos testes realizados exaustivamente e bem sucedidos, as requisições são liberadas, ficando prontas para sair na liberação de versão do sistema. Uma liberação de versão pode englobar mais de um sistema, portanto cabe aos técnicos em qualidade de software dos núcleos chegarem a um consenso para definição da data de liberação, bem como os prazos para execução dos testes.

O procedimento descrito acima se enquadra em um modelo de desenvolvimento de software que engenharia de software denomina como modelo cascata, conforme Sommerville (2000) e Pressman (2006). Este modelo também chamado de *"ciclo de vida clássico"* sugere uma abordarem sistemática e sequencial para o desenvolvimento de software, conforme pode ser visto na Figura 3. Todas estas etapas de análise de requisitos, projeto, implementação, testes, integração e manutenção de software ocorrem basicamente nesta ordem pela equipe de desenvolvimento da Data Cempro Informática.

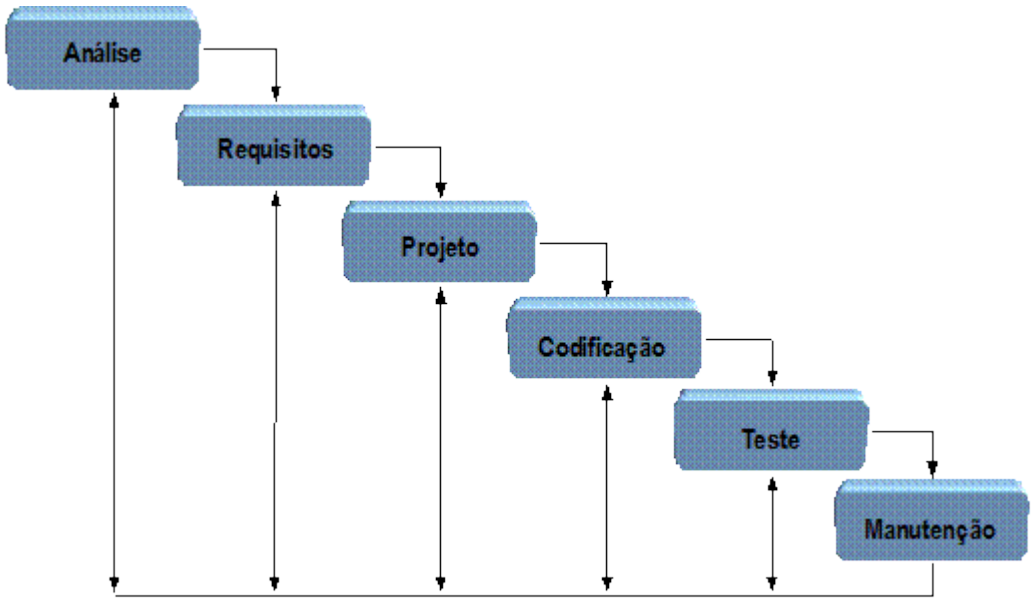

Figura 3 – Modelo Cascata (PRESSMAN, 2006)

Diferente das práticas ágeis de desenvolvimento onde ocorrem as reuniões diárias (*stand up meeting*), cada núcleo da Data Cempro realiza uma reunião semanal de uma hora aproximadamente em horários normalmente pré-estabelecidos. O objetivo é a troca de ideias entre os membros da equipe e exposição de problemas ou dúvidas que possam estar travando o processo de desenvolvimento. Outro fator positivo é a constante troca de ideias entre os desenvolvedores, mantendo um espírito de colaboração e troca mútua de opiniões, críticas e sugestões.

De certa forma, algumas premissas do Manifesto Ágil já são cumpridas na Data Cempro Informática. Ter o software em funcionamento é uma preocupação constante da equipe de desenvolvedores, pois isso vai ao encontro com manter a satisfação do cliente. Sempre que existe a necessidade, seja por uma falha do sistema ou mudança da legislação, é dada prioridade a manter os sistemas em funcionamento do que cumprir requisições planejadas anteriormente. Nestes casos as prioridades são alteradas de acordo com a demanda e a urgência das requisições.

4 MÉTODOS E PRÁTICAS PROPOSTAS

Depois de descrito o cenário atual e o processo de desenvolvimento de software, este capítulo tem por finalidade expor a proposta de melhoria, a forma como este projeto será desenvolvido e o que se espera alcançar ao término deste trabalho. De início, optou-se por adotar práticas de *Scrum* em conjunto às de *Extreme Programming* (XP), porém pela falta de alguns recursos como os cartões de histórias de usuários e a lousa para implantação do *Kanban*, além da impossibilidade de realização das reuniões diárias forçaram o projeto a ser desenvolvido em cima das práticas de *Extreme Programming*.

Com isso definido, o objetivo deste trabalho será validar a adoção das práticas de XP através de um estudo de caso de pelo período mínimo de 30 dias, englobando os núcleos contábil e fiscal da Data Cempro Informática. A ideia é que participem deste projeto o gerente de desenvolvimento destes núcleos, os dois técnicos em qualidade de software (um de cada núcleo), os cinco programadores do núcleo fiscal e os dois programadores do núcleo contábil, compondo assim um time de dez pessoas.

A proposta é de adotar todas as práticas possíveis, dentre as doze que a XP possui. De acordo com alguma limitação ou imposição por parte da gerência, uma prática ou outra pode ser abolida ou até mesmo adaptada. No entanto, anterior às práticas em si haverá uma reunião inicial onde será passado o objetivo deste trabalho para o grupo, bem como uma introdução sobre metodologias ágeis. Além disso, será aplicado um questionário como instrumento de pesquisa no intuito de identificar algumas características da equipe e do ambiente.

