Rastreabilidade de Requisitos em um Cenário de Desenvolvimento Ágil: Um Estudo de Caso

Diego Lucas Coelho[1], Maria Inés Castiñeira[2], Aran T.B. Morales[3]

[1,2]Análise e Desenvolvimento de Sistemas - Universidade do Sul de Santa Catarina (UNISUL) Florianópolis - SC - Brazil

[2,3] Sistemas de Informação– Universidade do Sul de Santa Catarina (UNISUL) Florianópolis – SC – Brazil

diegoclho@gmail.com, maria.castineira@unisul.br, aran.morales@unisul.br

Abstract: *Study on the management of software requirements in an agile development scenario. The paper analyzes the process and practices of software, with emphasis on the management of requirements, in a technology company, located in Greater Florianópolis. Initially the process of software development (As Is) is described, as well as the motivators and actions taken to apply the requisite management techniques and practices. After that, the process with the application of requirements traceability techniques, using the Jira Atlassian tool, to support the development cycle of the company is described. The report includes the prospect of suggestions and improvements.*

Keywords: *Requirements Management. Requirements Traceability.Software Engineering. Agile Development*

Resumo: *Estudo sobre o gerenciamento de requisitos de software em um cenário de desenvolvimento ágil. O trabalho analisa o processo e as práticas de software, com ênfase para o gerenciamento de requisitos, em uma empresa de tecnologia, situada na Grande Florianópolis. Inicialmente é descrito o processo de desenvolvimento de software (As Is), assim como os motivadores e as ações tomadas para aplicar as técnicas e as práticas de gerenciamento de requisitos. A seguir é apresentado o processo após a aplicação das técnicas de rastreabilidade de requisitos, utilizando a ferramenta JiraAtlassian, para apoiar o ciclo de desenvolvimento da empresa. O relato inclui a prospecção de sugestões e melhorias.*

Palavras-chave: *Gerenciamento de Requisitos. Rastreabilidade de Requisitos. Engenharia de Software. Desenvolvimento Ágil*

1. Introdução

A engenharia de requisitos auxilia a definir o real propósito do software e como o mesmo será atingido, influenciando diretamente no sucesso ou o fracasso do projeto de software. (HANS VAN VLIET, 2007, NUSEIBEH, EASTERBROOK, 2000).

Este trabalho descreve o estudo de caso na gestão de requisitos em uma empresa desenvolvedora de software. Um dos desafios no processo de desenvolvimento de software é a rastreabilidade de seus requisitos. Essa qualidade é bastante destacada no caso de um único produto de software que é utilizado por vários clientes, como no presente estudo de caso.

A empresa analisada possui seu produto próprio, que sofre constantes alterações de acordo com a necessidade de cada cliente. Assim, é de suma importância garantir que um novo requisito solicitado por um cliente não divirja com os requisitos dos demais. A rastreabilidade de requisitos é muito relevante, no que diz respeito à vinculação com cada funcionalidade impactada, sua documentação, assim como o histórico de alterações já realizadas e quais os principais envolvidos.

O objetivo deste trabalho é realizar a descrição de um estudo de caso sobre o processo de gerenciamento de requisitos de uma empresa, focando na rastreabilidade dos mesmos. O trabalho apresenta o processo inicial e como esse processo foi modificado para atingir uma rastreabilidade dos requisitos em prol da qualidade de software. Como objetivos específicos pode-se citar: a) analisar o processo de Gerenciamento de Requisitos; b) elencar as práticas de Rastreabilidade de Requisitos adotadas; c) validar a aplicação das práticas de rastreabilidades detalhadas; e finalmente, d) prospectar evoluções do processo.

No trabalho são analisadas soluções para rastreabilidade de requisitos, que contribuem na resolução de alguns problemas enfrentados no dia a dia, durante o gerenciamento do produto de software. Inicialmente é apresentada a revisão da literatura sobre engenharia de requisitos e métodos ágeis. A seguir é apresentado o estudo de caso e finalmente as considerações finais deste trabalho.

2. Revisão da Literatura

Pressman (2011) afirma que junto ao crescimento da importância do software, veio a necessidade de novas tecnologias, métodos e ferramentas para auxiliar no desenvolvimento e manutenção, focando na qualidade. Assim, surgiram uma série de práticas e ferramentas para auxiliar no correto desenvolvimento de software, que deixou de ser exclusivamente sinônimo de programas computacionais, mas sim, do conjunto de toda a configuração e documentação que descreve a estrutura, regras e premissas para que o software atinja o comportamento desejado. Essa foi a origem da Engenharia de Software (SOMMERVILLE, 2010). Um dos assuntos de atenção e estudo da engenharia de software é a prática e tratamento dos requisitos de software.

2.1 Engenharia de Requisitos

Com a premissa de que "Requisito" pode ser definido como "Uma condição ou capacidade necessária por um usuário para resolver um problema ou atingir um

objetivo" (IEE610, 1990), pode-se assumir que o processo de Engenharia de Requisitos foca na descoberta e análise dos reais problemas dos interessados, elicitação e documentação dos requisitos, validação e consequentemente o acompanhamento dos mesmos, durante todo o ciclo de vida de um software. (IEEE SWEBOK GUIDE, 2004).

