Uma abordagem para gerência de requisitos integrada com práticas ágeis de gerência de projetos

Rudiney Altair Franceschi[1], Ana Marcia Debiasi Duarte[2]

[1] SGI Sistemas Ltda

[2] Universidade do Oeste de Santa Catarina - Unoesc
Chapecó – SC – Brasil
rudiney@sgisistemas.com.br, ana.duarte@unoesc.edu.br

Abstract. *The Agile project management has become increasingly popular, offering alternatives to traditional processes. The implementation of the Scrum framework for managing software projects has also been widespread as an alternative since it is a less bureaucratic approach that facilitates the planning and project control. The software requirements management can be compromised in the medium and long-term since Scrum is based on histories as requirements and histories maintenance are not traceable during software product life cycle. This paper presents the results of applying the techniques of requirements management and Scrum on a software project management keeping the agility of Scrum and histories maintenance with requirements management.*

Resumo. *Os métodos ágeis de gerenciamento de projetos têm se tornado cada vez mais populares, propondo alternativas aos processos tradicionais. A aplicação do framework Scrum no gerenciamento de projetos de software também tem sido difundida como alternativa por se tratar de uma abordagem menos burocrática que facilita o planejamento e controle do projeto. Por não tratar especificamente da gestão de requisitos de software, a médio e longo prazo, a perenidade dos produtos de software construídos usando Scrum pode ficar comprometida. Este trabalho apresenta os resultados da aplicação de técnicas de gerência de requisitos integrada com práticas ágeis de gerência de projetos propostas pelo framework Scrum.*

1. Introdução

Existem muitos argumentos defendendo a gestão de requisitos no desenvolvimento de um software. Um dos mais simples é que não há como ser preciso sobre aquilo que não conhecemos. Outro argumento é relacionado à correção de defeitos: é sabido que corrigir um defeito enquanto o software é desenvolvido custa menos que corrigi-lo depois do software pronto [Eberlein, 2002].

Para Pfleeger (2004), requisitos incompletos são a causa principal de falhas em projetos. A engenharia de requisitos é um subprocesso da engenharia de software e tem o objetivo principal de identificar, analisar, documentar e validar os requisitos do sistema a ser desenvolvido. Portanto a engenharia de requisitos é fundamental para o sucesso de projetos de software.

Muitas metodologias têm sido propostas com o objetivo de entregar o software ao cliente de forma mais rápida e mais coerente atendendo todos os requisitos propostos

[Frauke, 2003]. Segundo [Koscianski, 2007], as metodologias ágeis prometem melhorar a qualidade e a produção do software. Estas abordagens ágeis compartilham os mesmos princípios: aumentar a satisfação do cliente, flexibilizar as alterações nos requisitos, entregar versões frequentemente e aproximar desenvolvedores e das pessoas que definem o negócio do software.

O objetivo deste trabalho é apresentar uma proposta de gerência de requisitos aplicada à metodologia Scrum de gerência de projetos. Esta proposta foi aplicada a uma empresa que necessitava gerenciar os requisitos de maneira mais formal em um ambiente ágil de gerenciamento de projetos. A gerência dos requisitos deveria incorporar a definição e documentação para a construção, além de garantir as informações necessárias após a construção para uma manutenção de qualidade.

2. Estrutura adotada para a solução

Em empresas que gerenciam seus projetos de forma ágil (e.g., Scrum), a gestão de requisitos não é, inicialmente, tratada da forma tradicional, pois possuem uma abordagem própria, contemplando histórias que são bases para o desenvolvimento do produto. Quando estes modelos ágeis são aplicados em empresas que trabalham na construção de produtos que são comercializados e que sofrem uma grande demanda de adaptações e evoluções constantes a qualidade do trabalho de evolução pode ficar comprometido [Thomas, 2006]. Isso acontece, pois os mecanismos de rastreabilidade que permitem identificar os impactos oriundos da construção do produto são quase inexistentes. Segundo Espinoza [2009], esta situação pode causar vários problemas nos processos de gestão de mudança, análise de impacto, e estimativa. Outro fator a se considerar, é que os usuários não participam diretamente da construção das histórias conforme é preconizado no Scrum. Geralmente, as empresas possuem internamente, profissionais que são responsáveis por identificar as necessidades, avaliar os conflitos entre as solicitações e definir quais serão selecionadas para o desenvolvimento. Assim, em muitas empresas existe a necessidade de manter os históricos da construção dos produtos para beneficiar-se da rastreabilidade entre os requisitos e os arquivos fontes, sem perder a agilidade obtida com o gerenciamento Scrum. A forma encontrada para controlar a evolução de um produto e manter as características ágeis do Scrum foi desenvolver uma ferramenta para automatizar o processo, uma vez que seriam incorporadas mais informações e controles ao processo ágil de produção.