O que se espera com este trabalho é a melhora da codificação dos sistemas, principalmente o *WinLivros* (visto na seção 3.2.2) e o *ContabMillenium* (visto na seção 3.2.3). A consequência deste trabalho seria a diminuição de bugs e retrabalhos causados pelos mais diversos motivos, como a falha de

comunicação por exemplo. Outro resultado deste processo poderá ser visto em médio prazo com o aumento da satisfação dos clientes.

A proposta para este trabalho de conclusão de curso é o uso de um estudo de caso para avaliar a proposta de avaliação da adoção de práticas ágeis em um cenário já apresentado. Como usaremos estudos de caso neste trabalho, a seguir temos algumas definições para estes tipos de estudos de caso, segundo Oates (2006):

- Estudo de caso exploratório: é usado para definir as questões ou hipóteses a serem usadas no decorrer do estudo. É usado para ajudar o pesquisador a entender o problema de pesquisa;
- Estudo de caso explanatório: além de analisar detalhadamente um fenômeno particular em seu contexto, tenta explicar porque ele aconteceu e os seus resultados. Procura identificar os múltiplos e inter-relacionados fatores que influenciam nos resultados. Também pode comparar os resultados encontrados com teorias na literatura.

Para este trabalho usaremos um estudo de caso explanatório, onde usaremos uma instância para analisar e observar os resultados e comportamento das práticas sugeridas.

Após a conclusão do estudo de caso serão apresentados os resultados alcançados com a adoção das práticas e métodos ágeis, descrevendo resumidamente a influência deste projeto na equipe e no processo de desenvolvimento de software. Resultados quantitativos não serão levantados pelo fato de o período ser muito curto para que se possa realizar métricas e estimativas de produtividade. Os únicos resultados quantitativos apresentados, dizem respeito ao questionário aplicado à equipe.

5 ESTUDO DE CASO

O estudo de caso representa o resultado de um projeto de aplicação de métodos ágeis em uma equipe de composta por dois núcleos de desenvolvimento da empresa Data Cempro Informática durante o período de 30 dias, entre os meses de outubro e novembro de 2011. Os métodos escolhidos foram baseados nos valores e práticas da XP por se tratar de uma metodologia mais em nível operacional, com o foco no desenvolvimento de software. A aplicação de práticas do *Scrum* foi descartada pelos motivos citados no capítulo anterior e pelo fato desta ser uma metodologia com enfoque mais gerencial, o que dificultaria a participação do autor que executa na equipe o papel de desenvolvedor de software.

O início do processo de implantação dos métodos e práticas ágeis na equipe se deu em meados do mês de setembro por uma reunião inicial, onde foram apresentados os objetivos do projeto, expondo ao grupo tudo o que se pretendia fazer para melhorar o processo atual de desenvolvimento. Para instigar a equipe, procurando obter o engajamento do time, foi passado um resumo sobre as metodologias ágeis, falando um pouco de seu histórico, aplicabilidade, benefícios, etc.

Nesta mesma reunião foi aplicado um questionário na equipe como instrumento de pesquisa, conforme descrito na seção 5.1. Com isso procurou-se saber um pouco mais sobre os integrantes do time, como suas experiências na área de TI, conhecimentos em metodologias ágeis, dentre outras coisas. O resultado desta coleta de dados encontra-se na seção 5.2 do presente artigo.

Através de um acordo envolvendo os *stakeholders*, decidiu-se pela adoção das práticas de XP pela equipe durante um período experimental de 30 dias. Caso os objetivos do trabalho sejam cumpridos e houvesse aceitação da equipe, as práticas poderiam ser mantidas mesmo após o período pré-estabelecido. Do contrário, o projeto seria abortado. Nas próximas seções serão apresentadas situações específicas do projeto, selecionadas para ilustrar o resultado da aplicação da *Extreme Programming* na equipe.

Após a reunião, ficou definido que seria utilizado pelo time de forma experimental durante 30 dias um processo híbrido com as melhores práticas ágeis da XP, de acordo com a realidade e aceitação da equipe. Lembrando que, conforme dito por Teles (2007), a essência das metodologias ágeis está focada nos princípios e valores. As práticas são adaptadas para uma realidade específica.

5.1 Instrumento de pesquisa

Esta etapa serviu de base para o estudo de caso, pois com isso o leitor pode ter uma visão geral da equipe envolvida neste projeto. O questionário foi dividido em duas etapas, contendo questões gerais e questões sobre desenvolvimento de software e foi composto de 14 questões, conforme descrito a seguir:

5.1.1 Questões gerais

1) Nome: _____

2) Sexo: ( ) Masculino ( ) Feminino

3) Idade:
   a. Menor de 20 anos
   b. 20 a 25 anos
   c. 26 a 30 anos
   d. 31 a 40 anos
   e. Mais de 40 anos

4) Escolaridade:
   a. Superior completo
   b. Superior incompleto
   c. Ensino técnico completo
   d. Ensino médio completo

5) Experiência profissional na área de TI:
   a. Mais de 5 anos
   b. 3 a 5 anos
   c. 1 a 3 anos
   d. Menos de 1 ano

6) Quando você começou a trabalhar na Data Cempro Informática (mês/ano)? __/____

5.1.2 Questões sobre desenvolvimento de software

7) Qual sua função na equipe?
   a. Gerente de desenvolvimento
   b. Desenvolvedor (a)
   c. Técnico (a) em qualidade de software
   d. Técnico (a) em suporte

8) Sobre metodologias de desenvolvimento, assinale as que você conhece:
   a. Metodologia tradicional (Modelo Cascata)
   b. Scrum
   c. *Extreme Programming* (XP)
   d. *Crystal Method*
   e. *Feature Driven Development* (FDD)
   f. *Test Driven Development* (TDD)
   g. Outra: _____
   h. Nenhuma

9) Ainda sobre metodologias de desenvolvimento, assinale as que você já trabalhou:
   a. Metodologia tradicional (Modelo Cascata)
   b. Scrum
   c. *Extreme Programming* (XP)
   d. *Crystal Method*
   e. *Feature Driven Development* (FDD)
   f. *Test Driven Development* (TDD)
   g. Outra: _____
   h. Nenhuma

10) Se você pudesse selecionar algum tipo de metodologia para um novo projeto, qual seria?
   a. Metodologia tradicional (Modelo Cascata)

b. Metodologia ágil
c. Metodologia híbrida

11) Quando você precisa se comunicar com outro membro da equipe, qual meio de comunicação você utiliza preferencialmente?
    a. Telefone
    b. Mensagem instantânea (MSN ou *Send*[15])
    c. E-mail
    d. Pessoalmente

12) Você utiliza algum tipo de documentação de processos para auxiliar no seu trabalho?
    a. Sim, sempre.
    b. Sim, às vezes.
    c. Raramente.
    d. Nunca.