A Engenharia de Requisitos auxilia no processo de software para que a necessidade do cliente seja entendida e suprida, através do levantamento e especificação de requisitos, até a fase de validação e gestão dos mesmos. Apesar da Engenharia de Requisitos ser um processo dentre vários que compõem a Engenharia de Software, a mesma também possuí seus próprios métodos e processos, que por sua vez podem ser claramente evidenciados em sete etapas (WIEGERS, 2003). São elas:

- Concepção: Em sua maioria, os Projetos de Softwares nascem de um problema ou uma necessidade real, desta forma, a etapa de Concepção da Engenharia de Requisitos já entra nesta fase inicial do projeto, auxiliando na identificação dos interessados, e também de um primeiro levantamento dos requisitos, assim como uma certa análise de viabilidade.
- Levantamento: Apesar de parecer simples, a fase de levantamento dos requisitos é coberta de dificuldades e imprevistos. Na maioria das vezes a real necessidade do usuário não é realmente aquela que o mesmo menciona. Em sua maior parte, isto ocorre pelo fato do usuário ter um errôneo entendimento do seu problema ou limitações tecnológicas, acarretando repetidamente na omissão de informações que adotam como "óbvias".
- Elaboração: A partir das informações coletadas nas fases de Concepção e Levantamento, se faz necessária uma análise detalhada das mesmas. Na fase de elaboração, todas as necessidades são refinadas, e o modelo de requisitos é elaborado, contendo os possíveis cenários, documentações e protótipos necessários para um correto entendimento de cada comportamento esperado pelo usuário.
- Negociação: Durante a fase de Engenharia de Requisitos, um número considerável de novos requisitos ou alterações de requisitos já validados irão surgir, e muitos deles podem extrapolar ou interferir diretamente no andamento do projeto, e até mesmo em requisitos estabelecimentos por outros interessados. Na fase de negociação, estes conflitos de prioridade e alterações de requisitos devem ser convergidos entre todos os afetados.
- Especificação: Uma especificação de requisitos é a documentação da necessidade e a solução da mesma, de maneira que as mesmas fiquem claras, objetivas e coerentes. A Especificação pode ser formada por documentos escritos em um modelo padronizado, gráficos, protótipos, fluxogramas, etc. Estes por sua vez, são chamados de Artefatos da Especificação de Requisitos.
- Validação: Os artefatos produzidos na etapa de especificação são entregues, avaliados e validados como os interessados.
- Gestão de Requisitos: A pesar de validados, é extremamente comum que ocorram no decorrer de um ciclo de vida de software, mudanças de requisitos já implementados. Estas alterações se dão por vários motivos, seja

por novas necessidades, problemas que já não ocorrem mais, alterações em normas e fluxos, etc.

É valido explicitar que as etapas apresentadas acima, são flexíveis, e muitas vezes ocorrem em sobreposição e repetições. Também pode-se afirmar que as etapas podem e devem ser adequadas para cada cenário de projeto de software, permitindo ainda que sejam incrementadas ou omitidas novas etapas. (PRESSMAN, 2011; CHRISTEL; KANG, 1992). Dentre a literatura cabe citar a norma ISO/IEC/IEEE 29148, que trata especificamente da gerência de requisites (ISO/IEC, 2011).

2.2 Rastreabilidade de Requisitos

Uma das principais atividades que acontecem no ciclo de vida de praticamente todo desenvolvimento de software, é a chamada evolução do software. Nesta etapa, são realizadas muitas modificações nas implementações e regras de negócio iniciais do sistema, para que o software continue atendendo as exigências do mercado e também as novas necessidades do cliente, quando inexoravelmente novas especificações e alterações de artefatos são geradas. (PRESMANN, 2011; HEINDL; BIFFL, 2005).

Em projetos de software, principalmente os de maiores escalas e formados por mais de uma equipe de desenvolvimento, é praticamente impossível lembrar-se da causa e do motivo de todos os requisitos, assim como suas ligações. Isso pode tornar o software altamente receptível a erros de quebras de requisitos e comportamentos. (PINHEIRO, 2004).

Para garantir que um novo comportamento a ser implementado não impacte em um requisito pré-estabelecido e validado, é necessário monitorar e gerenciar todas as especificações de requisitos, desde a fase de concepção, dando início ao que chamamos de Rastreabilidade de Requisitos. Segundo Gotel e Finkelstein (1994, p.01), podemos definir a Rastreabilidade de Requisitos como “a capacidade de descrever e seguir a vida de um requisito, de forma bidirecional, tanto para frente, quanto para trás”.

A rastreabilidade pode auxiliar na elicitação de novos requisitos, no que se diz respeito ao impacto que os mesmos irão trazer caso implementados (pré-rastreabilidade), ou então para obter informações a respeito de seu uso, e o que levou a funcionalidade a estar com o comportamento atual (pós-rastreabilidade). (PINHEIRO, 2004).

As vantagens da rastreabilidade de requisitos são inúmeras, e o mais interessante é que não é somente o engenheiro de requisitos que usufrui destes benefícios, mas sim todos os envolvidos no projeto. Por exemplo, para o Gerente de Projetos a rastreabilidade de requisitos além de permitir uma visão mais clara do andamento do projeto, torna possível prever futuros conflitos e estimar o impacto que um novo requisito irá gerar ao ser implementado. Já o Cliente, possui uma melhor visibilidade referente às suas priorizações, e permitir a validação de que suas necessidades estão sendo cumpridas ou não. (PINTO, et al, 2003; HARRINGTON; RONDEAU, 1993).