A ferramenta foi desenvolvida a partir da necessidade de uma empresa que utiliza Scrum para gerenciar seus projetos de software, mas que mantém várias versões de seu produto que precisam ter suas evoluções rastreadas.

Para o desenvolvimento da ferramenta foi necessário identificar todos os artefatos produzidos durante os ciclos de desenvolvimento e quais destes artefatos deveriam ser controlados e gerenciados. Esta análise foi realizada levando em consideração o custo e os benefícios da utilização de uma ferramenta para a gestão de requisitos sem burocratizar o processo ágil de desenvolvimento adotado na empresa, para não comprometer a produtividade.

O principal objetivo da ferramenta desenvolvida é relacionar cada um dos artefatos e organizar os seus relacionamentos de uma maneira que seja possível analisar o impacto de um requisito em outros requisitos do produto desenvolvido. Como pode ser observado na Figura 1, o repositório de requisitos do produto é um dos artefatos

gerenciados onde um requisito pode ter impacto em histórias e em outros requisitos. Além do impacto identificado, um requisito pode ter um ou mais arquivos fonte que implementam tal requisito. A associação dos requisitos com os arquivos fonte que os implementam é fundamental para garantir a rastreabilidade. Para chegar a esta estrutura, houve um estudo da relação entre *(i)* história – que define o que deve ser feito do ponto de vista do projeto, *(ii)* tarefa – que define o conjunto de tarefas necessárias para construir uma história, *(iii)* artefatos fonte – que armazena todos os artefatos criados e/ou mantidos durante uma tarefa, e *(iv)* requisitos – que definem todos os requisitos funcionais que fazem parte do produto de software construído. O embasamento desta estrutura e a justificativa para o seu uso é apresentado em Franceschi (2010).

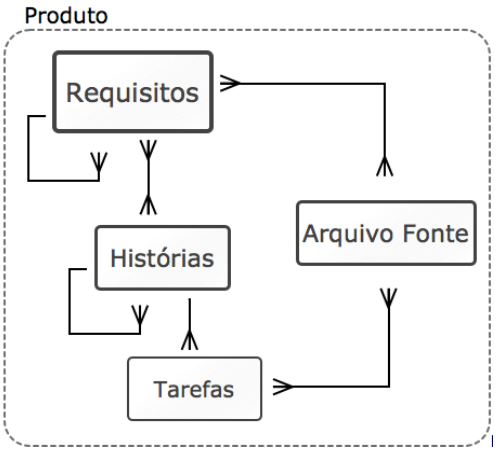

Figura 1. Estrutura das informações registradas pela ferramenta

O software foi desenvolvido para um ambiente *web*, escrito na linguagem Ruby[1], utilizando o *framework Rails*[1]. Utiliza como banco de dados o PostgreSQL[2] e tem como servidor web o Apache[3] com o *plugin* Phusion Passenger[4] para servir aplicações escritas em Ruby on Rails através do Apache. Os detalhes do funcionamento do software são apresentados a seguir. Para possibilitar o controle de versionamento dos artefatos, houve uma integração com o SVN[5], que é apresentada na Seção 3.2 deste trabalho.

3. Descrição da Ferramenta

Ao iniciar o sistema, o usuário da ferramenta escolhe o produto que deseja trabalhar e assim tem acesso ao seu repositório, com todos os requisitos e o seu histórico de desenvolvimento. Seguindo o conceito do Scrum, cada produto cadastrado na ferramenta, possui um *Product Backlog*, que contém todas as histórias que o compõem. A Figura 2 apresenta como é tratado o *Product Backlog* na ferramenta e os atributos que fazem parte do seu registro.

---

1 "Rails", "Ruby on Rails", “Rails” são marcas registradas de David Heinemeier Hansson.

2 Distribuído através da licença PostgresSQL. Ferramenta de código aberto - www.postgresql.org

3 Distribuído através da licença Apache versão 2.0. Ferramenta de código aberto - httpd.apache.org

4 "Phusion Passenger" é marca registrada de Hongli Lai &Ninh Bui - www.modrails.com

