UNIVERSIDADE DO SUL DE SANTA CATARINA

HUMBERTO MACHADO FILHO

RASTREABILIDADE DE REQUISITOS COM USO DO *EXTREME PROGRAMMING* (XP)

Palhoça

2010

HUMBERTO MACHADO FILHO

RASTREABILIDADE DE REQUISITOS COM USO DO *EXTREME PROGRAMMING* (XP)

Trabalho de Conclusão de Curso apresentado ao Curso de Especialização em Gerência de Projetos de Tecnologia da Informação da Universidade do Sul de Santa Catarina, como requisito parcial à obtenção do título de Especialista.

Orientadora: Prof. Maria Inés Castiñeira. Dra.

Palhoça

2010

HUMBERTO MACHADO FILHO

RASTREABILIDADE DE REQUISITOS COM USO DO *EXTREME PROGRAMMING* (XP)

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do título de Especialista em Gerência de Projetos de Tecnologia da Informação e aprovado em sua forma final pelo Curso de Especialização em Gerência de Projetos de Tecnologia da Informação da Universidade do Sul de Santa Catarina.

Palhoça, 30 de julho de 2010.

Professora e orientadora Maria Inés Castiñeira, Dra.
Universidade do Sul de Santa Catarina

Prof. Rodrigo Santana, MSc.
Universidade do Sul de Santa Catarina

O mestre na arte da vida faz pouca distinção entre o seu trabalho e o seu lazer, entre a sua mente e o seu corpo, entre a sua educação e a sua recreação, entre o seu amor e a sua religião. Ele simplesmente persegue sua visão de excelência em tudo que faz, deixando para os outros a decisão de saber se está trabalhando ou se divertindo. Ele acha que está fazendo as duas coisas simultaneamente (Texto Zen-budista).

RESUMO

Com o aumento da complexidade dos sistemas na década de 1970, começou haver atrasos na entrega de software, além de problemas de orçamento nos projetos e estes eram normalmente ultrapassados. Em função desses problemas surgiu a Engenharia de Software, com as suas boas práticas fundamentadas em documentação que garanta a rastreabilidade de requisitos, necessária para avaliação do impacto de mudanças, mas que demandam tempo. Atualmente, no entanto, o ambiente competitivo de mercado exige entregas rápidas em função da grande concorrência, privilegiando os métodos ágeis de desenvolvimento, entre os quais o *Extreme Programming* (XP). Em função dessa aparente incompatibilidade, realizou-se uma pesquisa bibliográfica no sentido de apresentar as alternativas do XP para a rastreabilidade de requisitos, através da apresentação das boas práticas da engenharia de software, da pesquisa de métodos ágeis com foco no *Extreme Programming* (XP) e, finalmente, da confrontação daquelas boas práticas com este método ágil. Estudos realizados comprovam que, com uma documentação mínima que promova o gerenciamento das mudanças nos requisitos, é possível garantir a sua rastreabilidade, sem perder a agilidade do método.

Palavras-chave: Rastreabilidade de requisitos. Métodos ágeis. *Extreme Programming* (XP).

SUMÁRIO

1 INTRODUÇÃO: 7
1.1 TEMA 7
1.2 NATUREZA DO PROBLEMA 7
1.3 OBJETIVOS 9
1.3.1 OBJETIVO GERAL 9
1.3.2 OBJETIVOS ESPECÍFICOS 9
1.4 JUSTIFICATIVA 10
1.5 METODOLOGIA 10
1.6 ORGANIZAÇÃO DO TRABALHO 11
2 ENGENHARIA DE REQUISITOS 12
3 O MÉTODO ÁGIL *EXTREME PROGRAMMING* (XP) 14
4 O MÉTODO XP E A RASTREABILIDADE DE REQUISITOS 16
5 CONCLUSÕES E TRABALHOS FUTUROS 22
REFERÊNCIAS 24

1 INTRODUÇÃO:

As boas práticas da engenharia de software sugerem a manutenção de uma matriz de rastreabilidade de requisitos, documento no qual é possível encontrar as ligações entre os mesmos, permitindo a análise do impacto de uma mudança solicitada pelo cliente, por exemplo (SOMMERVILLE, 2007). Os métodos ágeis, entre os quais o *Extreme Programming ou* XP (BECK, 2000) não são voltados à documentação, privilegiando uma interação maior entre as equipes de negócio e desenvolvimento, objetivando geração rápida de código executável.

Este trabalho busca pesquisar as alternativas para a rastreabilidade de requisitos com o uso do método ágil *Extreme Programming* (XP), sem que este perca a sua agilidade.

1.1 TEMA

Rastreabilidade de requisitos com uso de métodos ágeis.