Atualmente existe uma vasta quantidade de métodos, modelos e ferramentas para realizar a rastreabilidade de requisitos, como por exemplo:

- Esquema de Referências Cruzadas: A técnicas de referências cruzadas, foi a primeira técnica utilizada na rastreabilidade de requisitos, e serviu como base para todas as demais. Basicamente, em cada requisito era mencionado uma documentação, objeto ou outro requisito, geralmente com a frase: "Ver seção/requisito/documentação X". (AIZENBUD-RESHEF, et al, 2006).
- Matriz de Rastreabilidade: A Matriz de Rastreabilidade nos ajuda a criar facilmente links entre os requisitos. A técnica consiste em listar todos os requisitos em uma matriz, tanto na vertical quanto na horizontal. O vínculo entre os requisitos se dá ao sinalizar a intersecção dos mesmos. (WIERINGA, 1995).
- Dependência de Palavra Chave: Utilizada para verificar impactos, esta técnica consiste em comparar as expressões entre as entidades já criadas, a fim de resultar em uma lista com as possíveis entidades que serão impactadas. (JACKSON, 1991).

É interessante ressaltar, que a preocupação e aplicação destas técnicas tiveram origem nos anos 70, com o surgimento do método de referências cruzadas (AIZENBUD-RESHEF, et al, 2006, EVANS, 1989), e partem do fundamento de relacionamento, onde são criados vínculos entre um ou mais requisitos, artefatos, grupo de entrega de valor, cliente, tipo de requisito, funcionalidades e módulos do sistema, versões, criticidade entre outros. (GOTEL; FINKELSTEIN, 1994; WIERINGA,1995).

Contudo, são perceptíveis alguns problemas na aplicação de rastreabilidade de requisitos (principalmente quando o processo de rastreabilidade ainda não está maduro o suficiente), como por exemplo, a baixa adesão por parte dos desenvolvedores e a falta de recursos humanos para manter a documentação atualizada. Apesar disso, o custo benefício da rastreabilidade claramente se destaca e agrega valor, de acordo com o amadurecimento do processo. (WIERINGA, 1995).

2.3 Desenvolvimento Ágil

De forma generalizada, o atual cenário da economia está em constante mudança. Pelo fato de estar direta ou indiretamente presente na maioria dos setores e negócios existentes, o Software e seus processos de desenvolvimentos tiveram que se ajustar para entregar a máximo em um menor espaço de tempo possível, e no meio deste processo ainda ser possível realizar as alterações de requisitos que acabaram de ser implementados. (SOMMERVILLE, 2010)

A partir deste movimento, começaram a surgir então os primeiros métodos e práticas para acelerar o processo de desenvolvimento, denominados como métodos de desenvolvimentos ágeis. A origem se deu em fevereiro de 2001 quando dezessete renomados desenvolvedores, autores, e consultores da área de software se uniram na chamada "Aliança Ágil" para assinar o chamado "Manifesto para o Desenvolvimento Ágil de Software". Eles defendiam a entrega incremental de requisitos, separando o desenvolvimento em pequenas e motivadas equipes, sendo fomentadas somente pelos artefatos necessários. (FOWLER, 2016; HIGHSMITH, 2001; PRESSMAN, 2011).

Atualmente existem várias metodologias que seguem os princípios sugeridos pelo manifesto ágil, como por exemplo, o SCRUM que trabalha sobre os processos básicos da engenharia de software, passando desde os requisitos até a entrega em si, trabalhando em ciclos de desenvolvimento que variam de duas a quatro semanas, chamados de "Sprint". O ciclo de desenvolvimento se inicia com o planejamento no qual é definido o chamado "Sprint Backlog", que consiste nos requisitos que serão implementados no próximo Sprint. A partir deste ponto, o time de desenvolvimento realizada o planejamento técnico do Sprint Backlog, e o Sprint se inicia. Durante o ciclo de desenvolvimento, são realizadas reuniões diárias, com objetivo de remover possíveis impedimentos que os desenvolvedores possam estar encontrando em determinada tarefa.

Ao fim de cada Sprint, é realizada a chamada "Demo", que é o momento quando o time irá apresentar e entregar as funcionalidades do Sprint Backlog que se comprometeram entregar. (PRESSMAN, 2011).

Apesar de revolver uma série de problemas existentes em métodos de desenvolvimento tradicionais, o desenvolvimento com base em métodos ágeis também traz algumas dificuldades. Sendo preciso avaliar o grau de modelagem do software, comunicação e autogerenciamento da equipe, assim como a disponibilidade de tempo dos clientes interessados para sanar os possíveis impedimentos durante o processo de desenvolvimento. (PRESSMAN, 2011; GLAZER et al, 2008).

3. Desenvolvimento do Estudo de Caso

Atuando nacional e internacionalmente, a organização estudada oferece soluções para um mercado restrito e com rígidas legislações que requerem alta precisão, disponibilidade e confiabilidade.

Com ênfase na evolução do seu software como um produto e uma postura colaborativa entre os clientes, é frequente e extremamente natural à presença de alterações e customizações de funcionalidades compartilhadas, de acordo com a necessidade de cada cliente. Fazendo com que entre outros pontos, seja de suma importância garantir que um novo requisito de determinado cliente não divirja com os requisitos dos outros clientes.

A Figura 1 mostra um esquema simplificado sobre a evolução do produto e o compartilhamento de requisitos e funcionalidades entre clientes.

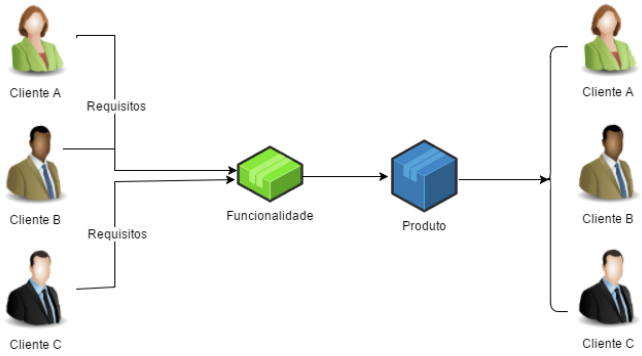

Figura 1 – Compartilhamento de requisitos entre clientes

3.1 Análise e Macro Discriminação do Cenário de Estudo

No cenário deste estudo, o processo de desenvolvimento se dá com o inicio de um projeto, partindo da premissa de que "Projeto", é caracterizado como um contrato firmado com um novo ou já existente cliente, para desenvolvimento e/ou evolução de um grupo específico de funcionalidades.