5 Desenvolvido por CollabNet Inc. Ferramenta de código aberto – www.collab.net

Na medida em que as histórias são construídas o produto vai sendo desenvolvido e as funcionalidades são disponibilizadas no produto. A solução proposta neste trabalho considera a rastreabilidade entre as histórias e tarefas usadas para construir as funcionalidades (requisitos do produto). Desta forma é possível identificar quais histórias foram utilizadas para construir e manter as funcionalidades disponibilizadas no produto. A Figura 3 apresenta as informações que fazem parte de um requisito do produto depois de sua construção, com a sua descrição e seus relacionamentos. No entanto, para completar a rastreabilidade até no nível dos arquivos fonte do produto para viabilizar a análise de impacto no processo de manutenção (seja evolutiva ou corretiva), a relação mais importante fica a cargo da associação das funcionalidades e dos arquivos fonte manipulados pelos desenvolvedores. Na Figura 3 ainda é possível identificar as relações entre os requisitos como fortes e fracas. As fortes estão representadas na cor vermelha e as fracas na cor azul. As relações fortes representam uma dependência que tem impacto, independente do tipo da manutenção. As associações fracas podem representar um impacto, depende da natureza da manutenção.

Product Backlog

Nova História | Mapa de Impacto | Pontos: 51/2 | Em Desenv.: 5

#	?	Módulo	História	Sprint	Situação	P	T
H18	Não		Melhorias no Desktop	1	Pronta		P
H19	Não	Faturamento	Definição de preços de venda	1	Pronta		P
H21	Não		Consulta Pedidos vendidos por cliente	0	Iniciada	2	P
H27	Sim	Faturamento	Operações de saída	0	Iniciada	3	M
H20	Sim	Faturamento	Condição de pagamento	1	Iniciada		P
H33	Sim		Código de digitação nota fiscal	0	Aguardando	0	M
H34	Não		Consulta Pedido/NF	0	Aguardando	1	P
H22	Sim		Correções no Sistema	1	Aguardando		P
H23	Não		Cadastro sobre produtos	1	Aguardando		M
H24	Não		Cadastro de clientes	1	Aguardando		G

Figura 2. Product Backlog

A solução construída e apresentada neste trabalho prevê que o cliente ou usuário do produto de software terá como referência sempre o produto e as funcionalidades com as quais ele possui contato. Quando o cliente está utilizando o produto, esta é a referência usada e não necessariamente as histórias que foram especificadas para construir o requisito. Desta forma, a ligação entre funcionalidades e arquivos fontes permite identificar a rastreabilidade independente da(s) história(s) usada(s) para construí-la. Este fator foi fundamental para a seleção dos repositórios que a ferramenta iria contemplar e para determinar a relação entre cada componente controlado na ferramenta.

Quando há a necessidade de alguma análise de impacto para a manutenção (corretiva ou evolutiva) envolvendo um requisito, a tela de visualização do requisito permite identificar os possíveis impactos (Figura 3).

Figura 3. Visualização do requisito e seus relacionamentos

A manutenção do repositório da ferramenta acontece durante o processo de desenvolvimento. No momento de trabalhar com as histórias, através das tarefas que são executadas, os desenvolvedores criam ou alteram artefatos de construção que são identificados. A Figura 4 apresenta a solução construída para implementar a rastreabilidade. Na primeira coluna existe o identificador das tarefas. Através deste identificador, após o *commit* executado pelo desenvolvedor, o sistema importará os arquivos fonte do repositório e relacionar-los-á com as histórias cadastradas na ferramenta. Os arquivos fonte são controlados pela ferramenta TortoiseSVN.

Figura 4. Tarefas e arquivo(s) fonte de uma história

A ferramenta ainda apresenta uma tabela para a análise da relação entre as histórias, pois enquanto uma *sprint* é executada, as histórias são as referências usadas como principal forma de controle dos impactos, uma vez que os requisitos ainda estão em construção. A Figura 5 mostra como são apresentadas as dependências entre as histórias. Seguem o mesmo padrão para dependências fortes e fracas, nas cores vermelha e azul, respectivamente.

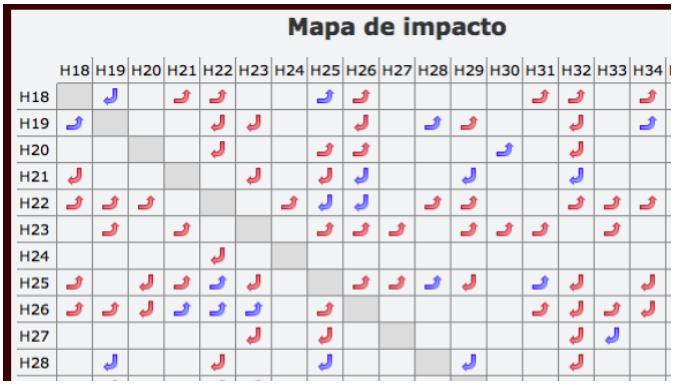

Figura 5. Mapa de impacto entre as histórias de uma sprint (parcial)

