Comparação e Avaliação entre o Processo RUP de Desenvolvimento de Software e a Metodologia Extreme Programming

Marcos Leandro Nonemacher
marcos@cisc.com.br
Universidade Federal de Santa Catarina – Curso de Pós Graduação em Ciência da Computação

Anita Maria da Rocha Fernandes
anita@inf.univali.br
Curso de Ciência da Computação – Centro de Ciências Tecnológicas, da Terra e do Mar – CTTMar – Universidade do Vale do Itajaí – UNIVALI

Resumo

O objetivo deste artigo é comparar e avaliar a metodologia Extreme Programming (XP) e o processo RUP de desenvolvimento de software. Tanto XP quanto RUP são metodologias recentes, a primeira no entanto, faz parte de um grupo de metodologias ditas como ágeis, já a segunda é o refinamento de outras metodologias orientadas a objetos, que contempla dentre as suas inúmeras técnicas e práticas, a criação de modelos e diagramas. Uma série destas práticas são contraditórias entre estas duas metodologias, devido a isto tem-se a motivação deste trabalho em aplicar as metodologias na criação de um produto de software, com os mesmos requisitos e escopo de projeto para que pudesse ser comparado o resultado final. Como conclusão da pesquisa, identificou-se que a XP obteve um custo benefício menor, tendo como base os esforços de tempo necessários para o desenvolvimento e testes foram praticamente os mesmos e produziram um resultado final de menor qualidade. Os dados resultantes da pesquisa levam a concluir que a utilização da metodologia RUP seria mais adequada, levando-se em consideração que o objetivo era saber qual das duas metodologias proporciona o desenvolvimento de um software de melhor qualidade, com o menor custo possível.

Palavras Chave: Extreme Programming, processo RUP, desenvolvimento de software.

Abstract

The goal of this paper is to compare and evaluate the Extreme Programming Methodology – XP and RUP software development process. XP and RUP are recent methodologies. The first is in a group of agile methodologies, and the second is the refinement of the other object oriented technologies, and contemplates among its many techiniques and pratices, the criation of models and diagrams. A serie of this pratices are contradictories between the two methodologies, so this is the motivation of this research, in order to apply the methodologies in the creation of a software, with the same projects requisit and scope to compare the final result. The conclusion of this research identified that XP had a lower rate of cost and benefit, having with sustenance the fact that the efforts needed to development and tests were just the same and gave a final result with low quality. The final data od this research concluded that the use of RUP technology would be the best, considering that the goal was to know what of the two technologies give a best software development, with a low cost and benefit.

Key Words: Extreme Programming, RUP process, software development.

1. Introdução

A rápida evolução e abrangência dos sistemas de informação no decorrer dos últimos 40 anos, confirmam a importância dos mesmos como sendo a alavanca das mudanças significativas no modelo de negócio das empresas, proporcionando para as organizações a obtenção de vantagens competitivas e estratégicas.

O software, por ser um dos componentes de maior importância dos sistemas de informação, vem sendo alvo de constantes estudos objetivando ter um desenvolvimento de maneira previsível e em determinado período, com utilização eficiente e eficaz dos recursos.

Um fator chave para se alcançar estes objetivos é a adoção de uma metodologia para o desenvolvimento de software. Uma metodologia de desenvolvimento é um conjunto de regras e padrões que orientam a abordagem utilizada em todas as tarefas do ciclo de

desenvolvimento, prescrevendo ferramentas e até como elas devem ser utilizadas.

Inúmeras metodologias já surgiram, e certamente novas serão desenvolvidas. Algumas são mais utilizadas e conhecidas do que outras, porém o emprego das metodologias mais conhecidas pode trazer alguns benefícios para as empresas, mas não garante necessariamente que por serem mais difundidas sejam as mais eficientes e eficazes.

A metodologia de desenvolvimento de software impõe um processo disciplinado sobre o desenvolvimento do mesmo com o objetivo de fazê-lo mais previsível e eficiente. Algumas fazem isso através do desenvolvimento de um processo detalhado, com uma forte ênfase no planejamento, inspiradas por áreas da engenharia. A crítica sobre elas é que são muito burocráticas, sendo necessária a produção de muito material para ser possível seguí-las. Devido a isto, frequentemente são chamadas de "peso-pesado" ou metodologias monumentais.