1.2 NATUREZA DO PROBLEMA

Um dos grandes problemas enfrentados no desenvolvimento de sistemas de software grandes e complexos é o da engenharia de requisitos. Esta está relacionada com a definição do que o sistema deva fazer, suas propriedades emergentes desejáveis e essenciais e as restrições quanto à operação do sistema e quanto aos processos de desenvolvimento de software. Podemos, portanto, pensar na engenharia de requisitos como o processo de comunicação entre os clientes e os usuários de software e os desenvolvedores de software (SOMMERVILLE, 2007).

Segundo Pressman (PRESSMAN, 2001), uma completa compreensão dos requisitos é fundamental para um desenvolvimento de software bem sucedido. Não importa

quão bem projetado ou codificado foi o software se o mesmo não atender aos requisitos do usuário.

Independente do tipo de processo utilizado, algumas atividades da engenharia de requisitos são fundamentais (SOMMERVILLE, 2005):

- Elicitação: identificação das necessidades (ou requisitos) dos *stakeholders* (interessados no projeto);
- Análise: classificação dos requisitos obtidos na fase de elicitação;
- Validação: exame da consistência dos requisitos;
- Negociação: obtenção de consenso entre os *stakeholders* ;
- Documentação: produção de documento entendível por todos;
- Gerenciamento: controle das modificações dos requisitos.

Todas as etapas acima citadas são formalmente documentadas e controladas, sendo anteriores a qualquer codificação. Isto, naturalmente, demanda tempo.

Os negócios atualmente operam em um ambiente sujeito a rápidas mudanças. Eles têm de responder a novas oportunidades e mercados, mudanças de condições econômicas e ao surgimento de produtos e serviços concorrentes. Desenvolvimento e entrega rápidas são, muitas vezes, o requisito mais crítico para sistemas de software (SOMMERVILLE, 2007).

Os processos de desenvolvimento de software baseados em especificações completas de requisitos, conforme sugerido pelas boas práticas da engenharia de requisitos, parecem não estar voltados para desenvolvimento rápido de aplicações.

O *Extreme Programming* (XP) é talvez um dos mais conhecidos e amplamente usados dos métodos ágeis. Neste, todos os requisitos são expressos como cenários (chamados histórias do usuário), e são implementados diretamente como uma série de tarefas (SOMMERVILLE, 2007). O foco é nas pessoas e não no processo, havendo sempre um representante do usuário disponível para o projeto. A documentação é mínima, ficando a rastreabilidade de requisitos aparentemente comprometida.

Diante destes dois cenários, quais sejam: garantir a rastreabilidade de requisitos para gerenciar mudanças (o que demanda tempo), ou atender rapidamente às requisições dos clientes utilizando metodologia ágil, formula-se o problema de pesquisa que será explorado nesta monografia, qual seja:

É possível compatibilizar o método ágil *Extreme Programming* (XP) com a rastreabilidade de requisitos, para que se possa avaliar o impacto das mudanças num projeto de software, em tempo e de forma tal que não prejudique a agilidade do método?

1.3 OBJETIVOS

Apresentados a seguir como objetivo geral e específico.

1.3.1 OBJETIVO GERAL

Apresentar as alternativas do método ágil *Extreme Programming* (XP) para a rastreabilidade de requisitos.

1.3.2 OBJETIVOS ESPECÍFICOS

- Apresentar as boas práticas da engenharia de software quanto à rastreabilidade de requisitos;
- Pesquisar os conceitos relacionados com metodologias ágéis;
- Descrever o método ágil *Extreme Programming* (XP);
- Confrontar o modelo pregado pelas melhores práticas da engenharia de requisitos com o método ágil *Extreme Programming* (XP), apresentando a alternativa deste último para a rastreabilidade de requisitos.

1.4 JUSTIFICATIVA

Conforme foi exposto na formulação do problema, por um lado têm-se as boas práticas da engenharia de requisitos, as quais sugerem um trabalho prévio de análise e documentação que garanta a rastreabilidade dos requisitos, anterior a qualquer codificação. Por outro lado existem os métodos ágeis, entre os quais o *Extreme Programming* (XP), o qual prega a agilidade no desenvolvimento, privilegiando as pessoas envolvidas no projeto em detrimento dos processos.

Entende-se que para essa aparente incompatibilidade cabe uma pesquisa, buscando apresentar os conceitos de ambos os pontos de vista, sabendo-se que os dois métodos possuem projetos implementados com sucesso.

1.5 METODOLOGIA

Este trabalho, apesar da abordagem técnica, enquadra-se num modelo teórico dialético (GHEDIN e FRANCO, 2008), no sentido de que procura uma resposta para a aparente contradição entre a metodologia ágil de desenvolvimento *Extreme Programming* (XP) com as boas práticas de documentação que garantam a rastreabilidade de requisitos, fundamental na gerência de mudanças de requisitos num projeto de software.