Neste ponto é levantado quais são as necessidades do cliente, e quais e como as mesmas serão atendidas, podendo acontecer via desenvolvimento de uma nova funcionalidade ou ainda a evolução ou adequação de uma funcionalidade já existente.

O primeiro desafio e/ou necessidade em relação aos requisitos do produto, que pode se sanar com alguns ajustes no processo, em conjunto com práticas de rastreabilidade de requisitos pode ser nominado como: "Validar a elegibilidade dos requisitos".

Com a definição de quais requisitos serão atendidos, dá-se início à elaboração de um documento constituído pela discriminação e esboço de cada funcionalidade. Nele, devem ser levados em consideração todos os pontos de atenção encontrados ao validar a elegibilidade de cada requisito, além de analisar minuciosamente cada comportamento que está sendo proposto, a fim de corroborar a consistência do mesmo em relação ao restante do sistema.

Isto leva à necessidade de validar o impacto destas alterações em relação às demais funcionalidades ou requisitos de outros clientes, notando também que "Assegurar a consistência entre requisitos" é outro ponto passível de melhoria.

Com os requisitos especificados, o time de desenvolvimento analisa e estima o esforço necessário para implementar cada um deles. As estimativas são realizadas em dias, contabilizando o alocamento de apenas um desenvolvedor, e servirão como base para precificação do projeto. É necessário ter disponível o máximo de informações sobre implementações similares já realizadas, implicações que a nova funcionalidade irá gerar, etc. Nesta fase pode se destacar o requisito "Garantir estimativas mais precisas".Com as estimativas realizadas, o documento de especificação de requisitos é enviado para aprovação por parte do cliente e passa a ser priorizado.

A partir desta etapa, as especificações de requisitos são divididas em várias tarefas, para que possam ser organizadas e disponibilizadas para desenvolvimento. As mesmas serão reestimadas pelo time de desenvolvimento em Story Points e incluídas dentro do ciclo de desenvolvimento chamado Sprint.

A divisão dos requisitos em tarefas menores torna muito mais ágil o desenvolvimento, porém dificulta o ato de acompanhar o andamento do projeto, permitindo elencar outra necessidade: “Mensurar o andamento do projeto”. Ao finalizar cada Sprint, as tarefas serão reunidas em uma única versão do sistema e disponibilizada para homologação pelo cliente.

A Figura 2 exemplifica de forma macro o processo acima descrito, destacando apenas as necessidades visíveis ao discriminar o mesmo. São elas:

- Validar a elegibilidade dos requisitos;
- Assegurar a consistência entre requisitos;
- Garantir estimativas mais precisas;
- Mensurar o andamento do projeto.

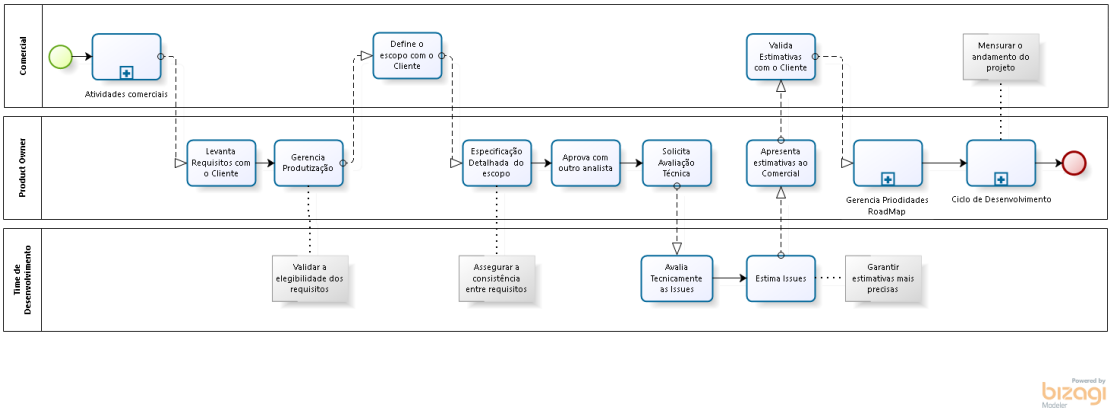

Figura 2 – Demonstração Macro do Processo de Desenvolvimento

3.2 Cenário de Rastreabilidade de Requisitos Encontrado

Apesar de o desenvolvimento ser baseado em uma metodologia ágil, a documentação sempre foi muito valorizada. Porém, como a quantidade de recursos humanos era desproporcional com as demandas dos clientes, era necessário muito planejamento e gerenciamento de tempo para garantir que todos os prazos fossem cumpridos. Uma documentação de requisitos completa, bem especificada e categorizada demanda algum tempo, assim, esta etapa do processo acabava não recebendo a devida atenção.

No cenário de estudo, o gerenciamento dos requisitos era realizado com o auxílio de um software e cada funcionalidade era separada em uma ou mais tarefas, divididas em:

- **Feature:** Tarefas criadas para descrever uma nova funcionalidade ou requisito que o sistema deve possuir. Normalmente são as tarefas que necessitam de um maior detalhamento, podem se tratar de funcionalidades novas, que não herdam conceitos que o sistema possui.
- **Bug:** São tarefas destinadas a correção de comportamentos não esperados no sistema. Geralmente esta tarefa é reportada pela equipe de suporte ou então pelo time desenvolvimento.
- **Task:** Tarefas técnicas que não geram valor direto, não são percebidos pelo cliente e principalmente não gera versão nova do produto.