Em reação às "peso-pesado", surgiu nos últimos anos um novo grupo de metodologias, denominadas por um tempo de "peso-leve", mas agora conhecidas por "metodologias ágeis". Diferentemente das "peso-pesado", as metodologias peso-leve providenciam apenas o processo suficiente para se obter um retorno adequado; são menos burocráticas e menos formais, no entanto, essa informalidade poderá ocasionar situações onde a falta de uniformidade esteja mais intrínseca.

Como conseqüência desta contradição de idéias abre-se espaço para pesquisas com o intuito de amadurecer e refinar o que cada metodologia apresenta de positivo. O objeto de estudo deste trabalho será comparar as metodologias RUP (*rational unified process*) e XP (*Extreme programming*).

A importância deste estudo está no fato de contribuir para a comunidade de desenvolvimento de software, com o aperfeiçoamento das técnicas utilizadas para realização dos testes de software, conseqüentemente fornecendo ao usuário final um produto de melhor qualidade.

O objetivo principal deste trabalho é comparar a metodologia de desenvolvimento ágil, XP – *Extreme Programming* com uma metodologia do grupo tradicional, RUP – *Rational Unified Process* e identificar qual delas produzirá um software de maior eficácia em relação à qualidade e o esforço de tempo necessário, dentro de um determinado escopo restrito para a pesquisa.

2. Fundamentação Teórica

2.1 Metodologias Ágeis

Em fevereiro de 2001 reuniram-se em Utah, EUA, representantes das metodologias *Extreme Programming*,

SCRUM, DSDM, *Adaptive Software Development*, *Crystal*, *Feature Driven Development*, *Pragmatic Programming*, e outros simpatizantes, que até então eram chamadas de metodologias de "peso leve". O objetivo destas pessoas era buscar caminhos alternativos para os processos tradicionais das metodologias "peso pesado" (HIGHSMITH, 2001).

O resultado deste encontro foi chamado de *Manifesto for Agile Software Development*, cujo propósito era "descobrir maneiras de desenvolver software fazendo e ajudando outros a fazer". Os valores enfatizados no manifesto foram: (i) indivíduos e interações são mais importantes do que processos e ferramentas; (ii) software funcionado é mais importante do que documentação compreensiva; (iii) colaboração do cliente é mais importante do que a negociação do contrato; (iv) reagir a mudanças é mais importante do que seguir um plano.

Ainda de acordo com Highsmith (2001), o movimento ágil não é uma anti-metodologia, mas sim tem o objetivo de restaurar a credibilidade da palavra metodologia, estabelecendo um equilíbrio. Adotar a modelagem, mas não apenas para arquivar diagramas em um repositório empoeirado; utilizar-se de documentação, mas não desperdiçar centenas de páginas de papel que nunca são lembradas e raramente atualizadas. Planejar, mas também reconhecer os limites do planejamento em um ambiente turbulento.

2.2 Metodologia eXtreme Programming (XP)

Extreme Programming (XP) é uma metodologia que se encaixa no grupo das metodologias ágeis para desenvolvimento de software. XP usa times integrados de programadores, clientes e gerentes para desenvolver software de alta qualidade em velocidade alta. Reúne também um conjunto de práticas de desenvolvimento de software já testadas, que estimulada a sinergia entre elas gerará vasta melhoria em produtividade global e satisfação do cliente (HAYES, 2001).

Para Beck (2000), criador da *Extreme Programming*, XP é uma metodologia ágil para equipes pequenas e médias desenvolvendo software com requisitos vagos e em constante mudança.

Extreme Programming segundo Jefries (2002) é uma disciplina de desenvolvimento de software baseada nos valores simplicidade, comunicação, realimentação e coragem. Trabalha reunindo o time inteiro na presença de práticas simples, através do feedback permite a equipe ver onde ela está e afinar as práticas para situações únicas.

De acordo com Fowler (2000) em seu artigo The New Methodology, de todas as metodologias "peso-leve", *Extreme*

Programming é que tem tido a maior atenção. Em parte isto é devido ao fato de que os líderes de XP possuem uma habilidade notável, atraindo uma atenção em particular. E também pela capacidade de Kent Beck em atrair o interesse das pessoas pelo assunto.