O trabalho baseia-se na pesquisa bibliográfica com abordagem qualitativa (GROPPO e MARTINS, 2007), realizada através de trabalhos de pesquisa que descrevem as boas práticas da engenharia de requisitos no que se refere à rastreabilidade de requisitos, fundamental para avaliação do impacto das mudanças num projeto de software, mudanças estas que invariavelmente ocorrem.

Os métodos ágeis, entre os quais o *Extreme Programming* (XP), vêm ganhando espaço no sentido de que tem respostas rápidas aos requisitos, privilegiando a codificação rápida num ambiente colaborativo, em detrimento da documentação. Diante disso, é realizada uma pesquisa bibliográfica do método ágil *Extreme Programming* (XP), apresentada a seguir, para averiguar as suas alternativas de documentação que garantam a rastreabilidade de requisitos.

1.6 ORGANIZAÇÃO DO TRABALHO

No capítulo 2 são apresentadas as boas práticas da engenharia de software quanto à rastreabilidade de requisitos.

No capítulo 3 são colocados alguns conceitos relacionados a metodologias ágeis, sendo a seguir descrito o método ágil *Extreme Programming* (XP);

No capítulo 4 são pesquisados os princípios do método ágil *Extreme Programming* (XP) com foco em como este pode resolver a rastreabilidade de requisitos.

No capítulo 5 é apresentada a conclusão do trabalho, assim como é colocada uma proposição para trabalho futuro.

2 ENGENHARIA DE REQUISITOS

Talvez o maior problema enfrentado no desenvolvimento de sistemas de software grandes e complexos seja o da engenharia de requisitos. Esta está relacionada com a definição do que o sistema deve fazer, suas propriedades emergentes desejáveis e essenciais e as restrições quanto à operação do sistema e quanto aos processos de desenvolvimento de software (SOMMERVILLE, 2007).

No início da década de 70, com o rápido crescimento da demanda, a complexidade e a inexistência de técnicas estabelecidas para o desenvolvimento de sistemas, somados à imaturidade da engenharia de software como profissão, culminou com a chamada crise do software, acarretando situações como (ALVES e ALVES, 2009):

- Projetos estourando o orçamento;
- Projetos estourando o prazo;
- Software de baixa qualidade;
- Software não satisfazendo os requisitos;
- Projetos ingerenciáveis e código difícil de manter.

Segundo estudo do Standish Group – Chaos Report de 2000, um dos motivos dos problemas relacionados acima refere-se à falta de controle na mudança de requisitos.

É importante salientar que os sistemas computacionais, inicialmente utilizados apenas como ferramentas de apoio, passaram a ser parte essencial e estratégica do processo, de tal forma que o processo não mais ocorria havendo falha no sistema que o suporta.

Independente do tipo de processo utilizado, algumas atividades da engenharia de requisitos são fundamentais (SOMMERVILLE, 2005):

- Elicitação: nesta fase são identificadas as informações sobre o sistema baseando-se nas necessidades e expectativas dos *stakeholders*;

- Análise: os requisitos iniciais, obtidos através da fase de elicitação, são utilizados como base para esta análise. Eles são distribuídos em categorias, as relações entre eles são exploradas e são classificados conforme sua importância, de acordo com as prioridades dos *stakeholders*;

- Validação: os requisitos são examinados quanto à sua pertinência, consistência e integralidade. Deve ser assegurado que todos os erros tenham sido detectados e corrigidos;

- Negociação: inevitavelmente as opiniões e exigências dos *stakeholders* poderão ser conflitantes. A negociação é realizada para obter consenso no sentido de se obter um conjunto consistente de requisitos;

- Documentação: deve ser produzido um documento de especificação de requisitos entendível pelos *stakeholders* e os desenvolvedores do software;

- Gerenciamento: é responsável por controlar as modificações nos requisitos, as quais inevitavelmente ocorrerão. Esta tarefa é crítica, e pode comprometer o sucesso do projeto se não houver a possibilidade da rastreabilidade dos requisitos, necessária para avaliação do impacto de mudanças nos mesmos.