A estrutura de cada tarefa é compartilhada entre todos os tipos (com algumas exceções). Ela compreende as seguintes informações:

- **Título:** Breve descrição sobre o objetivo da tarefa;
- **Descrição:** Especificação com um nível maior de detalhamento;
- **Situação:** Nova, Fechada, Atribuída ou Rejeitada;
- **Versão:** Versão do sistema onde a tarefa será aplicada;
- **Sprint**: Sprint onde a tarefa foi priorizada;
- **Tempo Estimado:** Estimativa do esforço necessário para concluir atarefa, definida pelo time de desenvolvimento;
- **Tempo Gasto:** Designado para registrar as horas reais gastas na tarefa.
- **Grupo Funcional:**A qual grupo funcional esta tarefa faz referência. Exemplo: Módulo de Cadastros;
- **Dono**: Designado para referenciar quem reportou a tarefa;
- **Fonte de Custo**: Utilizado para informar o cliente ou projeto relacionado a esta tarefa;
- **Atribuído Para**: Designado para referenciar o desenvolvedor responsável pela implementação da tarefa;
- **Fonte:** Campo exclusivo de tarefas de Bugs. É utilizado para informar quem está reportando o mesmo, podendo ser Suporte ou Desenvolvimento;
- **Severidade:** Campo exclusivo de tarefas de Bugs. Utilizado para informar a prioridade da tarefa com relação às demais. Podendo ser Alta, Média ou Baixa.
- **Subtarefas:** Utilizado para mencionar as subtarefas, caso esta tarefa necessite ser dividida em outras menores.
- **Tarefas Relacionadas:** Utilizada para mencionar as tarefas que podem ser impactadas direta ou indiretamente ao realizar a implementação da mesma.

Essa estrutura não permite uma rastreabilidade de requisitos efetiva. Ela oferece algumas informações sobre a fonte do requisito, os interessados e algum relacionamento entre funcionalidades impactadas, limitando no que diz respeito ao rastreio de cada requisito e a garantia de consistência entre os mesmos.

Na seção a seguir será descrito o processo de aplicação das técnicas de rastreabilidade de requisitos e os benefícios alcançados.

3.3. Evolução do Cenário de Rastreabilidade de Requisitos

Com o aumento no volume de projetos, tarefas, funções e até mesmo no quadro dos integrantes da equipe, o cenário de rastreabilidade em questão foi se mostrando insuficiente, tanto para a área operacional quanto para a área gerencial. Percebeu-se então a necessidade de uma reestruturação que iniciou com a análise do cenário atual, já apresentada.

3.3.1 Escopo da Evolução do Cenário

Por se tratar de um software já maduro, com diversas funcionalidades já implementadas e em produção, antes de aplicar alguma das práticas de rastreabilidade encontradas, reduzimos os escopo da reestruturação à novos requisitos, optando então pela não rastreabilidade ativa retroativa dos requisitos sem documentação, em outras palavras, requisitos sem documentação até este ponto, serão rastreados somente a partir do momento em que os mesmos sofrerem alguma alteração.

3.3.2. Prática de Rastreabilidade Adotada

Das inúmeras práticas de rastreabilidade existentes, cada uma com suas vantagens e desvantagens, com base nas necessidades da organização, foi necessário listar primeiro as premissas que a prática a ser adotada deve atender, sendo elas:

- Prática: A inserção de um novo requisito deve ser fácil e rápida;
- Ágil: Deve ser compatível com o processo de desenvolvimento ágil da empresa;
- Flexível: Deve suportar alteração, evolução, configuração e personalização.
- Verificável e Disponível: Deve permitir o fácil e rápido acesso à informação de determinado requisito ou funcionalidade, a qualquer interessado. Seja por parte de um Desenvolvedor, seja por parte de um Gestor de Projeto.

Com as premissas definidas, foi realizada a análise das práticas de rastreabilidade e optou-se pela técnica mais comum e que deu origem as demais: Referências Cruzadas.

Foram definidas algumas técnicas de referências cruzadas, como: hipertexto, tags e relacionamentos, assim,foram estabelecidos os elos e o tipo de relacionamento a ser utilizado para cada um.

Optou-se por esta técnica, justamente por ser de fácil manipulação, entendimento e por permitir o aproveitamento da estrutura atual de documentação de um requisito, caracterizando-se literalmente como uma evolução escalável do nosso cenário atual e não uma reformulação total.

Notou-se também, a necessidade de uma ferramenta para apoiar a aplicação desta técnica, para isso foi escolhido um software online, que cumpre as mesmas premissas acima listadas e também prove uma completa plataforma de controle de

desenvolvimento, conforme o fluxo ágil adotado pelo time. A ferramenta utilizada para tal finalidade foi o JiraAtlassian.

3.3.3. Níveis de Rastreabilidade

O cenário de estudo não possuía uma estrutura hierárquica para documentação de requisitos, centralizando tudo na criação de tarefas. Aproveitando a arquitetura do software de gerenciamento de requisitos adotado, foi implementada uma estrutura de três níveis, conforme mostra a figura 3.