A XP é uma disciplina de desenvolvimento de software baseada nos seguintes valores: comunicação, simplicidade, feedback e coragem. Ela também prega uma dúzia de práticas que projetos de XP devem seguir. Muitas destas práticas são antigas, técnicas tentadas e já testadas, ainda que esquecidas por muitos, são incluídas na maioria dos processos planejados. Ressuscitando estas técnicas, XP tece uma sinergia entre ela, onde cada uma é reforçada pela outra (FOWLER, 2000).

2.3. Práticas de eXtreme Programming

As doze práticas promovidas pela XP se forem examinadas individualmente apresentarão falhas, mas uma das forças da XP é que as práticas se combinam de um modo mútuo apoiando-se. Juntas as práticas conduzem a um complexo comportamento emergente. Cada prática tem sua função para manter o custo de mudança baixo (HAYES, 2001).

De acordo com Beck (2000), as doze práticas da XP são mapeadas em três categorias, a primeira englobando as práticas de programação, a segunda englobando as práticas orientadas para a equipe e a terceira contemplando os processos através dos quais a equipe de programação relaciona-se com o cliente. As práticas estão divididas na seguinte forma: (i) *Programming*: projeto simples, testes, reconstrução e código padrão; (ii) *team*: código coletivo, integração contínua, metáfora, código padrão, 40 horas por semana, programação em par, versões pequenas; (iii) *Process*: cliente no local, testes, versões pequenas, planejamento do jogo.

2.4 Metodologia Rational Unified Process (RUP)

No decorrer da década de 90, o paradigma da orientação a objeto ganhou força tornando-se mais evidente suas vantagens dentro da engenharia de software. Diante deste cenário começaram a proliferar inúmeras metodologias baseadas neste conceito, revivendo as experiências negativas que já haviam ocorrido com as metodologias estruturadas. Devido a este motivo, ganhou força a idéia de uma metodologia de desenvolvimento de software unificada, que se aproveitasse da experiência e conceitos da linguagem UML. Dentre as metodologias baseadas neste paradigma está o RUP (*rational unified process*) (SILVA e VIDEIRA, 2001; BOOCH, 2000).

RUP surgiu há pouco tempo, mais precisamente em 1998, apesar disso, contempla em suas origens idéias e experiências vividas nos últimos trinta anos, em especial abordagens seguidas na Ericson onde trabalhou Ivar Jacobson até 1987. Na seqüência, Jacobson fundou a empresa Objectory AB, a qual foi adquirida pela Rational em 1995. A Rational por sua vez vinha desenvolvendo desde 1981 um conjunto de iniciativas com o objetivo de criar um ambiente interativo que permitisse aumentar a produtividade do desenvolvimento.Tinha como um dos seus principais mentores, Philippe Kruchten, mas foi com a entrada de Grady Booch, James Rumbaugh e Ivar Jacobson, que começaram as iniciativas para unificação das metodologias, cujo resultado foi o *Rational Objectory Process*, que a partir de 1998 passou-se a chamar *Rational Unified Process*.

O *Rational Unified Process* é um processo de engenharia de software, que procura disciplinar as atribuições de tarefas e responsabilidades dentro de uma estrutura de desenvolvimento de software. Sua meta principal é garantir a produção de software com alta qualidade satisfazendo as necessidades dos seus usuários, dentro de um cronograma e orçamento possível (RATIONAL SOFTWARE COOPORATION, 2002).

O RUP também aumenta a produtividade da equipe provendo fácil acesso a bases de conhecimento com diretrizes e modelos para atividades de desenvolvimento críticas, proporcionando que todos os membros do desenvolvimento tenham acesso a mesma base de conhecimento. Desta forma assegura-se que todos na equipe compartilham processos e um idioma comum, tendo assim a mesma visão do software a ser desenvolvido (RATIONAL SOFTWARE COORPORATION, 2002).

Uma característica das atividades desta metodologia é de criar e manter modelos, em vez de focar a produção de grande quantidade de documentos, procura enfatizar o desenvolvimento e manutenção de modelos ricos na representação semântica.

O RUP é um processo configurável, pois um único processo não é satisfatório para o desenvolvimento de software. O processo unificado ajusta-se tanto para pequenas equipes de desenvolvimento quanto para grandes organizações. O Processo Unificado é fundamentado em uma arquitetura de processo simples e claro, e ainda pode ser adaptado para acomodar situações diferentes, dependendo da necessidade da organização. Possui várias práticas que integradas completam sua estrutura de atuação.