Os requisitos de um sistema são descrições dos serviços fornecidos pelo sistema e as suas restrições operacionais. Estes requisitos refletem as necessidades dos clientes de um sistema que ajuda a resolver algum problema, por exemplo, controlar um dispositivo, enviar um pedido ou encontrar informações. O processo de descobrir, analisar, documentar e verificar esses serviços e restrições é chamado de engenharia de requisitos. O objetivo do processo de engenharia de requisitos é criar e manter um documento de requisitos do sistema. Este processo inclui quatro subprocessos de alto nível, quais sejam (SOMMERVILLE, 2007):

- Estudo de viabilidade, cujo resultado deve gerar um relatório que recomenda se vale a pena ou não prosseguir com os processos de engenharia de requisitos;

- Elicitação e análise de requisitos, quando fica conhecido o domínio da aplicação e quais serviços o sistema deve fornecer, assim como as suas restrições;

- Validação de requisitos, onde é verificado se realmente os requisitos definem o sistema que o usuário deseja;

- Gerenciamento de requisitos, que se propõe a compreender e controlar as mudanças dos requisitos do sistema.

A mudança ou evolução dos requisitos durante o processo de engenharia de requisitos ou mesmo após a entrada de um sistema em operação é inevitável. À medida que a definição dos requisitos se desenvolve, uma compreensão maior das necessidades dos usuários é obtida, realimentando as informações do usuário que pode, então, propor uma mudança nos requisitos. Além disso o ambiente do sistema e os objetivos da empresa podem mudar, acarretando mudanças nos requisitos (SOMMERVILLE, 2007).

3 O MÉTODO ÁGIL *EXTREME PROGRAMMING* (XP)

Na década de 1980 e início da década de 1990, havia uma visão geral de que a melhor maneira de obter o melhor software era por meio de um cuidadoso planejamento de projeto apoiado pelas melhores práticas da engenharia de software, o que demandava tempo. No entanto, dependendo do porte do projeto, muitas vezes o tempo gasto com o planejamento era maior que o empregado no desenvolvimento, causando insatisfação aos interessados.

Os negócios atualmente operam em um ambiente global sujeito a rápidas mudanças. Eles têm que responder a novas oportunidades e mercados, mudanças de condições econômicas e ao surgimento de produtos e serviços concorrentes (SOMMERVILLE, 2007).

Com o "Manifesto Ágil" em 2001, o termo "método ágil" se tornou conhecido e vem se difundindo entre equipes de desenvolvimento. O foco nas pessoas e interações cria novas condições para se abordar requisitos, e algumas boas práticas são recomendadas para o sucesso desta atividade (ALVES e ALVES, 2009):

- Indivíduos e interações ao invés de processos e ferramentas;
- Software executável ao invés de documentação;
- Colaboração do cliente ao invés de negociação de contratos;
- Respostas rápidas a mudanças ao invés de seguir planos.

O *Extreme Programming* (XP) é talvez o mais conhecido e mais amplamente usado dos métodos ágeis. O nome foi dado em função do envolvimento do cliente com o projeto em níveis "extremos" (SOMMERVILLE, 2007).

Dentre as principais diferenças da XP em relação às outras metodologias de desenvolvimento de sistemas estão (SOARES, 2004):

- *Feedback* constante;
- Abordagem incremental;
- A comunicação entre as pessoas é encorajada.

Comparações entre métodos ágeis e metodologias tradicionais mostram que, para equipes menores onde haja proximidade entre os grupos de desenvolvimento e de negócios, projetos utilizando os métodos ágeis podem obter melhores resultados em termos de cumprimento de prazos, de custos e padrões de qualidade. O mesmo estudo demonstrou ainda que o tamanho dos projetos e das equipes que utilizam as metodologias ágeis tem crescido, provando que a metodologia não se aplica apenas a pequenos projetos e equipes (ALVES e ALVES, 2009).

O método XP é ideal para ser usado em projetos em que os *stakeholders* não sabem exatamente o que desejam e podem mudar muito de opinião durante o desenvolvimento do projeto. Com *feedback* constante, é possível adaptar rapidamente eventuais mudanças nos requisitos (SOARES, 2004).

No *Extreme Programming* todos os requisitos são expressos como cenários (chamados histórias do usuário), que são implementados diretamente como uma série de tarefas. Os programadores trabalham em pares e desenvolvem testes para cada tarefa antes da escrita do código. Em um processo XP, os clientes estão intimamente envolvidos na especificação e priorização dos requisitos de sistema. O cliente é parte da equipe de desenvolvimento e discute cenários com os outros membros da equipe (SOMMERVILLE, 2007).

4 O MÉTODO XP E A RASTREABILIDADE DE REQUISITOS

A insatisfação com as abordagens de desenvolvimento pesadas e orientadas à documentação levou um número de desenvolvedores de software na década de 1990 a propor métodos ágeis de desenvolvimento, entre os quais o *Extreme Programming* (XP). Este permite que a equipe de desenvolvimento se concentre no software somente, em vez de em seu projeto e documentação, apresentando uma abordagem iterativa para especificação, desenvolvimento e entrega de software. Estes métodos ágeis foram criados para apoiar principalmente o desenvolvimento de aplicações de negócios nas quais os requisitos de sistema mudam rapidamente durante o processo de desenvolvimento (SOMMERVILLE, 2007).