13) Você tem a colaboração direta do cliente durante a execução de seu trabalho?
    a. Sim, sempre.
    b. Sim, às vezes.
    c. Raramente.
    d. Nunca.

14) Como você avalia a produtividade da sua equipe de trabalho?
    a. Ótima, sempre atingimos as metas.
    b. Boa, normalmente atingimos as metas.
    c. Regular, mas pode melhorar.
    d. Ruim, precisamos rever os processos.

Com base nas informações coletadas com o *survey*[16] acima, pode-se identificar os elementos que compõem o ambiente e a equipe de trabalho utilizada como estudo de caso. Somente após avaliar a população afetada foi possível delinear as melhores práticas a serem adotadas e propor um plano de ação. Na seção seguinte é apresentada uma breve análise das informações coletadas.

5.2 Análise dos dados coletados

- 80% da equipe possui entre 31 e 40 anos;
- Somente 50% da equipe cursou ou está cursando o ensino superior;
- 90% do grupo possui mais 5 anos de experiência na área de TI;
- Todos os membros da equipe conhece pelo menos uma metodologia ágil;
- Apenas 10% da população afetada já trabalhou com *Scrum*;
- A TDD foi a metodologia ágil mais utilizada pelos membros do time, com percentual de 80%;
- Para um novo projeto, 70% dos entrevistados optariam por uma metodologia híbrida;
- A mensagem instantânea foi escolhida de forma unânime como o meio de comunicação mais utilizado pela equipe;
- Somente 30% dos entrevistados conta raramente com a colaboração do cliente e, mesmo assim, o contato não é de forma pessoal;
- A produtividade da equipe é considerada boa por 70% dos entrevistados, pois normalmente atingem as metas, enquanto 30% considera regular, achando que podem melhorar.

5.3 Práticas de XP adotadas no projeto

[15] Software interno de comunicação entre os colaboradores.

[16] Levantamento; um método para coleta de informações quantitativas sobre os itens em uma população.

Com base nas práticas da XP apontadas por Beck (1999) e Jeffries (2001), a equipe optou por adotar a aplicação das mesmas de maneira flexível e gradual, conforme acordado na reunião inaugural. Em razão do impacto que uma mudança cultural deste porte pudesse provocar, se preferiu a adoção das práticas de uma forma gradual, seguindo o princípio *Kaizen* de melhoria contínua (YOSHIMA, 2011).

Cada uma das práticas foi analisada minuciosamente pelo gerente de desenvolvimento acompanhado com o objetivo de avaliar se o resultado seria benéfico, ou não, para a empresa. O resultado da aplicação das práticas de XP é descrito nas seções seguintes, informando o que é cada uma das práticas, como ela foi implantada e os resultados obtidos.

5.3.1 Planejamento

A etapa de planejamento é onde se decide o que deve ser feito e o que pode ser adiado no projeto. Em outras palavras, é durante jogo do planejamento que se determina o escopo da próxima versão combinando prioridades de negócio e estimativas técnicas. Assim sendo, quando a realidade se opuser ao planejamento, o mesmo deve ser atualizado (BECK, 1999).

Esta prática já é utilizada pela equipe e é de responsabilidade do técnico em qualidade de software juntamente com a equipe de suporte definir o escopo da próxima liberação de versão. A voz do diretor da empresa também é de extrema influência neste tipo de decisão. Para tornar este procedimento flexível as requisições são colocadas em ordem de prioridade, o que facilita na hora de remover as tarefas menos prioritárias do projeto.

As reuniões semanais que já ocorrem na equipe envolvendo os dois núcleos (contábil e fiscal) servem para, dentre outras coisas, discutir os prazos e elaborar ou modificar o plano de ação. Em nenhum momento do estudo de caso foram utilizados os cartões com histórias de usuário, pois este recurso foi descartado pelo gerente de desenvolvimento (ver capítulo 4). Mas é válido lembrar que durante as reuniões cada integrante da equipe tem total liberdade de expor suas dificuldades, sugerir melhorias ou novas ideias.

Como o planejamento pode ser considerado um método já utilizado pela equipe, o objetivo que se buscou nesta etapa foi a diminuição do replanejamento, ou seja, alterar o mínimo possível o planejamento estipulado. Durante o período de estudo de caso, houve apenas uma liberação de versão e a lista de prioridades não foi alterada. Com isso conclui-se que o resultado esperado foi alcançado.

5.3.2 Entregas frequentes

"Pode parecer impossível criar boas versões muitas vezes, mas todas as equipes XP estão fazendo isso o tempo todo." (JEFFRIES, 2001). Esta afirmação dita por Ron Jeffries em 2001 é o chamado cenário ideal quando se quer garantir a satisfação do cliente e manter o software em integração contínua. A visão de Beck (1999) sobre entregas frequentes é de que cada versão entregue deve ter o menor tamanho possível, contendo o maior valor para o negócio.

No entanto, no desenvolvimento deste estudo de caso isso se tornou inviável. A ideia de *releases* pequenos não foi posta em prática, pois o número de requisições em execução pelos programadores normalmente é alto e demanda muito tempo de desenvolvimento. As exceções ocorrem devido à uma falha na programação, onde determinado cliente necessita de urgência. Nestes casos, é realizado um clone da última versão liberada e colocado nele somente as alterações necessárias para corrigir a falha em questão.