O *Rational Unified Process*, reúne muitas das melhores práticas em desenvolvimento de software moderno e coloca a disposição dos projetos e organizações: (i) desenvolvimento interativo de software; (ii) requerimentos de gerenciamento; (iii) arquitetura baseada em componentes; (iv) visualização do modelo de software; (v) verificação da qualidade de software; (vi) controle de mudanças para o software.

A arquitetura RUP apresenta-se dividida em duas dimensões: a primeira demonstra o tempo, os aspectos dinâmicos do processo e como ele é ordenado. É representada em termos de ciclos, fases, interações e marcos (*milestones*). A segunda dimensão representa os aspectos estatísticos do processo, descreve os termos das atividades, artefatos, participantes do processo e o fluxo de trabalho (*workflow*) (SILVA e VIDEIRA, 2001).

3. Metodologia

Após a revisão literária realizada em livros, artigos e material disponível na Internet, foram apresentados os objetivos da pesquisa a um grupo de desenvolvedores e testadores de software, os quais foram convidados a participar do desenvolvimento de um software utilizando as metodologias XP e RUP.

Essa equipe foi composta por oito pessoas, destes, seis têm formação acadêmica em Ciência da Computação ou Sistemas de Informação, um em Administração e o último em Economia. Foram revisados juntamente com a equipe os conceitos das metodologias, para que todos os envolvidos tivessem o domínio das técnicas a serem utilizadas no projeto.

Alguns membros da equipe foram divididos em dois grupos, o primeiro com a responsabilidade de desenvolver um componente de software utilizando XP e outro desenvolvedor com a responsabilidade de desenvolver um componente utilizando RUP. O restante da equipe realizou tarefas comuns aos dois grupos.

Após a equipe estar montada, os papéis e as responsabilidades definidas, infra-estrutura de hardware software disponível, foram repassados aos engenheiros de software as regras de negócio, modelo de dados e os requisitos primários. É importante ressaltar que ambos os grupos, XP e RUP, partiram do mesmo ponto em comum.

3.1 Modelagem do Negócio

As empresas de uma maneira geral trabalham com a política de comissionamento de suas equipes de vendas. Desta forma, se faz necessário um controle das operações de débito, crédito e também o saldo destas comissões.

Este módulo que foi desenvolvido tem a finalidade de permitir ao usuário,

lançar valores de comissão para os vendedores cadastrados no sistema. Considerou-se que já existia no sistema o cadastramento dos vendedores.

Após o recebimento deste relatório, as equipes de XP e RUP partiram para próximas etapas do ciclo de desenvolvimento. O ciclo completo do desenvolvimento ficou dividido em dois sub-ciclos ou duas interações. No primeiro ciclo foram implementados os requisitos pelos desenvolvedores e efetuados os testes de aceitação pela equipe de testes, no segundo ciclo foram implementadas as correções pelos desenvolvedores dos erros detectados pela equipe de testes no primeiro ciclo, assim como os re-testes após as correções terem sido efetuadas.

3.2 Atividades do Grupo de XP

As atividades do grupo de XP foram orientadas de acordo com as práticas recomendadas pela metodologia, as quais se dividem nas categorias de processos, programação e equipe.

3.2.1 Processos

Dentre as práticas que envolvem os processos, os desenvolvedores iniciaram o planejamento, fazendo definição incremental dos requisitos do sistema e a criação das *user stories*, pois na metodologia XP não são definidas as especificações formalmente com relatórios e diagramas e muito menos definidos já na sua totalidade.

As demais práticas que foram seguidas pelos desenvolvedores dentro do tópico de processo foram: a presença constante e a interação com o cliente, representado pelo analista de negócios, o qual repassou os requisitos e as necessidades do cliente; liberação de pequenos *releases* para avaliação do mesmo; os processos de testes também foram seguidos, as *user stories* utilizadas como insumos de entrada para criação do plano de teste de aceitação, assim como a criação dos scripts de teste.