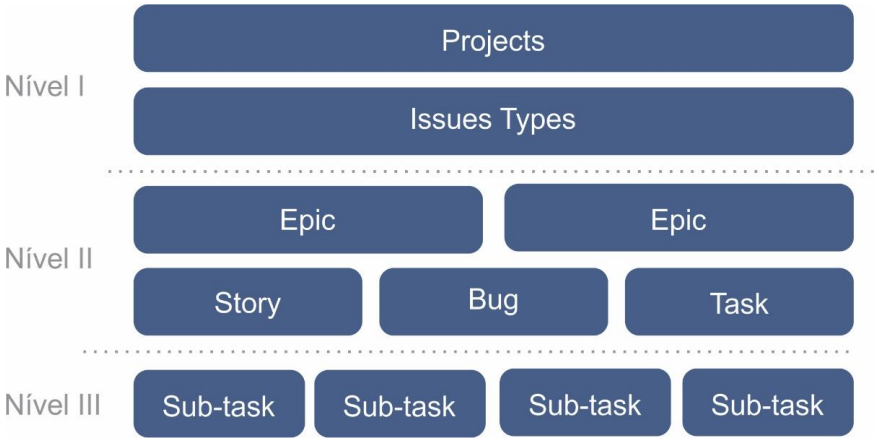

Figura 3 – Estrutura Hierárquica de Categorização de Requisitos

A estrutura descrita, é bastante usual nos projetos ágeis. Assim, tem-se:

Nível I

A empresa estudada possui o conceito de "projeto" para cada um dos produtos de seu portfólio, cada um destes projetos possui um ou mais times de desenvolvimento, trabalhando com requisitos de clientes distintos, ou do mesmo cliente, quando um requisito impacta em mais de um produto.

Dentro de cada projeto há as chamadas Issues, que são entregáveis parciais ou completos de um requisito que devem sempre entregar algum valor. Dentro da estrutura proposta, adotamos os seguintes tipos de tarefas, que veremos a seguir: Epic, Story, Bug, Task e Sub-task.

Nível II

Dentro do nível 2, as issues tem o seu tipo definido, podendo ser:

- **Epics** define uma funcionalidade que irá gerar um grande impacto, ou um grande esforço, que faça necessário à divisão em tarefas menores,

para que o valor seja entregue ao cliente. Geralmente cada item do roadmap do produto irá gerar um Epic.

- **Story:** são tarefas com objetivo de entregar valor.
- **Bug:** Toda issue com objetivo de corrigir um comportamento no software, seja de erro de programação ou um comportamento não condizente com o esperado pelo usuário.
- **Tasks:**quaisquer tarefa que não gere código, mas que se faça necessária para investigação ou estudo para resolução de um determinado problema.

Nível III

No último nível, temos as Sub-Tasks, que por sua vez são a menor unidade de trabalho dentro desta estrutura e estão diretamente ligadas a uma Story. Geralmente utilizada para separar o desenvolvimento em partes para que mais de um desenvolvedor possa codificar simultaneamente na mesma tarefa.

3.3.4. Rastreabilidade de Issues- Nível I

Seguindo o conceito de rastreabilidade por referências cruzadas, cada parâmetro ou forma de categoria dentro de uma determinada tarefa são considerados elos, que podem ser utilizados como referência para outras e até como métricas para relatórios. De uma forma geral, todas as issues detêm alguns elos em comuns e outros particulares. Os elos padrões, estabelecidos para qualquer tipo de issue são:

Project: relacionado aos produtos internos da empresa.

IssueType: Pode ser categorizado em: Epic, Story, Bug, Task ou Sub-task.

Summary: Utilizado para descrever resumidamente o objetivo da tarefa.

Priority: Neste campo, deve ser informado a criticidade da tarefa em relação às demais. Os valores possíveis são: *Highest, High, Medium, Low*, e *Lowest.*

Version: versão em que as alterações implementadas estarão disponíveis.

Assignee: Desenvolvedores responsáveis por entregar a implementação desta tarefa.

Reviewer: Desenvolvedores responsáveis por testar e revisar o código gerado.

Reporter: Referente ao responsável e criador da tarefa.

Description: Especificações de requisitos, contém requisitos funcionais, regras de negócio, critérios de aceite, etc. Também casos de testes que falharam e/ou evidências de bugs.

Attachement: Anexos protótipos, documentação adicional, evidências, ou qualquer outro tipo de artefato necessário.

LinkedIssue: Este é um dos campos mais importantes na documentação, no que diz respeito à rastreabilidade por referências. É utilizada a técnica de relacionamento entre

os requisitos, podendo ser dos seguintes tipos:Relates to, Isfixedby, Duplicates, Isblockedby, Blocks, Fixes, Iscausedby.

Labels: Outro campo muito importante para a rastreabilidade dos requisitos por referências cruzadas, porém desta vez utilizando a técnica de tags. Este campo é utilizado para informar qual o time de desenvolvimento será responsável pela implementação da tarefa.

Sprint: Utilizado para informar em qual Sprint a issue foi realizada.

Target Customer: Permite sinalizar quais são os clientes alvos desta implementação.

Story Points: Estimativa em pontos realidade pelos times de desenvolvimento durante o planejamento do Sprint ou do Release.

Epic Link: Campo aplicável a issues do tipo Story, Task e Bug. É utilizado para vincular qual é o Epic pai das tarefas.

Created Date: não é editável. Apresenta a data de criação da issue.

Updated Date: Apresenta a data da última atualização de qualquer um dos campos.

Workflow State: Representa o estado na tarefa em relação ao fluxo de trabalho, podendo ser:ToDo, In Progress, PullRequest, Test, Reject, Done.

State: uma issue pode estar em dois estados: Ok, Impedida.

Time: Tempo gasto pelo desenvolvedor para implementar e entregar a issue.

Comments: Campo no formato de fórum, que permite a inserção de comentários por parte dos interessados.