As regras, práticas e valores da metodologia XP proporcionam um agradável ambiente de desenvolvimento de software para os seus seguidores, que são conduzidos por quatro valores (SOARES, 2004):

- Comunicação, que visa manter o melhor relacionamento possível entre clientes e desenvolvedores, favorecendo conversas pessoais a outros meios de comunicação;

- Simplicidade, que permite manter códigos simples, sem funções desnecessárias. A aposta da XP é que é melhor fazer algo simples hoje e pagar mais amanhã para fazer modificações necessárias, do que implementar algo complicado hoje e que talvez não seja usado;

- *Feedback*, que permite ao programador ter informações constantes do código e do cliente. A informação do código é dada pelos testes constantes. O retorno do cliente acontece em função de que este terá frequentemente uma parte do software totalmente funcional para avaliar.

- Coragem, para implantar os três valores anteriores.

Além dos quatro valores acima, a metodologia XP contém algumas práticas, sendo as mais importantes as seguintes (BECK, 2000):

- Jogo do Planejamento: Deve ser simples, fácil e rápido o plano do próximo release, definindo o escopo em cima das prioridades, possibilidades de implementação e estimativas técnicas;

- Releases pequenos: Os releases devem ser de pouca duração e sempre produzir software de valor;

- Metáforas: Guiam o desenvolvimento do projeto através e uma estória que explique como o sistema funcione;

- Testes: Testes unitários são feitos pelo programador e testes de aceitação são especificados pelo cliente ditando o que deve funcionar para que o software seja aceito;

- Cliente no local: O cliente é parte integrante da equipe e deverá estar disponível para eventuais dúvidas;

- *Yesterday weather:* Estimar ações futuras baseado em ações iguais ou semelhantes a alguma já realizada no passado.

O amadurecimento metodológico é favorecido pela compilação de experiências aprendidas com o objetivo de oferecer um conjunto de recomendações conhecidas pelo termo "Boas Práticas". Neste sentido, a Agile Modeling (AGILE MODELING, 2010) fornece uma lista de boas práticas para abordagem de requisitos com uso de métodos ágeis , conforme segue:

- Participação ativa dos interessados – stakeholders. O cliente deve participar do projeto, sentido-se parte do time;

- Adotar modelos inclusivos. É importante o uso de modelos e ferramentas de modelagem e documentação de requisitos que reduzam as barreiras de comunicação de forma a manter o envolvimento. O uso de recursos simples com quadro branco, *post-it* e diagramas em alto nível são recomendados;

- Fazer uma primeira abordagem de forma abrangente. Para que se defina o escopo do projeto, é importante delimitar o que será desenvolvido sem detalhar os requisitos;

- Detalhar os requisitos *Just in Time* (JIT). O detalhamento dos requisitos deve ocorrer o mais próximo possível de sua implementação, pois desta forma a memória do detalhe não se perde. Este é o argumento dos agilistas. Cabe aqui, porém, uma questão: O desenvolvimento pode ocorrer próximo ao detalhamento do requisito, mas como fica a manutenção que poderá ocorrer um tempo depois, que não terá à disposição a documentação detalhada e talvez nem a lembrança do cliente ?

- Tratar os requisitos em uma pilha de prioridades. Isto permite definir quais os requisitos devem entrar primeiro em desenvolvimento, viabilizando a prática do detalhamento Just in time tratado no item anterior;

- Visar à implementação do requisito e não à sua documentação. Produzir, manter e rastrear documentação exige um esforço que na visão ágil pode comprometer o tempo de entrega do software ao cliente. Não produzir uma documentação detalhada pode

gerar o benefício em curto prazo, porém pode criar dificuldades de manutenção a longo prazo, em função de que com o tempo pode ocorrer o esquecimento dos detalhes do requisito.

- Reconhecer que existem muitos interessados. É aconselhável que se defina uma pessoa que deverá negociar as várias visões dos requisitos e suas prioridades com todos os interessados no projeto;

- Abordar requisitos independente de plataforma. Abordar requisitos orientados a uma tecnologia pode criar pré-condições para necessidades dos clientes que são independentes de plataforma;

- Decompor os processos em funcionalidades. Menores grupos de requisitos favorecem o entendimento, a estimativa, são mais fáceis de priorizar e consequentemente de gerenciar;

- Tratar a rastreabilidade de requisitos. Rastreabilidade é a capacidade de estabelecer uma relação entre artefatos do projeto. Normalmente é alcançada através de uma matriz de rastreabilidade, a qual pode mapear cada requisito com os seus respectivos diagramas e artefatos do projeto, podendo chegar até a nível de código fonte. Esta matriz é muito importante quando da análise do impacto de determinada mudança em um requisito e pode facilitar também a compreensão de projetos mais complexos envolvendo grande número de requisitos.