Uma *user story* pode ter um ou mais testes de aceitação, conforme definição do cliente e/ou usuários que estarão desenvolvendo estes testes para assegurar que as funcionalidades do sistema estejam de acordo com as especificações. Estes testes garantem também ao cliente e aos desenvolvedores que o sistema em desenvolvimento está progredindo na direção certa. Enquanto todos estes testes criados (cliente/usuário) não forem satisfeitos o produto não pode ser entregue.

3.2.2 Programação

A codificação deste sistema foi feita em dupla, ou seja, dois desenvolvedores na mesma máquina. A padronização do código também foi aplicada, pois no revezamento dos desenvolvedores o código escrito por um, deveria ser entendido pelo outro de

forma clara sem que nenhum dos dois precisasse justificar a parte desenvolvida.

Antes dos desenvolvedores iniciarem a criação do código definitivo, eles trabalharam na construção dos *scripts* de teste ou testes unitários. Estes *scripts* eram alterados e executados cada vez que uma nova implementação era feita no código, para tentar assegurar que não ocorressem impactos negativos. Assim, eram armazenados num repositório junto ao código do sistema de forma que a execução fosse automatizada, garantindo uma proteção essencial contra erros para avaliação entre o tempo de alteração do teste e quantidade de erros encontrados.

Além disso, foi implementada a característica de simplicidade no projeto do sistema, para que os desenvolvedores codificassem apenas as funcionalidades essenciais ao sistema, assim, gastando menos tempo do que desenvolver um código complexo; também sendo essencial esta simplicidade para as alterações que os desenvolvedores faziam no código.

Durante a codificação também se aplicou a prática de reconstrução de alguns códigos, foram eliminadas as funcionalidades desnecessárias para o projeto, fazendo com que a simplicidade aumentasse, evitando a desordem e a complexidade. Em resumo, os desenvolvedores procuraram não adicionar funcionalidades antes do necessário, apenas as funcionalidades específicas de cada *user story*, facilitando a construção e execução dos testes, alteração e reconstrução do código.

3.2.3 Equipe

Das práticas relacionadas à equipe, que foram aplicadas as que se destacaram foram a de código coletivo, pois nenhum dos dois desenvolvedores se considerava pai do código, pois o mesmo tinha sido criado em dupla, possibilitando que qualquer um dos dois executasse a manutenção, e a outra foi a integração contínua que ao final de cada componente desenvolvido no sistema era integrado ao todo e executados todos os *scripts* de testes.

Terminada a etapa de desenvolvimento do software da equipe XP, passou-se para aos testes de aceitação. Estes testes foram realizados pelo analista de testes que teve a responsabilidade de criar e executar os planos de testes.

3.3 Atividades do Grupo de RUP

O grupo de RUP trabalhou paralelamente ao grupo XP no ciclo de vida do software, seguindo as fases de concepção, elaboração, construção e transição, cada uma com várias iterações. Dentro destas fases, os processos também foram seguidos: os *workflows* de requisitos; análise e projeto; implementação; testes.

3.3.1. Levantamento e Detalhamento dos Requisitos

Com base nos requisitos recebidos foi desenvolvido um documento contendo o objetivo geral e o objetivo específico. Nesta fase também foram detalhados os requisitos e especificadas as funções do sistema (funcionalidades do sistema que têm ligação direta com todos os requisitos recebidos), sendo estas detalhadas ao máximo para que na fase de implementação não tivesse nenhuma dúvida.

3.3.2 Desenho e Arquitetura

Nesta fase, definiram-se os atributos do sistema como: (i) tempo de resposta; (ii) tolerância a falhas; (iii) plataforma (s) utilizada (s).

Após o levantamento dos atributos, definiu-se o protótipo de interface do sistema, contendo todos os tipos de objetos utilizados e suas funcionalidades, assim como a imagem de cada uma das telas criadas como protótipo a serem aprovadas pelo cliente.

Foram desenvolvidos os casos de uso do sistema, com a utilização da ferramenta Rational Rose 98, a partir dos requisitos estabelecidos no detalhamento. As tabelas de banco de dados envolvidas já estavam modeladas e disponíveis, bastando ao desenvolvedor utilizá-las para a documentação.

Por último, foi efetuado nesta fase, o detalhamento das especificações técnicas de codificação do sistema.