3.1. Integração com o repositório SVN

A importação das informações do repositório SVN é a funcionalidade da ferramenta que relaciona os arquivos fontes, codificados pelos desenvolvedores, e devidamente relacionadas com as tarefas, com as histórias e requisitos do produto gerenciados pela ferramenta. O desenvolvedor, para codificar uma tarefa, realiza manutenções em um ou mais arquivos fonte. Esta funcionalidade busca, no repositório central, quais arquivos fonte foram mantidos para codificar uma tarefa e assim relacioná-los com as histórias e com os requisitos do produto, automaticamente. A Figura 6 exemplifica a ação de registrar uma tarefa no repositório (através do comando commit) utilizando no comentário o identificador da tarefa dois da história vinte. Deste modo é registrado no repositório o arquivo fonte criado ou modificado para realizar determinada tarefa.

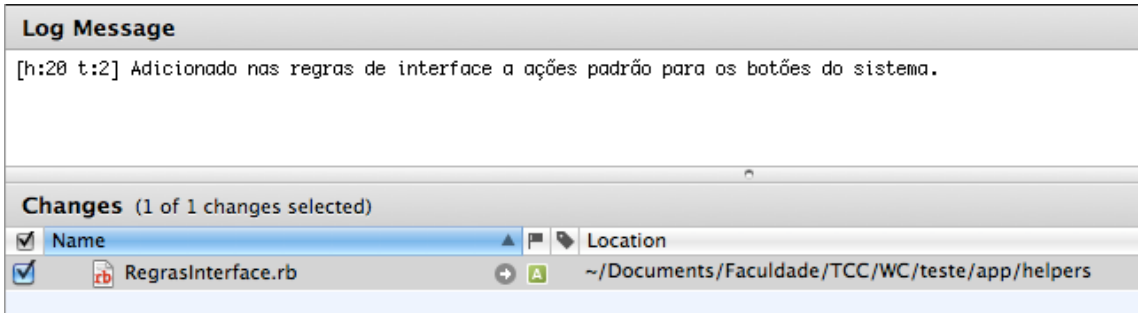

Figura 6. *Commit* utilizando o identificador de uma tarefa

A ferramenta apresenta ainda a possibilidade de alterar informações da importação como o endereço físico, usuário e senha de conexão com o repositório. O resultado de uma importação pode ser de sucesso ou não. Quando o processo de importação encontra problemas para relacionar os arquivos fonte, um retorno com os possíveis erros é exibido ao usuário. Desta forma, um controle sobre as inconsistências é possível de ser controlado. Como o processo de associação dos arquivos fonte com as tarefas é manual, foi necessário desenvolver uma forma de verificação para resolver possíveis conflitos encontrados.

3.2. Resultados da utilização da ferramenta

A ferramenta descrita neste trabalho foi construída a partir de um problema observado na empresa SGI Sistemas, com sede na cidade de Chapecó no estado de Santa Catarina. O problema foi tema de uma monografia de conclusão de curso (Franceschi, 2010), está em funcionamento desde outubro do ano de 2010 e serve para a documentação de dois produtos desenvolvidos atualmente. Depois da implementação da ferramenta foi possível observar o impacto positivo na gestão do produto, através do controle sobre as informações dos requisitos e dos artefatos ou arquivos fonte associados. Foi possível também identificar a evolução do produto através do acompanhamento destas informações.

Para a apresentação deste trabalho, são apresentados os dados registrados no banco de dados da ferramenta, relacionado a um módulo do produto da empresa. O módulo se refere a remuneração de gerentes de redes de loja. Formado por um conjunto relativamente pequeno de histórias, se comparado ao produto comercializado, este módulo permite observar a complexidade do relacionamento entre histórias e requisitos vivenciados ao longo do seu desenvolvimento. A Figura 7 apresenta as cinco histórias

identificadas no módulo avaliado no uso da ferramenta proposta. É possível perceber a relação complexa entre histórias e requisitos do produto, como a história 19, por exemplo.

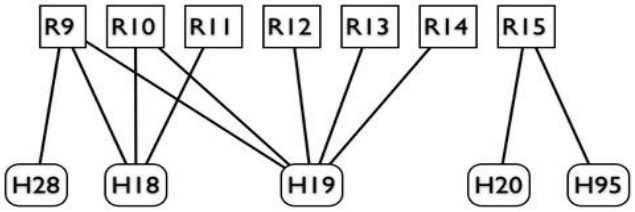

Figura 7. Relação entre requisitos e histórias

Outra informação oferecida pela utilização da ferramenta é a complexidade da rastreabilidade entre os requisitos e os artefatos criados para atendê-los. Sem a identificação da relação entre requisitos e os arquivos fonte, ficaria muito difícil de "lembrar", ou encontrar todos os possíveis impactos em uma análise para realização de uma evolução ou correção no produto. A Tabela 1 apresenta uma média de 56,14 arquivos para cada requisito identificado.