A manutenção da rastreabilidade consome tempo e esforço, não se encaixando no paradigma ágil, por se basear fortemente em documentação. Este é um ponto polêmico e difícil de ser tratado. Aconselha-se um debate aberto entre os envolvidos para se avaliar o custo-benefício de se manter uma matriz de rastreabilidade e tomar a decisão em conjunto;

- Explicar as técnicas. Explicar as técnicas usadas pelo pessoal de desenvolvimento faz com que o usuário sinta-se mais envolvido no projeto;

- Utilizar palavras apropriadas ao negócio. É interessante manter um glossário com os termos específicos da área de negócio para facilitar a comunicação com os usuários e criar maior identificação do projeto com o negócio a que se destina;

- Criar um ambiente descontraído. Um ambiente de stress permanente não favorece a produtividade;

- Obter apoio da alta administração. Isto motiva as várias áreas da organização a se envolverem com as atividades do projeto e adotarem as eventuais mudanças culturais que se fizerem necessárias para o sucesso do projeto.

Um preceito fundamental da engenharia de software tradicional é que você deve projetar para a mudança, ou seja, você deve antecipar mudanças futuras para o software e

projetá-lo de tal maneira que essas mudanças possam ser implementadas facilmente. O *Extreme Programming* (XP) descarta o princípio de projetar para a mudança, alegando ser este um esforço inútil, em função de que geralmente as solicitações de mudanças são completamente diferentes (SOMMERVILLE, 2007).

O XP utiliza o conceito de simplicidade em vários aspectos do projeto para assegurar que a equipe se concentre em fazer, primeiro, apenas aquilo que é claramente necessário e evite fazer o que poderia vir a ser necessário, mas ainda não se provou essencial (IMPROVE IT, 2010).

Requisitos são fatores críticos na construção de sistemas, construir bem o produto errado é um dos piores resultados do desenvolvimento de software. Isto pode ocorrer quando existem distorções de entendimento dos requisitos entre usuários e desenvolvedores (ALVES e ALVES, 2009).

Existem vários relacionamentos entre os requisitos e entre os requisitos e o projeto do sistema. Existem também ligações entre os requisitos e os motivos básicos de por que esses requisitos foram propostos. A rastreabilidade de requisitos é a propriedade de uma especificação que reflete a facilidade de encontrar os requisitos relacionados (SOMMERVILLE, 2007).

A rastreabilidade de requisitos é vista pela engenharia de software tradicional como uma técnica fundamental no apoio às diversas atividades do projeto, assegurando que sistemas e software estejam em conformidade às mudanças. Apesar do aumento do custo ou tempo, sem a utilização de mecanismos de rastreabilidade de requisitos podem surgir inconsistências durante o processo de adição, remoção ou modificação de requisitos (CERRI, 2007).

As metodologias ágeis efetivamente não tratam os requisitos com base em documentação. A não documentação do conhecimento na fase de requisitos pode representar uma dificuldade para futura manutenção, através da perda deste conhecimento em médio e longo prazo (ALVES e ALVES, 2009).

Um dos pressupostos universais na engenharia de software é que o custo de modificar um programa aumenta exponencialmente ao longo do tempo. De acordo com o XP, a busca pela simplicidade, testes automatizados constantes e uma atitude de constante refinamento do projeto derrubam este pressuposto, pois tornam o software algo sempre conhecido e mais fácil de ser mantido (BECK, 2000).

A aposta do método ágil XP é de que é melhor fazer algo simples hoje e pagar um pouco mais amanhã para fazer as eventuais modificações necessárias, ao invés de

implementar algo mais complicado, lembrando que o custo da alteração de software diminuiu muito nas últimas décadas em função das várias ferramentas de apoio ao desenvolvimento existentes atualmente (SOARES, 2004).

Projetos adotando práticas do *Extreme Programming* (XP) em equipes avaliadas com o nível 2 de qualidade do SW-CMM (*Capability Maturity Model for Software*), o qual é reconhecido mundialmente, demonstram que uma documentação formal mínima dos requisitos podem ajudar a manter a rastreabilidade de requisitos.

Como exemplo de documentação mínima é citada a criação e manutenção de um Plano de Desenvolvimento do Sistema, documento no qual são registradas as ações de planejamento e acompanhamento do projeto, entre as quais a gerência de mudança, na qual ficam registrados os requisitos de todas as *releases* do sistema, viabilizando alguma rastreabilidade futura (CARDOSO, 2004).