Pela impossibilidade de aplicação desta prática, não foram definidos os resultados a serem obtidos. Futuramente com um trabalho mais elaborado e com maior tempo de experimento, esta prática pode gerar bons resultados à Data Cempro Informática.

5.3.3 Metáfora

Para Jeffries (2001), a metáfora é uma descrição simples de como o programa funciona, na melhor das hipóteses. Já Kent Beck (1999) define a metáfora como uma simples história compartilhada por todos, servindo de guia para o desenvolvimento, descrevendo como o sistema funciona como um todo.

Neste sentido foi realizado um processo de melhora e clareza nas informações contidas nos textos das requisições no sistema de suporte. A ideia foi facilitar o entendimento e tornar mais claro o que a requisição pede, diminuindo os termos técnicos e utilizando uma linguagem mais informal e melhor entendível por todos os membros da equipe.

Foi feito um trabalho também no sentido de o programador, ao alterar a situação da requisição e

passa-la para teste, orientar o testador através de uma breve descrição no campo de observações, que pouco vinha sendo utilizado. Exemplo: Foi aumentado o tamanho do campo que armazena o valor contábil de uma nota fiscal de serviços. Testar o cadastro com um valor de tamanho excedente.

O objetivo da aplicação desta prática é a melhora na comunicação entre testador e programador, fator este que costuma gerar requisições mal entendidas e até mesmo retrabalhos. O uso de uma linguagem mais simples na descrição das requisições facilita também o entendimento por parte daqueles que não compreendem a equipe técnica, como clientes e até mesmo a direção da empresa.

5.3.4 Projeto simples

"O sistema deve ser projetado da maneira mais simples possível em qualquer momento. A complexidade desnecessária é removida assim que for descoberta." (BECK, 1999). Kent Beck (1999), um dos precursores da XP ainda define que o projeto correto para o software em qualquer momento é aquele que: (1) executa todos os testes; (2) não tem lógica duplicada; (3) expressa todas as intenções importantes para os programadores e (4) tem o menor número possível de classes e métodos.

A ideia de satisfazer os requisitos atuais sem a preocupação com requisitos futuros foi um ponto bastante discutido no decorrer do projeto. Esta prática vai de encontro ao que geralmente se houve: *"Implemente para hoje, projete para o amanhã"*. Convencer a gerência e a equipe a adotar esta mentalidade sugerida por Beck (1999) de *"acrescentar o que for preciso quando você precisar"* não foi uma tarefa das mais fáceis.

Esta preocupação de facilitar o trabalho de amanhã, refatorando sempre que possível, é uma cultura empresarial, portanto não seria de uma hora para outra que isso mudaria. A solução encontrada foi a adoção desta prática seguindo a cultura *Kaizen* da melhoria contínua (YOSHIMA, 2011). A cultura *Kaizen* foca na melhoria do ambiente com pequenas mudanças incrementais e constantes. Com o passar do tempo, este trabalho fortalecerá a cultura da empresa, pois ela compreenderá suas falhas com provas palpáveis que serão a motivação para mudanças.

O resultado desta abordagem só poderá ser observado mais adiante, com esta prática já consolidada na equipe de desenvolvimento. No momento, ficou apenas a satisfação da gerência pelo esforço contínuo da equipe na busca de melhora dos processos atuais.

5.3.5 Testes

Um dos problemas mais recorrentes em projetos de desenvolvimento de software é a incidência de defeitos, o que gera retrabalho para o programador e insatisfação para o cliente. Ninguém pode prever certos problemas em um software, mas a ideia é sempre dificultar para que eles não aconteçam. E como se faz isso em se tratando de codificação de sistemas? Com o desenvolvimento orientado a testes, que é uma das práticas mais utilizadas pelas metodologias ágeis em geral, não só a XP.

Neste estudo de caso existiram alguns fatores que tornaram esta etapa não tão eficiente. A linguagem de programação utilizada, no caso o Delphi, não favorece muito questão dos testes unitários. Neste caso, foi preciso ter jogo de cintura e utilizar-se de alguns recursos que a realidade do ambiente de desenvolvimento proporcionava. Conforme dito no capítulo 3, onde foi descrito o cenário atual, a equipe de desenvolvimento já utiliza a prática da TDD através da *DUnit*, que é um framework de testes unitários inspirado na *JUnit* que foi escrita em Java por Kent Beck e Erich Gamma.

Outro ponto a ser considerado em relação ao Delphi é que, pela linguagem ser procedural, os testes de validação são focados quase que exclusivamente nas regras de negócio do sistema, conforme dito por Daniel Wildt (2007). Por este motivo, a equipe desenvolveu classes de validação para tratar a entrada de dados nos sistemas, como importação de arquivos de terceiros, por exemplo.

No entanto, apesar de todos os contrapontos foi possível observar uma evolução na qualidade dos softwares principalmente nas rotinas de importação, como a da Nota Fiscal Eletrônica (NF-e), de cupons fiscais, de lançamentos gerados por sistemas de terceiros, entre outras. Foi bastante perceptível pelos técnicos em qualidade de software a diminuição do número de requisições referentes à importações com falha de programação.

5.3.6 Refatoração

A XP pede que a refatoração seja feita apenas quando necessário, ou seja, quando o desenvolvedor

percebe que é possível simplificar o módulo atual sem perder a funcionalidade (BECK, 1999). Com alguns anos de prática em desenvolvimento de sistema já é possível perceber que refatorar é uma tarefa que pode ser oportuna, mas ao mesmo tempo arriscada. O que pode ser útil para resolver novos problemas dos usuários pode também fazer com que módulos do sistema deixem de funcionar corretamente.