History: É mantido um histórico com todas as alterações realizadas em qualquer campo, informando data, autor, alteração e informação original que foi alterada.

3.3.5. Rastreabilidade de Epics, Stories e Tasks – Nível II

Além dos campos padrões, há para os Epics, Stories e Tasks, alguns outros campos próprios, a fim de auxiliar na categorização e rastreio dos requisitos. Sendo eles:

Epic Name: identificação do Epic.

Financier implementation: tipo de financiamento de cada atividade. As diversas modalidades de financiamento são: a) Venda de horas (projeto), b) cessão de horas (projeto, não previsto), c) Requisito da Licença (projeto, previsto), d) Requisito de piloto (mercado), e) Investimento de Produto (mercado), f) Investimento de Produto (infraestrutura).

Project (Funding): o cliente responsável por financiar a implementação.

Productization Factor: O fator de produtização, é o percentual do requisito a ser implementado com relação ao interesse do mercado. Ou seja, se o requisito desenvolvido pode atender outros clientes ou é algo muito específico.

Technology Upgrade fator: O Fator de Atualização Tecnológica é o percentual de melhorias e atualizações realizadas no código de uma funcionalidade já existente ao ser incluído um novo requisito.

Issues in Epic: relaciona o Epic às issues filhas, que serão utilizadas para implementar os requisitos do mesmo.

3.3.6. Rastreabilidade de Bugs

Para os Bugs existem alguns outros campos próprios, a fim de auxiliar na categorização e rastreio dos requisitos. Sendo eles: Ticket Number, Root cause,Font;

3.3.7. Documentação de Entregas

Ainda como ferramenta para rastreabilidade e requisitos, são tomadas duas ações no final de cada iteração de desenvolvimento, ao final de cada Sprint e ao final de cada Release (Entrega formada por até quatro Sprints). As açõessão:

a) **Atualização do Manual de Funcionalidade do Sistema**

É mantido em uma plataforma com acesso compartilhado, um manual contendo todas as funcionalidades do sistema, dividido em áreas, módulos e funcionalidades.

Como a evolução do produto recebe como insumo novos requisitos ou alteração dos requisitos existentes para uma mesma funcionalidade, é imprescindível que seja mantido um histórico de alterações para que possa ser utilizado como ferramenta para a validação de novos requisitos, e saber se os mesmos não irão impactar em requisitos de outros clientes.

Para isto, são vinculados na página de cada funcionalidade do manual, toda e qualquer issue que impactou, direta ou indiretamente, os requisitos desta página. As seguintes informações são utilizadas:

- **Código:** Código da issue contendo um **link** que leva o usuário para a tela de detalhes da tarefa.
- **Tipo:** Epic, Story, Bug ou Task;
- **Título;**
- **Status:** ToDo, In Progress, Pull Request, Test, Rejected ou Done.

b) Anotações de Entregas de Versão: É mantido em uma plataforma com acesso compartilhado, um relatório separado por versão, todas as issues que essa versãopossui, chamado Release Notes. Além do número e data de lançamento da versão, este relatório contém as seguintes informações de cada issue:

- **Tipo:** Epic, Story, Bug ou Task;
- **Código:** Código da issue contendo um **link** que leva o usuário para a tela de detalhes da tarefa.
- **Descrição:** Descriçãodos requisitos implementados ou alterados;
- **Implantação:** ações necessárias para que o requisito esteja disponível para o cliente.
- **Detalhes Adicionais:** comentários relevantes para a issue em questão, geralmente com pontos de atenção sobre mudanças críticas realizadas.
- **Funcionalidades Impactadas:** relaciona as páginas do manual de cada funcionalidade que foiimpactada com esse novo requisito.

3.3.8. Disponibilidade, Verificação e Extração de Requisitos

Com a estrutura de rastreabilidade adotada, em conjunto com a ferramenta online para gerenciamento dos requisitos, a vinculação com o manual e o release notes, ficou de fácil e rápido acesso a todos, a verificação dos detalhes de cada requisito assim como suas dependências e elos. Em cada Issue os requisitos podem ser pesquisados utilizando qualquer um dos parâmetros vistos anteriormente, para qualquer tipo de issue. Por exemplo é possível listar as issues do tipo Bug com o Status Done. A partir dos detalhes de cada issue criada, é possível verificar seus elos e se há ou não algum relacionamento com outros Requisitos.

Além da pesquisa de requisitos, com a inclusão dos novos elos e estrutura adotada, é possível extrair informações importantes para geração de relatórios estatísticos paras as equipes de Desenvolvimento, Gestão de Projetos e Negócios da Empresa.

4. Considerações finais

Com o amadurecimento de uma organização, seja no que diz respeito a seus processos, número de funcionários, clientes ou projetos, surgem também várias necessidades que anteriormente eram de fácil controle e gerenciamento.

Essa situação também acontece nas organizações de desenvolvimento de software, em particular empresas que possuem um produto que vive em constante evolução e alteração. Essa solução, baseada na necessidade de vários clientes, torna difícil gerenciar as expectativas, prazos, requisitos e alterações de requisitos.

O principal motivador da implantação de uma estrutura de rastreabilidade de requisitos é facilitar o acesso à informação a respeito de determinado comportamento e histórico evolutivo de determinada funcionalidade.

Este trabalho apresentou como a evolução da estrutura de rastreabilidade de requisitos no cenário estudado supriu as necessidades que o crescimento rápido da empresa fez emergir. A partir dos novos elos, categorias e vínculos criados com cada requisito, em conjunto com a correta documentação das funcionalidades, foi possível realizar um controle maior no que diz respeito à validação da elegibilidade dos requisitos, andamento do projeto, estimativas mais precisas por parte dos

desenvolvedores e proporcionando uma visão mais ampla sobre onde estava sendo empregado mais esforço e se esse esforço estava gerando receita ou prejuízo.

