UNIVERSIDADE FEDERAL DA FRONTEIRA SUL
CAMPUS CHAPECÓ
CURSO DE CIÊNCIA DA COMPUTAÇÃO

MELHORIA NO PROCESSO DE ESPECIFICAÇÃO E VALIDAÇÃO DE REQUISITOS EM PROJETOS ÁGEIS

LEDUAN RABAIOLI

CHAPECÓ
2018

LEDUAN RABAIOLI

MELHORIA NO PROCESSO DE ESPECIFICAÇÃO E VALIDAÇÃO DE REQUISITOS EM PROJETOS ÁGEIS

Trabalho de conclusão de curso de graduação apresentado como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.
Orientador: Prof. Dra. Raquel Aparecida Pegoraro

CHAPECÓ
2018

Rabaioli, Leduan

Melhoria no processo de especificação e validação de requisitos em projetos ágeis / por Leduan Rabaioli. – 2018.
81 f.: il.; 30 cm.

Orientador: Raquel Aparecida Pegoraro
Monografia (Graduação) - Universidade Federal da Fronteira Sul, Ciência da Computação, Curso de Ciência da Computação, SC, 2018.

1. Métodos Ágeis. 2. Engenharia de Requisitos. 3. Levantamento de Requisitos. I. Pegoraro, Raquel Aparecida. II. Título.

E-mail: leduancbr@gmail.com

LEDUAN RABAIOLI

MELHORIA NO PROCESSO DE ESPECIFICAÇÃO E VALIDAÇÃO DE REQUISITOS EM PROJETOS ÁGEIS

Trabalho de conclusão de curso de graduação apresentado como requisito para obtenção do grau de Bacharel em Ciência da Computação da Universidade Federal da Fronteira Sul.

Orientador: Prof. Dra. Raquel Aparecida Pegoraro

Aprovado em: 04/07/2018

BANCA EXAMINADORA:

Dra. Raquel Aparecida Pegoraro - UFFS

Dra. Graziela Simone Tonin - UFFS

Me. Gláucio Adriano Fontana - UFFS

AGRADECIMENTOS

Primeiramente agradeço imensamente à Deus por me conceder saúde, força e persistência para que eu pudesse realizar este trabalho. Agradeço em especial aos meus pais, Adair e Cheila, meu avô Edecir e minha avó Inedina que desde sempre me motivaram e prestaram todo o apoio possível para que eu pudesse concluir mais esta etapa de minha vida. Agradeço igualmente a todos os demais familiares e amigos que de alguma forma contribuiram e torceram por mim durante a minha graduação.

Agradeço aos meus professores que se dedicaram a ensinar e compartilhar o seu conhecimento, em especial a minha orientadora Profa. Dra. Raquel Aparecida Pegoraro, pela disponibilidade, esforço, paciência e por me guiar tão bem durante a execução deste trabalho. Agradeço também a banca examinadora deste trabalho composta pela Dra. Graziela Simone Tonin e pelo Me. Glaucio Adriano Fontana pelas contribuições para a melhoria deste trabalho.

RESUMO

O sucesso do desenvolvimento de software depende diretamente de um bom levantamento de requisitos, um entendimento correto do escopo evitará retrabalho e consequentemente aumento de prazo e custo. Nos métodos ágeis a engenharia de requisitos utiliza-se de técnicas baseadas na participação ativa e contínua do cliente para levantar e elicitar os requisitos. Embora os métodos ágeis tenham surgido como uma alternativa para minimizar os problemas de desenvolvimento de software, muitos problemas ainda são enfrentados nas etapas de especificação e validação de requisitos. Este trabalho tem como objetivo propor melhorias no processo de especificação e validação de requisitos nos métodos ágeis com o intuito de reduzir problemas relacionados a essas etapas do projeto. Para alcançar o objetivo será realizada a identificação dos problemas que ocorrem na especificação e validação de requisitos, propondo uma melhoria no processo com base no modelo V, sugerindo um modelo de documentação para auxiliar na especificação e validação de requisitos. Com o resultado, espera-se auxiliar as empresas que adotam métodos ágeis para melhorar estas etapas do processo de um projeto de software.