Uma definição simples e ao mesmo tempo completa diz que *"A refatoração é o processo de fazer mudanças em um código existente e funcional sem alterar seu comportamento externo. Em outras palavras, alterar como ele faz, mas não o que ele faz."* (ASTELS, 2003). Baseado nisso, esta prática foi enfatizada objetivando estabelecer um uso razoável do *refactory*, evitando assim que o programador cometesse excessos.

Um ponto importante desta prática foi a troca de informações entre os membros da equipe sobre técnicas de refatoração. Para exemplificar, em uma das sessões de *pair programming* (assunto abordado na seção seguinte) um dos programadores de uma das duplas comentou sobre um recurso utilizado para a limpeza de código-fonte (desacoplamento) através de um componente de terceiro, o *CnPack*[17]. Após utilizar e se beneficiar desta funcionalidade, o outro programador da dupla tratou de passar um e-mail sobre como fazer uso deste recurso para os demais setores de desenvolvimento da Data Cempro.

Como os sistemas dos dois núcleos atuantes neste estudo de caso não são novos houve momentos deste trabalho dedicados à refatoração. A remoção de classes, métodos e funcionalidades antigas e não mais utilizadas pelos programadores foi realizada em sessões, em dias não específicos. Foram dedicados pelo menos 4 dias para este fim, buscando melhorar a qualidade da codificação dos programas *WinLivros* e *ContabMillenium*.

Como resultado, a compilação dos sistemas citados anteriormente ficou levemente mais rápida devido aos projetos de ambos os softwares não carregarem mais pacotes de projetos antigos e obsoletos. Além disso, foram eliminados diversos pontos de código duplicado, ou seja, duas funções que fazem exatamente a mesma coisa em dois locais diferentes foram centralizadas.

5.3.7 Programação em pares

Existem numerosos estudos que comprovam que o olho humano tem uma tendência a ver o que ele espera ver (WEINBERG, 1971). É muito comum, durante a etapa de codificação, que um programador ignore erros banais como um caractere a mais ou a menos pela simples falta de atenção e pelo seu olho estar "capacitado" a não enxergar o que ele não quer.

Para diminuir essa tendência a erros de programação, a XP desenvolveu uma técnica de programação em pares, onde dois programadores trabalham em um mesmo computador. Enquanto um desenvolvedor está com o controle do teclado e do mouse implementando o código, o outro observa continuamente o trabalho que está sendo feito, procurando identificar erros sintáticos e semânticos e pensando em como melhorar o código que está sendo implementado. Esses papéis podem e devem ser alterados continuamente, possibilitando que um aprenda com o outro (BECK, 1999).

Para adotar na prática a programação em pares neste estudo de caso, foi necessária uma longa negociação com o gerente de desenvolvimento, haja vista que ele se mostrava contrário à adoção desta prática. Com isso, ficou decidido que esta prática seria adotada em sessões, preferencialmente nas sextas-feiras, pois é o dia que se tem por hábito executar tarefas menos complexas e com menor prioridade. O receio por parte da gerência era de que ao trabalhar em duplas, houvesse dispersão por parte da equipe, o que não acabou ocorrendo.

Como o time era composto por sete programadores, o gerente de desenvolvimento também aderiu à prática, formando assim oito duplas. A regra da equipe foi de que não pudessem se repetir as duplas durante as sessões de *pair programming*, que ocorriam em turnos. Assim sendo, se pela manhã duas pessoas formaram um par, à tarde elas faziam duplas com outros colegas. E os dois maiores motivos de entrave para a adoção desta prática não se fizeram presentes em nenhum momento durante o estudo de caso, que seria a dispersão e a perda de foco.

Como resultado, as sessões de programação em pares possibilitaram uma disseminação do conhecimento muito maior, fazendo com que a prática atingisse seu real objetivo. Além disso, houve melhora evidente da codificação em duplas, pois enquanto um programador codificava o outro estava

[17] Fonte: http://www.cnpack.org/.

acompanhando e pensando mais estrategicamente, e corrigindo erros que passariam despercebidos por um único programador.

5.3.8 Propriedade coletiva

Segundo Jeffries, Anderson e Hendrickson (2001), *"todas as classes e métodos pertencem à equipe e qualquer membro da equipe pode melhorar o que for necessário"*. Esta prática da *Extreme Programming* habilita os desenvolvedores a trabalharem em qualquer fragmento do código-fonte, mesmo em partes onde ele nunca trabalhou. Dessa forma, a equipe de trabalho possui uma maior segurança e tranquilidade com relação à produtividade do time, pois *"todo o trabalho fica menos suscetível de ser afetado se alguém ficar doente (...) ou faltar (...). Isso não apenas reduz as variações no cronograma, mas também eleva as chances de ter alguém por perto quando o código tiver que ser modificado."* (WEINBERG, 1971).

Esta talvez tenha sido a prática da XP mais familiarizada pela equipe, que já se utiliza de sistemas controladores de versão de código-fonte, neste caso o *Tortoise SVN*[18]. Este software é muito prático e de fácil manuseio, pois é completamente integrado com o Windows Explorer e também a IDE de desenvolvimento do Delphi. Isso facilita o trabalho e encoraja o programador a codificar sem medo, pois o *Tortoise* permite que sejam executados comandos de *update* (sincronia dos fontes dos sistemas na estação de trabalho), *commit* (sincronia dos fontes no servidor repositório), *revert* (reversão das alterações efetuadas), *diff merge* (comparação de arquivos), entre outros recursos.

Para este estudo de caso, esta prática em si não mostrou nenhum resultado relevante que já não fosse esperado, pois conforme citado ela já era utilizada pela equipe.

5.3.9 Integração contínua

A prática da integração contínua é defendida por Kent Beck (1999) em virtude do problema causado pelo código-fonte defasado com relação aos demais programadores, isto é, os programadores de uma equipe de desenvolvimento necessitam ser capazes de evoluir rapidamente sem interferir no trabalho uns dos outros. A integração contínua surgiu para eliminar este tipo de problema, pois através dela os programadores integram o que produzem com a versão mais atualizada do código de produção, quantas vezes forem necessárias ao dia.