Na mesma linha do artigo de Cardoso (CARDOSO, 2004), modificações feitas na metodologia de desenvolvimento em empresas desenvolvedoras de software que adotam o XP, conseguiram enquadrá-las no nível G do MPS.Br (Modelo de Melhoria de Software Brasileiro), através da criação de uma matriz de rastreabilidade de histórias dos usuários, tornando essas empresas mais competitivas no mercado (em função da avaliação da qualidade de software MPS.Br), sem perder a agilidade do método XP (SANTANA et all, 2006).

Para viabilizar o enquadramento dessas empresas no MPS.Br (Modelo de Melhoria de Software Brasileiro), é sugerida a criação de uma matriz de rastreabilidade de histórias de usuário, na qual cada história é inserida em uma linha e uma coluna da matriz, marcando-se com um x as eventuais dependências entre as histórias (as quais representam os requisitos) na intersecção das linhas/colunas. O quadro 1 a seguir ilustra o que foi colocado acima:

ID de história de usuário	1.1	1.2	1.3	2.1	2.2	2.3	3.1	3.2
1.1		X			X			X
1.2			X		X			
1.3	X							X
2.1			X					X
2.2			X					
2.3	X							
3.1								X
3.2							X	

Quadro 1: Matriz de rastreabilidade de histórias de usuário
Fonte: AUTOR, 2010

No quadro acima, as histórias de usuário são representadas pelas respectivas IDs (identificação), apresentando o exemplo oito histórias ou requisitos, representadas por 1.1 até 3.2. Com o uso dessa matriz é possível visualizar facilmente as dependências entre os requisitos do sistema, facilitando muito o rastreamento do impacto de alguma eventual mudança em qualquer requisito.

Para projetos maiores, nos quais a quantidade de requisitos seja muito grande, faz-se necessário o apoio de ferramentas automatizadas, com o uso de banco de dados, sendo a idéia, no entanto, a mesma que a ilustrada com o quadro 1. Dessa forma, fica viabilizada a rastreabilidade de requisitos com o uso do método ágil *Extreme Programming* (XP).

5 CONCLUSÕES E TRABALHOS FUTUROS

As metodologias tradicionais têm uma forte orientação para que todas as atividades e artefatos sejam formalmente documentados e controlados. Por outro lado, os métodos ágeis, entre os quais o *Extreme Programming* (XP) têm seu foco na iteratividade dos interessados no projeto e menos na documentação, cada qual apresentando suas vantagens e desvantagens.

Os fatores culturais da organização devem ser bem avaliados para que se adote um método ágil de desenvolvimento, pois estes requerem sintonia e proximidade entre as equipes de negócio e desenvolvimento, o que nem sempre acontece.

A natureza do projeto também deve ser avaliada na hora de decidir entre uma metodologia ágil ou tradicional de desenvolvimento. Projetos nos quais os requisitos são bem conhecidos e definidos e tendem a não se alterar muito ao longo do tempo podem adotar uma metodologia tradicional de desenvolvimento, aproveitando as boas práticas da engenharia de requisitos. No entanto, projetos em que os requisitos não estão bem definidos e tem alta probabilidade de serem alterados, têm maior probabilidade de sucesso se adotarem uma metodologia ágil de desenvolvimento.

Tanto os métodos ágeis quanto os métodos orientados à documentação têm cada qual o seu nicho de projetos. Onde um método trabalha bem, o outro tem dificuldades. O ideal é adotar uma abordagem híbrida, tendendo mais para um ou para outro, dependendo da natureza do projeto, a partir de uma análise de riscos (BOEHM, 2002).

Seguindo a mesma linha do artigo de Boehm (BOEHM, 2002), Cardoso (CARDOSO, 2004) coloca que, empresas avaliadas com o nível 2 de qualidade de software do SW-CMM (*Capability Maturity Model for Software*) que resolveram adotar práticas do *Extreme Programming* (XP) com um mínimo de documentação de requisitos para não prejudicar a agilidade do método ágil, viabilizaram a rastreabilidade de requisitos.

Da mesma forma, conforme Santana (SANTANA et all, 2006), através da criação de uma matriz de rastreabilidade de histórias de usuários, empresas desenvolvedoras de software que adotam o XP, conseguiram enquadrar-se no nível G do MPS.Br (Modelo de Melhoria de Software Brasileiro), tornando essas empresas mais competitivas no mercado, sem perder a agilidade do método XP .