Palavras-chave: Métodos Ágeis. Engenharia de Requisitos. Levantamento de Requisitos.

ABSTRACT

The success of software development depends directly in a good requirements gathering, an correct understanding of the scope will avoid rework and consequently the increase of time and cost. In the agile methods requirements engineering uses techniques based on the active and continuous participation of the client to gather and elicit the requirements. Although the agile methods have arisen as an alternative to minimize the problems of software development, many problems still are faced in the steps of requirements specification and validation. This paper has as goal to propose improvements in the requirements specification and validation in the agiles methods in order to reduce problems related to these steps of the project. To achieve the objective will be performed the identification of the problems that occur in the specification and validation of requirements, proposing an improvement on the process based on the V model, suggesting an documentation model to help in the specification and validation of requirements. With the result, it is expected to help the companies that adopt agile methods to improve these steps of the process of an software project.

Keywords: Agile methods. Requirements Engineering. Requirements gathering .

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE APÊNDICES

LISTA DE ABREVIATURAS E SIGLAS

BDD	*Behavior Drive Development*
PO	Product Owner
RN	Regras de negócio
RNF	Requisitos não Funcionais

SUMÁRIO

1 INTRODUÇÃO

O levantamento de requisitos é uma etapa crucial para o sucesso do desenvolvimento de software, um bom entendimento do problema proposto pelo usuário evitará retrabalho. Segundo Sommerville [48], os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições ao seu funcionamento.

Sabe-se que inúmeros problemas ocorrem durante o desenvolvimento de um software, dificuldades com cumprimento de prazos [38], elevação de custos [11] e manutenção do sistema pelas mudanças de requisitos [48], são alguns dos problemas comuns. Grande parte desses problemas ocorrem devido a um mau levantamento de requisitos. Na abordagem tradicional a engenharia de requisitos é baseada em documentos formais, ocorre no início do projeto e não aceita mudanças nos requisitos nas fases posteriores do projeto [10].

Com o propósito de amenizar vários problemas em relação ao desenvolvimento de software surgiram os métodos ágeis como uma alternativa aos métodos tradicionais. Segundo Sommerville [48], os métodos ágeis nasceram da insatisfação com os métodos tradicionais e da necessidade de desenvolver softwares cada vez mais adaptáveis às inovações tecnológicas e ao mercado competitivo.

Por mais que o uso dos métodos ágeis tenha amenizado os problemas de desenvolvimento de software, ainda existem muitas dificuldades enfrentadas pelos desenvolvedores [26]. Embora várias técnicas como histórias de usuários, priorização de requisitos e comunicação cara a cara são utilizadas para levantar e elicitar requisitos, é notável que existem muitos problemas e limitações para a gerência de requisitos nos métodos ágeis [10]. Os métodos ágeis dependem da colaboração e iteração do cliente [1], porém esse processo não é tão simples de ser executado.

Alguns dos problemas mais comuns incluem a dificuldade de atender as expectativas do cliente [10], insuficiência de documentação para implementação e manutenção do software [10], histórias de usuários inadequadas e com baixo nível de detalhes [20], requisitos não funcionais que muitas vezes são ignorados[11], e a baixa qualidade no produto desenvolvido pela falta de clareza entre o problema e a solução proposta [4].

Um desenvolvimento ágil de sucesso depende da disponibilidade do cliente e da interação entre o cliente e os desenvolvedores, especialmente durante os estágios iniciais do projeto [11], porém sabe-se que nem sempre é possível ter um contato próximo e frequente com o

cliente, assim dificultando o entendimento do problema. Se os representantes do cliente não existirem, a avaliação e priorização dos requisitos é realizada pelos desenvolvedores que podem não conhecer o mercado [33].