Trazendo esta situação para o estudo de caso, este tipo de problema não ocorreu em nenhum momento durante estes 30 dias de experimento pelo fato de os programadores sincronizarem frequentemente suas estações de trabalho através do comando *update* do *Tortoise SVN*, conforme citado na seção 5.3.8. O que aconteceu com alguma frequência durante a sincronia de fontes localmente foi de algum arquivo atualizado por outro programador conflitar com alguma alteração em produção pelo programador que estava sincronizando seu ambiente de trabalho. Neste caso, o versionador de código-fonte facilitou com o recurso de edição de conflitos.

A integração contínua resultou em uma boa demanda de modificações nos sistemas implementadas paralelamente. Neste caso, o gerenciador de versões de código-fonte foi bastante utilizado e atendeu com êxito as necessidades da equipe.

5.3.10 Ritmo sustentável

Em suma, os objetivos da XP são obter a maior produtividade da equipe de desenvolvimento, entregar software da melhor qualidade e obter, como consequência disso, a satisfação do cliente. Porém, para que isso aconteça a XP também mantém uma preocupação constante com o cronograma. O bom rendimento da equipe depende de um ritmo sustentável, de até 40 horas semanais. Eventuais horas extras são admitidas em XP apenas quando a produtividade é maximizada ao longo do prazo (BECK, 1999).

Durante este estudo de caso não se abriu mão de cumprir a carga horária de 44 horas semanais de trabalho, que é padrão da empresa. No entanto, durante o período dos 30 dias de experimento, não houve a necessidade de que nenhum programador da equipe fizesse hora extra. Este tipo de situação só acontece em situações extremas, às vésperas de uma liberação de versão de sistema, por exemplo, quando se faz necessário. O horário de trabalho na Data Cempro Informática vigora das 8h30 ao meio-dia e das 13h30 às 18h15. Ainda assim, a empresa oferece três pausas opcionais de 15 minutos durante o expediente: duas para lanche (manhã e tarde) e uma para ginástica laboral, no início da tarde.

[18] Fonte: http://tortoisesvn.net/about.html.

Com base nisso, conclui-se que a equipe atendeu com êxito as expectativas desta prática, pois foi possível trabalhar em um ritmo saudável durante o período do estudo de caso. Das 10 pessoas envolvidas no projeto, nenhuma fez hora extra neste período além de pelo menos 7 destas 10 pessoas frequentarem assiduamente as sessões de ginástica laboral.

5.3.11 Cliente presente

Segundo a definição desta prática por Kent Beck (1999), um cliente real deve ficar com o time de desenvolvimento, permanecer disponível para responder questões, resolver disputas e definir algumas prioridades. Em outras palavras, a XP defende que o cliente precisa estar fisicamente disponível, ao lado do desenvolvedor enquanto ele codifica a produção do sistema. Esta prática tem como sua maior objeção o fato de que usuários reais do sistema em desenvolvimento são muito valiosos para serem emprestados à equipe.

Pelo problema citado, esta talvez tenha sido a prática com maior dificuldade de se adotar pela equipe, tanto que ela foi descartada. Existe uma regra na Data Cempro de que o contato com o cliente deve ser feito preferencialmente pela equipe de suporte, evitando ao máximo o contato entre programador e cliente. Salvo raras exceções, é possível que haja a troca de informações via e-mail, telefone ou mensagens instantâneas (MSN, Skype, etc.), porém este tipo de contato é coibido pela diretoria da empresa.

Este impedimento de se colocar a prática do cliente presente em ação é o típico caso de cultura empresarial. Talvez com o passar do tempo e uma maior aceitação por parte dos gestores, principalmente, a XP possa vir a ser adotada integralmente. No entanto, para este estudo de caso o resultado desta prática foi nulo.

5.3.12 Código padrão

Segundo Ron Jeffries (2001), equipes XP seguem um padrão comum de codificação, de modo que todo o código no sistema olha como se fosse escrito por um único indivíduo. As especificidades do padrão não são importantes, o que é importante é que todo o código parece familiar, em apoio à propriedade. A ideia é que o padrão adotado exija a menor quantidade de trabalho possível e seja consistente com a regra de evitar código duplicado.

A priori, a prática da padronização do código-fonte já é habitual pelo time de desenvolvimento da Data Cempro. Todo desenvolvedor iniciante na empresa é orientado a codificar seguindo as boas práticas de programação através de documentos de auxílio disponibilizados na intranet. A programação orientada a objetos também é exigida, pois ela dá sustento para modificações futuras evitando a duplicidade de código e mantendo a consistência e a clareza do mesmo.

Outro fator importante abordado neste período de estudo de caso foi a questão de reforçar o uso da biblioteca de componentes próprios da Data Cempro. Sempre que um programador encontrasse em alguma parte do código-fonte a possibilidade de fazer a troca de um componente nativo do Delphi por um próprio, deveria fazê-lo. Neste intervalo de 30 dias foi desenvolvido por um dos programadores um novo *frame* para busca de clientes através do código, documento (CPF/CNPJ) ou do nome. A vantagem deste *frame* é que toda correção ou melhoria implementada no *frame* beneficiará todas as telas que o utilizarem.

O engajamento da equipe na questão da padronização do código-fonte dos sistemas resultou em uma melhora significativa na qualidade dos mesmos. Dos 10 membros do time, 2 são programadores novatos na empresa, ainda sob o contrato de experiência. No entanto, foi esta ênfase em padronização que facilitou o entendimento e a familiarização com o código por parte deles.