Tabela 1. Relação entre requisitos e quantidade de arquivos fonte gerados

Requisito		Arquivos fonte
R9	Cadastro de indicadores	75
R10	Cadastro de valor do peso	52
R11	Definição de metas	74
R12	Remuneração fixa de gerentes e supervisores	26
R13	Faixas de atingimento de metas	26
R14	Bônus de remuneração	26
R15	Análise de metas de gerentes e supervisores	114

Este é um número elevado e que fica evidente através do uso da ferramenta.

4. Conclusão

A construção e uso da ferramenta proposta neste trabalho, mostrou que é possível integrar as práticas da engenharia de requisitos tradicional em um ambiente de desenvolvimento que adota práticas ágeis de gestão de projetos. No entanto, foi necessário um estudo aprofundado na área de gerência de requisitos para identificar como as estratégias existentes poderiam ser aplicadas em um ambiente ágil de gerenciamento de projetos. Com relação às atividades propostas pela engenharia de requisitos, foi abdicado do detalhamento mais específico dos requisitos. Este fator não causou impacto negativo na construção do produto, uma vez que os requisitos já eram utilizados em um formato reduzido de documentação em função das práticas ágeis. Nas práticas ágeis, foi necessário adicionar tarefas de prescrição. Houve uma percepção pelos desenvolvedores de diminuição da agilidade. No entanto não foi possível perceber a redução na produtividade associada à velocidade do time no controle realizado pelo Scrum. Este ponto não foi aprofundado, mas é sugerida uma investigação detalhada para medir o impacto das mudanças no processo em termos de produtividade.

Como partes dos produtos desenvolvidos pela empresa e que estão sendo controlados pela ferramenta já foram implantados e estão em uso nos clientes, foi possível observar o seu comportamento durante a análise de impacto que deve ser realizada antes do desenvolvimento. A ferramenta é consultada em três momentos: *(i)* durante a fase de estimativa, pois o impacto de histórias no produto já construído é levando em consideração para estimar o tamanho ou o esforço para o desenvolvimento, *(ii)* durante o desenvolvimento de histórias, quando os desenvolvedores fazem as consultas para identificar os pontos que devem ser incluídos no desenvolvimento, e *(iii)* durante os testes, quando os impactos devem ser considerados para reduzir os defeitos causados por impactos não tratados no produto.

Como principal resultado, a experiência relatada neste trabalho demonstra que é possível, do ponto de vista da gerência ágil de projetos, conviver com técnicas da engenharia de requisitos tradicionais e tirar proveito desta configuração durante o processo de desenvolvimento, mais especificamente durante o processo de manutenção, seja corretiva ou evolutiva.

De forma geral, o benefício face ao custo de manter as informações na ferramenta, foi maior, pois a manutenção evolutiva e corretiva é feita de forma mais segura, aumentando a qualidade. Os impactos são analisados, tanto para a realização das manutenções como para os testes. Isto reduz o retrabalho, e desta forma, reduz os custos associados ao produto.

Referências

Eberlein, A. Agile Requirements Definition: A View from Requirements Engineering. 2002.

Espinoza, A., Garbajosa, J. A study to support agile methods more effectively through traceability. Volume 7, Number 1. Innovations Syst Softw Eng. 2009. DOI 10.1007/s11334-011-0144-5. http://www.springerlink.com/content/gh3j34krh171vp04/fulltext.pdf

Franceschi, R. A. (2010) “Solução para gestão de requisitos integrado com metodologia ágil de desenvolvimento”. Monografia de Conclusão de Curso de Sistemas de Informação. Unoesc – Chapecó SC.

Frauke Paetsch, Armin Eberlein, and Frank Maurer. 2003. Requirements Engineering and Agile Software Development. In *Proceedings of the Twelfth International Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises* (WETICE '03). IEEE Computer Society, Washington, DC, USA, 308-.

Koscianski, A. Soares, M. S. Qualidade de Software. Segunda ed. Novatec. 2007

Pfleeger, S. Engenharia de software: teoria e prática, segunda ed. Prentice Hall, 2004.

Thomas, D. Agile Artifacts - Documenting, Tracking and Reporting, Trust The Source Luke!, in *Journal of Object Technology*, vol. 6 no. 3 March - April 2006, pp. 25-31 http://www.jot.fm/issues/issue_2007_03/column4.