3.3.3 Implementação

Nesta fase foram implementados todos os componentes e especificações do sistema descritos nas fases anteriores. Portanto, nesta fase o desenvolvedor criava e executava os testes unitários toda vez que um código era escrito e/ou modificados. Os erros eram corrigidos assim que encontrados, fazendo com que o desenvolvedor implementasse o sistema de forma com que a função codificada ficasse disponível no menor tempo possível, e conseqüentemente iria se integrar as outras funções mais rapidamente.

3.3.4 Testes

Durante o desenvolvimento do projeto RUP, foram realizados os seguintes testes: (i) teste unitário; (ii) teste de integração; (iii) teste de sistema; (iv) teste de aceitação.

Durante a execução das atividades do grupo de XP e RUP, foram cronometradas e quantificadas algumas métricas.

4. Resultados Obtidos

Após a conclusão das iterações de testes e acertos do software, os dados coletados foram tabulados e analisados.

Durante as atividades dos desenvolvedores de XP foram regristrados 21 erros e nas atividades de RUP 12, perfazendo um índice de 0,61 e 0,34 erros por hora de desenvolvimento, respectivamente.

Considerando os erros por linha de código, a contagem do número de linhas dos códigos fonte de software desenvolvido foi feita a partir do Software Visual Source Safe 6.0. Foram contabilizadas 1695 linhas para o software do grupo XP e 1433 para o software do grupo RUP. A partir destes números chega-se a uma comparação com a quantidade de erros encontrados na fase de testes. Para cada metodologia, o índice de erros por linha de código é compatível com a quantidade de erros encontrados; sendo 0,1 para XP e 0,003 para RUP. Durante a fase de testes foram registrados 22 erros para o software XP e 5 erros para o software RUP, gerando o índice de 7,14 e 15,53 erros por hora de testes.

Com base no índice de horas, verificou-se que com o tempo de testes de RUP (0,7 horas) foram encontrados 5 erros enquanto que com o dobro deste tempo de testes de XP (1,4167 horas) foram encontrados 22 erros. Portanto, com o esforço de tempo de testes em XP registrou-se um índice de erros por hora superior ao índice de erros por hora de RUP, pois se com 0,7 horas encontrou-se 5 erros, com 1,4167 horas equivalentemente, encontrar-se-iam de 10 a 12 erros, mas foram registrados 22 erros.

Dentro do ciclo de desenvolvimento de software com as duas metodologias propostas, foram registrados diferentes tipos de erros para ambas as equipes. Estes erros foram categorizados em: documentação, sintaxe, pacote, associação, interface, checagem, dados, funções, sistema e ambiente. Ocorreram 12 erros na categoria Sintaxe, 2 erros na categoria Associação e 8 erros na categoria Funções para a equipe XP, não sendo registrados erros nas demais categorias. Ocorreram 2 erros na categoria Sintaxe, 4 erros na categoria Interface e 6 erros na categoria Funções para a equipe RUP, não sendo registrados erros nas demais categorias.

Durante as atividades dos desenvolvedores de XP no segundo ciclo da etapa de desenvolvimento, foram registrados 9 erros e nas atividades de RUP, 4 erros, perfazendo um índice de 1,93 e 1,03 erros por hora de desenvolvimento, respectivamente. As atividades de desenvolvimento do segundo ciclo foram para alterar o software, corrigindo os incidentes encontrados durante a fase de testes do primeiro ciclo.

Nas atividades de testes do segundo ciclo, foram executadas as especificações de testes do plano de teste utilizado anteriormente. Assim, os testes de regressão

foram executados concomitantemente para o grupo XP e para o grupo RUP.

Os incidentes ocorridos nos testes do primeiro ciclo, para o grupo XP e para o grupo RUP, foram determinados como corrigidos, pois não ocorreram nos testes de regressão executados com o plano de testes.

5. Conclusão

Após a análise isolada dos dados obtidos pela coleta, partiu-se para uma avaliação conjunta dos resultados, o que proporcionou as conclusões a seguir.

No tocante ao tempo total necessário para o desenvolvimento, testes e entrega do produto de software, utilizando-se cada uma das metodologias, observou-se que as metodologias RUP e XP apresentaram resultados praticamente equivalentes, sendo 39 e 40 horas respectivamente. A justificativa para estes resultados está no fato que mesmo dedicando um tempo considerável na metodologia RUP para a modelagem e documentação, atividades que não se aplicam no XP, a atividade de criação de *scripts* de testes no XP fizeram com que o resultado final fosse praticamente idêntico.