Vale ressaltar que uma característica importante no gerenciamento de requisitos diz respeito à aderência com a realidade e as mudanças. Assim, sempre que algum requisito sofrer alterações deve verificar-se se os relacionamentos ainda estão corretos ou se precisam de alterações.

Documentação, Gerenciamento e Rastreabilidade das mudanças são necessários para qualquer tipo de projeto iterativo e evolutivo, podendo muitas vezes ser o fator decisivo entre o sucesso e o fracasso de uma organização.

5. Referências

AIZENBUD-RESHEF, N., et al. Model traceability, IBM systems journal, vol. 45, nº 3, p.515-526, 2006.

CHRISTEL, Michael G., KANG, KyoC. Issues in Requirements Elicitation, Requirements Engineering Project, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, Technical Report CMU/SEI-92-TR-012, 1992.Disponível em: <http://resources.sei.cmu.edu/library/asset-view.cfm?AssetID= 12553>. Acessoem: 08 de julho de 2016.

EVANS, Michael W. The Software Factory: A Fourth Generation Software Engineering Environment, John Wiley & Sons, New York, USA, 1989.

FOWLER, Martin, Highsmith, Jim.The Agile Manifesto. In: Software Development Magazine, august 01, 2001.Disponível em: <http://jimhighsmith.com/article-4/>. Acessoem: 08 de Julho de 2016.

GLAZER, Hillel, et al. CMMI or Agile: Why Not Embrace Both! Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, TechnicalReport CMU/SEI-2008-TN-003, 2008.Disponível em: <http://resources.sei.cmu.edu /library/asset-view.cfm?assetid=8533/>. Acesso em: 08 de julho de 2016.

GOTEL, Orlena C. Z., FINKELSTEIN, Anthony C. W.. An Analysis of the Requirements Traceability Problem, Imperial College of Science, Technology & Medicine, London, In: 2ª International Conference on Requirements Engineering (ICRE 1994), IEEE Computer Society Press, Colorado Springs, Colorado, USA, 18–22 Abril 1994, p.94–101. Disponível em: <http://ollygotel.com/downloads/an-analysis-of-the-requirements-traceability-problem.pdf>. Acessoem: 08 de julho de 2016.

HANS VAN VLIET, J.C. Software engineering: Principles and Practice, Wiley, 2007.

HARRINGTON, Gale Alicia, RONDEAU, Kathleen Marie. An Investigation of Requirements Traceability to Support Systems Development, Master's Thesis, Naval

Postgraduate School, Monterey, California, 1993.Disponível em: <http://calhoun.nps.edu/handle/10945/39948>. Acessoem: 08 de julho de 2016.

HEINDL, Matthias, BIFFL, Stefan. A case study on value-based requirements tracing. In: 10th European Software Engineering Conference Held Jointly with 13th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Lisbon, Portugal, p.60-69. Disponível em: <http://publik.tuwien.ac.at/files/pub-inf_3349.pdf>. Acesso em: 08 de julho de 2016.

IEE COMPUTER SOCIETY, SWEBOK: Guide to the Software Engineering Body of Knowledge, Version 3, IEEE Computer Society, 2004.

IEE STANDARD 610.12-1990, IEE Standard Glossary of Software Engineering Terminology, IEEE Computer Society, USA, 1990.

ISO/IEC/IEEE 29148. Systems and software engineering — Life cycle processes — Requirements engineering. Genebra, Suiza, 2011. Disponível em: <https://www.iso.org/obp/ui/#iso:std:iso-iec-ieee:29148:ed-1:v1:en>. Acesso em: 20 de setembro de 2017.

JACKSON, J. A keyphrase based traceability scheme. In: Tools and Techniques for Maintaining Traceability During Design, IEE Colloquium, London, 1991.

NUSEIBEH, Bashar, EASTERBROOK, Steve. Requirements engineering: A roadmap, London, 2000.In:22º International Conference on The Future of Software Engineering, Limerck, Ireland, 2000. P.35-46. Disponível em: <https://www.cs.toronto.edu/~sme/papers/2000/ICSE2000.pdf>. Acesso em: 08 de julho de 2016.

PINHEIRO, Francisco A. C. RequirementsTraceability. In: Perspectives on Software Requirements, ed. Dordrecht, The Netherlands: Kluwer Academic Publishers, p. 91-113, 2004.

PINTO, Rosa Candida, et al. A Process for Requirements Traceability in Agent Oriented Development. In: Software Engineering for Large-Scale MultiAgent Systems, Springer-Verlag, 2003, p. 52-72. Disponível em:<http://wer.inf.puc-rio.br/WERpapers/artigos/artigos_WER05/rosa_pinto.pdf>. Acesso em: 08 de Julho de 2016.

PRESSMAN, Roger S.. Engenharia de software: Uma abordagem profissional. 7ª edição. Porto Alegre: AMGH, 2011.

SOMMERVILLE, Ian. Software engineering: 9th edition, Pearson Education, 2010.

WIEGERS, Karl E. Software Requirements, Second Edition. 2nd ed. Microsoft Press. 2003.

WIERINGA, Roel. An Introduction to Requirements Traceability, Electrical Engineering, Mathematics and Computer Science (EEMCS), VrijeUniversiteit Faculty of Mathematics and Computer Science, VrijeUniversiteit, Amsterdam, 1995. Disponível em: <http://doc.utwente.nl/76217/>. Acesso em: 08 de julho de 2016.