6 RESULTADOS OBTIDOS

De acordo com o que foi descrito no capítulo anterior, foi possível constatar que o saldo da aplicação das práticas de XP foi positivo. Das doze práticas de XP aplicadas pela equipe, duas não puderam ser aplicadas pelos motivos já expostos: entregas frequentes e cliente presente.

No entanto, algumas práticas aplicadas apresentaram resultados extremamente satisfatórios, como por exemplo, a programação em pares, que possibilitou uma melhor disseminação do conhecimento e interatividade pela equipe. O trabalho de refatoração também foi de grande valia para a melhoria dos sistemas desenvolvidos pelos núcleos da empresa envolvidos no projeto.

As práticas de planejamento, projeto simples e testes agregaram valor em situações pontuais para

qualquer equipe de desenvolvimento de software como replanejamento, retrabalho. A comunicação entre testadores e programadores foi melhorada com a aplicação de metáforas na descrição das requisições. Já as práticas de propriedade coletiva, ritmo sustentável, integração contínua e padrões de codificação acrescentaram o suficiente para manter o padrão da equipe, pois o setor de desenvolvimento da Data Cempro já adotava estas práticas, mesmo antes de aderir a *Extreme Programming* neste estudo de caso.

Ao término do período experimental de aplicação das práticas da XP, foi realizada uma entrevista informal com cada participante do estudo de caso. O objetivo era descobrir de que maneira este trabalho afetou a equipe e a produtividade do time. Além disso, foi feito uma entrevista no intuito de saber da equipe quem aprovaria a continuidade da adoção de práticas XP no trabalho.

Segundo as próprias palavras do gerente de desenvolvimento da equipe, *"Neste período de 30 dias pude observar uma maior integração por parte da equipe, sem perder o foco no objetivo do time. (...) Inicialmente tive receio de que o trabalho de programação em dupla pudesse causar dispersão, mas do meu ponto de vista o saldo foi bastante positivo."*.

Os programadores com experiência em métodos ágeis aprovaram o estudo de caso. Disse um deles: *"Nunca tinha utilizado a XP na prática antes, apenas o Scrum, mas confesso que as práticas utilizadas neste trabalho foram muito importantes para uma série de pequenos problemas que a equipe possuía, como a falha de comunicação, por exemplo."*.

No entanto, houve algumas opiniões controversas, principalmente por aqueles que já estavam extremamente adaptados ao processo atual de desenvolvimento da equipe de desenvolvimento. Um programador com quase 10 anos de empresa relatou: *"Pra mim foi uma novidade trabalhar com XP. Apesar de ser considerada uma metodologia ágil, penso que meu rendimento é bem melhor seguindo o modo de desenvolvimento atual, pois já estou acostumado. Só o tempo pra mostrar se realmente vale a pena essa mudança de processo."*.

Os dois técnicos em qualidade de software, mesmo tendo participação menos ativa no estudo de caso em relação aos programadores, se mostraram indiferentes para a maioria das práticas, pois os resultados não puderam ser efetivamente mensurados. Porém, um dos testadores ressaltou: *"O uso de metáforas na descrição das requisições melhorou o entendimento dos problemas, pois muitos termos utilizados pelos programadores eram muito técnicos."*.

Em se tratando de produtividade, pode-se notar um decréscimo no número de requisições liberadas em comparação aos últimos *releases*, talvez pelo fato de quase todos os programadores estarem envolvidos com requisições complexas durante este estudo de caso. Além disso, pesou também o fato de o período experimental ser muito curto para se realizar uma melhor análise dos efeitos que as práticas ágeis podem causar.

Por outro lado, foi possível notar através do *feedback* do grupo, passado na reunião após o término do experimento, que o saldo foi positivo no que diz respeito à aceitação da XP. Houve unanimidade quando foi questionado se o trabalho de adoção das práticas ágeis poderia ter continuidade. Pode-se destacar a disseminação de conhecimento provocada pelas sessões de programação em pares como o item que mais agradou aos programadores, em especialmente. Para os técnicos em qualidade de software, o projeto foi benéfico por aumentar a comunicação face a face e pelo melhor entendimento de comunicação na relação testador-programador.

Com relação à satisfação do cliente, só poderá ser medida após o período de utilização das novas funcionalidades, melhorias e correções realizadas durante o estudo de caso. No entanto, vale ressaltar que os sistemas *WinLivros* e *ContabMillenium* apresentaram uma melhora elevada na codificação, além de uma diminuição relevante de bugs. Isso acabou facilitando o trabalho dos técnicos em qualidade de software que despenderam menos tempo envolvidos com testes.

7 CONSIDERAÇÕES FINAIS

Esta monografia realizou uma análise de problemas comuns em projetos de software de acordo com as pesquisas, tais como atrasos nas entregas, orçamentos extrapolados e funcionalidades que não solucionam o problema dos usuários. Com isso, o objetivo do trabalho foi propor a adoção de práticas ágeis, procurando uma alternativa viável para a resolução destes problemas presentes em diversos projetos de software.

Para fundamentar a adoção dos métodos abordados, procurou-se estabelecer uma significativa base teórica na forma de revisão da literatura. O intuito com este estudo foi de solidificar as razões pelas quais a adoção do desenvolvimento ágil de software pudesse funcionar de fato na busca da melhoria contínua do processo de desenvolvimento de software.

Este artigo também teve o importante papel de realizar um estudo de caso abordando as melhores práticas da metodologia *Extreme Programming* em uma equipe composta por 10 pessoas, contendo 7 programadores, 2 testadores e 1 gerente de desenvolvimento. O estudo de caso foi adotado pelo período experimental de 30 dias e, com base nos resultados obtidos, foi possível constatar que houve um resultado qualitativo bastante positivo.

Com base neste projeto, foi possível disseminar o conhecimento entre a equipe, instigando assim a equipe a buscar novas técnicas ágeis a fim de melhorar a qualidade do software desenvolvido. Entende-se que este é apenas o início de um projeto que visa alcançar resultados mais relevantes futuramente, buscando sempre a melhoria contínua no processo de desenvolvimento de software.