Em relação á ocorrência de erros, notou-se que nas etapas de projeto e implementação do primeiro ciclo, o grupo de XP apresentou 42% a mais que o grupo de RUP. Mesmo tendo a vantagem que o custo de um erro detectado nestas fases é inferior a um detectado pela equipe de testes ou até mesmo no cliente, o que chama a atenção é a quantidade de erros produzidos pela equipe de XP. Estes dados poderiam ser vistos por uma outra ótica, devido aos *scripts* de testes feitos pelos desenvolvedores XP antes da implementação definitiva foram de quantidade maior que a de RUP, mas esta visão não se confirmou devido a estas proporções de erros terem se mantido na etapa dos testes de aceitação realizados pela equipe de testes.

Comparando-se aplicação das duas metodologias, identificou-se que a XP obteve um custo benefício menor, tendo como base que os esforços de tempo necessário para o desenvolvimento e testes foram praticamente os mesmos e produziram um resultado final de menor qualidade, devido aos erros serem em maior quantidade que no RUP. Uma prova disso é a quantidade de erros por linhas de código. Além da equipe RUP conseguir um código mais enxuto, com menor número de linhas, o índice de erros por linha obtido por eles foi aproximadamente três vezes menor se comparado ao da equipe de XP.

Os dados resultantes da pesquisa levam a concluir que a utilização da metodologia RUP seria mais adequada, levando-se em consideração que o objetivo era saber qual das duas metodologias proporciona o desenvolvimento de um software de melhor qualidade, com o menor

custo possível. Porém, existem algumas considerações que precisam ser feitas a esse respeito. Não é prudente e nem possível efetuar uma conclusão genérica, e sim, afirmar que neste caso, sob estas condições de parâmetros variáveis e escopo apresentado no projeto, obteve-se melhor resultado com a aplicação das técnicas e práticas sugeridas pelo RUP.

Outra observação importante é a de que a pesquisa efetuou apenas um corte, coletando dados estatísticos medindo um momento em que as duas metodologias estavam sendo utilizadas pelas equipes sem uma grande experiência anterior em sua aplicação, pois esse era o objetivo e escopo deste trabalho. Existe uma hipótese, no entanto, que através da prática do uso das metodologias, os processos ficariam mais solidificados entre as equipes, podendo proporcionar os mesmos ou diferentes resultados numa segunda avaliação. Práticas orientadas por uma ou outra metodologia influenciariam a longo prazo, como por exemplo, a disseminação do conhecimento proporcionado pela programação em pares.

Como sugestões para trabalhos futuros, recomenda-se repetir a experiência em projetos de maiores proporções, tanto ao nível de pessoas envolvidas quanto ao nível da quantidade de requisitos do software.

Outra recomendação seria aplicar o mesmo experimento, mas mesclando algumas práticas de RUP e XP, comparando com os resultados da aplicação original.

Bibliografia

BECK, Kent., **Extreme Programming Explained: Embrace Change**, 2000.

BOOCH, Grady e RUMBAUGH, James e JACOBSON, Ivar. **UML Guia do Usuário**. Ed. Campus: Rio de Janeiro, 2000.

FOWLER, Martin. **The New Methodology,** 2000 . Disponível em http://www.martinfowler.com/articles/newmethodology.html. Acesso em 05 março 2002.

HAYES, Steve. **An Introduction to Extreme Programming**, 2001. Disponível em http://www.khatovartech.com. Acesso em 15 abril 2002.

HIGHSMITH, Jim. **History**: **The Agile Manifesto**. 2001, Disponível em http://www.agilealliance.org. Acesso em 24 de fevereiro de 2002.

JEFFRIES, Ron. **What is Extreme Programming.** Disponível em http://www.extremeprogramming.com. Acesso em 20 fev. 2002.

RATIONAL SOFTWARE CORPORATION. **Rational Unified Process – Best Practices for Software Development Teams.** Disponível em http://www.rational.com. Acesso em 08 abril 2002.

SILVA, Alberto M. Rodrigues e VIDEIRA, Carlos A Escaleira. **UML, , Metodologias e Ferramentas CASE.** 1.ed. Portugal: Centro Atlântico, 2001.