Este trabalho, no capítulo 2, apresentou as boas práticas da engenharia de software quanto à rastreabilidade de requisitos. O capítulo 3 pesquisou conceitos relacionados com

metodologias ágeis de desenvolvimento de software e descreveu em seguida o método ágil *Extreme Programming* (XP). O capítulo 4 apresentou os princípios do XP, confrontando-os com as melhores práticas da engenharia de requisitos, no sentido de apresentar alternativas deste método ágil para a rastreabilidade de requisitos. Dessa forma, pode-se afirmar que o objetivo desta proposta, qual seja, mostrar as alternativas do método ágil *Extreme Programming* (XP) para a rastreabilidade de requisitos, foi atingido satisfatoriamente.

Para que se favoreça a conciliação entre os benefícios dos métodos ágeis com a documentação sugerida pelas boas práticas da Engenharia de Software, sugere-se, como trabalho futuro, a pesquisa de ferramentas automatizadas para o gerenciamento de requisitos com uso de métodos ágeis, de forma que se possam implementar alterações nos sistemas com a garantia de qualidade pregada pela Engenharia de Software, tornando mais ágil a documentação.

REFERÊNCIAS

Agile Modeling (AM). Práticas eficazes de Modelagem e Documentação. Disponível em http://www.agilemodeling.com. Acesso em 07 jun. 2010.

ALVES, Sérgio de Rezende; André Luiz ALVES. "Engenharia de requisitos em métodos ágeis". In PUBLICAÇÕES DA IV MOSTRA DE PRODUÇÃO CIENTÍFICA DA PÓS-GRADUAÇÃO LATO SENSU DA PUC Goiás, 2009. Disponível em <http://www.cpgls.ucg.br/ArquivosUpload/1/File/CPGLS/IV MOSTRA/TECNOLOGIAS/Engenharia De Requisitos Em Mtodologias geis.pdf>. Acesso em 05 abr. 2010.

BECK, Kent. "Programação extrema explicada". 1a edição. Bookman Companhia Editora, 2000.

BOEHM, B. "Get Ready for Agile Methods, with Care". IEEE Computer, Janeiro de 2002. Disponível em <http://www.cin.ufpe.br/~processos/referencias/Boehm-GetReadyforAgileMethods_withCare.pdf>. Acesso em 15 abr. 2010.

CARDOSO, Carlos Henrique Rodrigues. "Aplicando práticas de extreme programming (XP) em equipes SW-CMM nível 2". VI Simpósio Internacional de Melhoria de Processos de Software, 2004 . Disponível em <http://www.simpros.com.br/Apresentacoes_PDF/Artigos/Art_05_Simpros2004.pdf>. Acesso em 31 mar. 2010.

CERRI, Elisa Cerri e. "Um modelo de rastreabilidade entre o documento de especificação de requisitos e o modelo de casos de uso do sistema". 2007. 190 f. Dissertação (Mestrado em Engenharia de Requisitos) - Pontifícia Universidade Católica do Rio Grande do Sul, Porto Alegre, 2007.

GHEDIN, Evandro; FRANCO, Maria Amélia Santoro. "Questões de método na construção da pesquisa em educação". São Paulo: Cortez, 2008.

GROPPO, Luís Antônio; MARTINS, Marcos Francisco. "Introdução à pesquisa em educação". 2. Ed. Piracicaba: Biscalchin Editor, 2007.

IMPROVE IT. "Extreme Programming". Disponível em: <http://improveit.com.br/xp>. Acesso em 10 abr. 2010.

MANIFESTO ÁGIL. Disponível em: < http://agilemanifesto.org> Acesso em 07 jun. 2010.

PRESSMAN, Roger S. "Engenharia de Software". McGrawHill, 2001.

SANTANA, Célio A.;TIMÓTEO, Aline L.;VASCONCELOS, Alexandre M.L. "Mapeamento do modelo de Melhoria do Processo de Software Brasileiro (MPS.Br) para empresas que utilizam Extreme Programming (XP) como metodologia de desenvolvimento". V Simpósio Brasileiro de Qualidade de Software, 2006 . Disponível em <www.sbc.org.br/bibliotecadigital/download.php?paper=973>. Acesso em 10 jun. 2010.

SOARES, Michel dos Santos. "Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software". Journal of Computer Science, 2004 . Disponível em <http://wiki.dcc.ufba.br/pub/Aside/ProjetoBitecIsaac/Met._Ageis.pdf>. Acesso em 15 abr. 2010.

SOMMERVILLE, Ian. "Integrated Requirements Engineering: A Tutorial". IEEE Software, 22(1), 16-23, Janeiro/Fevereiro 2005.

SOMMERVILLE, Ian. "Engenharia de Software". 8a edição. Pearson Addison-Wesley, 2007.