AGRADECIMENTOS

Primeiramente à minha melhor amiga e esposa, Mariane Pereira Hickmann, por todo seu apoio incondicional, companheirismo, incentivo, compreensão e paciência para comigo durante o período de elaboração deste trabalho. Por todo seu amor e carinho que estão sempre presentes em nosso dia-a-dia, cultivando todas as condições para que eu chegasse até aqui.

Ao meu melhor amigo e pai, Cleo Hickmann, por depositar confiança em mim acreditando no meu potencial e me apoiar financeiramente inclusive, por se mostrar disponível para eventuais problemas que eu tivesse em todo meu período acadêmico, além de ser um exemplo de superação pra mim.

Ao meu orientador, Prof. Anderson Yanzer, por sugerir e apoiar o tema desta monografia, pelas dicas de leitura, material indicado, críticas, pela sua disponibilidade e pelas observações.

Ao professor Christiano Cadoná por se manter sempre disponível a sanar todas as dúvidas, pela sua disposição e espontaneidade nos encontros aos sábados com os alunos concluintes.

Aos meus colegas de equipe dos núcleos contábil e fiscal da Data Cempro Informática que aceitaram e concederam a oportunidade de implantação dos métodos e práticas ágeis sugeridas por mim.

À Universidade Luterana do Brasil, englobando o campus de Guaíba e de Canoas, por me proporcionar um ótimo ambiente de estudo fazendo com que eu evoluísse no campo estudantil, profissional e pessoal.

REFERÊNCIAS

AMBLER, Scott. **Modelagem Ágil: Práticas Eficazes para a Programação Extrema e o Processo Unificado**. Bookman, 2004.

ASTELS, David. **Test-driven development: a practical guide**. 1. ed. Upper Saddle River, NJ: Prentice Hall PTR, 2003.

BECK, Kent *et al*. **Agile Manifesto.** Disponível em: <http://agilemanifesto.org>. Acesso em: 13 abr. 2011.

BECK, Kent. **Programação Extrema Explicada**. Boston, Bookman, 1999.

BECK, Kent. **Test-Driven Development by Example**. Addison-Wesley, 2003.

COCKBURN, Alistair. Agile software development. 1ª ed. Boston: Addison-Wesley, 2002.

DATA CEMPRO INFORMÁTICA LTDA. **ContabMillenium**. Data Cempro Informática Ltda. Disponível em: <http://www.datacempro.com.br/contabmillenium>. Acesso em: 05 nov. 2011.

DATA CEMPRO INFORMÁTICA LTDA. **WinLivros**. Data Cempro Informática Ltda. Disponível em: <http://www.datacempro.com.br/winlivros>. Acesso em: 05 nov. 2011.

FOWLER, Martin. **Refactoring: Improving the Design of Existing Code**. Addison-Wesley, 2000.

FOWLER, Martin. **The New Methodology**, 2005. Disponível em: <http://www.martinfowler.com/articles/newMethodology.html#N8B>. Acesso em: 22 mai. 2011.

HIGHSMITH, Jim. **Agile Project Management**, 2002.

JACOBSON, Ivar. **A Resounding "Yes" to Agile Processes – But Also More**. Cutter IT Journal, 2002.

JEFFRIES, Ron. **What is Extreme Programming?** 2001. Disponível em: <http://xprogramming.com/what-is-extreme-programming>. Acesso em: 07 nov. 2011.

JEFFRIES, Ron; ANDERSON, Ann; HENDRICKSON, Chet. **Extreme Programming installed**. 1ª ed. Boston: Addison-Wesley, 2001.

JOHNSON, Jim. **ROI, It's your job**. Published Keynote Third International Conference on Extreme Programming. Alghero, Itália, 2002.

MADEIRA, Fernando. **Métodos Ágeis**. Dev Day, Sisnema, 2011.

MARTIN, Robert C. **Clean Code: A Handbook of Agile Software Crafsmanship**. Prentice Hall, 2009.

OATES, Briony J. **Researching Information Systems and Computing**. London: Sage Publications, 2006.

PRESSMAN, Roger S. **Engenharia de Software**. 6ª ed. McGraw-Hill, 2006.

SOMMERVILLE, Ian. **Software Engineering**. 6 ª ed. Addison-Wesley, 2000.

STAVARENGO, Rafael. **Desenvolvimento ágil.** Revista Clube Delphi. Ano 9, 125ª ed. DevMedia, 2011.

TELES, Vinícius M.; WILDT, Daniel. **Entrevista com Daniel Wildt na Série Experiências Ágeis**. 18 jul. 2007. Disponível em: <http://improveit.com.br/podcast/improvecast-15-entrevista-daniel-wildt-experiencias-ageis>. Acesso em: 31 out. 2011.

THE STANDISH GROUP INTERNATIONAL. **Extreme Chaos**. The Standish Group International, Inc, 2001. Disponível em: <www.standishgroup.com/sample_research/showfile.php?File=extreme_chaos.pdf>. Acesso em: 03 nov. 2011.

YOSHIMA, Rodrigo. **O mito da Cultura Ágil**. 03 abr. 2011. Disponível em: <http://blog.aspercom.com.br/2011/04/03/o-mito-da-cultura-agil>. Acesso em: 30 mai. 2011.

YOSHIMA, Rodrigo. **Kaikaku – Kaizen**. 09 set. 2011. Disponível em: <http://blog.aspercom.com.br/2011/09/09/kaikaku-kaizen>. Acesso em: 05 out. 2011.

WEINBERG, Gerald M. **The psychology of computer programming**. 1ª ed. New York: Van Nostrand Heinhold Company, 1971.

WIKIPEDIA. **Desenvolvimento ágil de software**. Disponível em: <http://pt.wikipedia.org/wiki/Desenvolvimento_ágil_de_software>. Acesso em: 13 abr. 2011.