A documentação de requisitos nos métodos ágeis depende do conhecimento tácito e da comunicação frequente para apoiar explicitamente a partilha de conhecimentos e decisões [11]. A escassez da documentação escrita pode dificultar a compreensão e dificultar futuras mudanças por não fornecer um detalhamento do problema. Sempre que houver um erro de comunicação devido a mudanças de requisitos, indisponibilidade de representantes do cliente ou houver um projeto complexo, surgirão problemas. Torna-se difícil abordar tal situação com pouca ou nenhuma documentação [19].

A qualidade do projeto pode ficar comprometida pela falta de clareza da real necessidade do cliente, um mau planejamento inicial causará futuros problemas resultando em uma crescente dívida técnica. Segundo Cao e Ramesh [11], as práticas de requisitos nos métodos ágeis podem conduzir uma arquitetura inadequada ou imprópria devido ao curto período de tempo de planejamento. A arquitetura escolhida pela equipe em fases iniciais do projeto torna-se inadequada em estágios posteriores com novas exigências [38].

Partindo dessa premissa de que a engenharia de requisitos no desenvolvimento de software utilizando métodos ágeis enfrenta muitas dificuldades, este trabalho propõe investigar formas de minimizar esses problemas e melhorar questões como a especificação e validação de requisitos.

1.1 Questão de Pesquisa

Para este trabalho foi definida como questão principal (QP) de pesquisa:

QP. Como melhorar o processo de especificação e validação de requisitos nos métodos ágeis?

1.2 Objetivos

Nesta seção são apresentados os objetivos gerais e específicos da pesquisa.

1.2.1 Objetivo Geral

Propor melhorias no processo de especificação e validação de requisitos nos métodos ágeis com base no modelo V, na tentativa de auxiliar as equipes que utilizam scrum a reduzir

problemas relacionados a essas etapas do projeto.

1.2.2 Objetivos Específicos

- Identificar os aspectos positivos e os problemas que ocorrem na especificação de requisitos nos métodos ágeis.
- Identificar os aspectos positivos e os problemas que ocorrem na validação de requisitos nos métodos ágeis.
- Mapear práticas que permitam auxiliar na melhoria da especificação e validação de requisitos nos métodos ágeis, relacionando quais podem ser aplicadas em cada etapa do processo de desenvolvimento ágil.
- Propor a melhoria do processo de especificação e validação de requisitos baseando-se no modelo V, definindo um fluxo de atividades que possa ser aplicado de forma visual em um quadro kanban.
- Propor um modelo de documentação para auxiliar na especificação e validação de requisitos nos métodos ágeis através do BDD e outras documentações adicionais, sem perder as suas características de agilidade.

1.3 Justificativa

Foram identificados estudos sobre a engenharia de requisitos nos métodos ágeis, mas os estudos encontrados são revisões gerais que não tratam especificamente do entendimento das etapas da especificação e validação de requisitos.

Este trabalho se justifica por uma análise e identificação de aspectos positivos assim como a identificação de problemas relacionados a compreensão dos requisitos nas etapas de especificação e validação de requisitos nos métodos ágeis , aprofundando e sugerindo melhorias nas técnicas utilizadas pela engenharia de requisitos nos métodos ágeis na tentativa de minimizar as dificuldades encontradas durante estas etapas do processo, contribuindo assim para a melhoria da qualidade.

Uma das estratégias de realizar a melhoria do processo é através do modelo V, este modelo identifica que existem diferentes tipos de testes a serem realizados durante o processo

de desenvolvimento. Este modelo nos permite identificar problemas logo nas etapas iniciais do desenvolvimento e realizar testes diferentes para cada etapa.

Na tentativa de propor um modelo de melhoria de documentação será utilizado o BDD (Behavior Driven Development), essa técnica ajuda as equipes na compreensão e na construção das histórias de usuários, garantindo uma documentação mínima e permitindo a realização de testes a partir de um conjunto de cenários facilitando o entendimento dos requisitos e outros documentos necessários.

1.4 Estrutura do trabalho

Este trabalho está estruturado em 6 capítulos, no capítulo 1 foi apresentada a introdução, definida a questão de pesquisa, os objetivos e a justificativa. No capítulo 2 é relatada a fundamentação teórica. No capítulo 3 é exposta a metodologia utilizada neste trabalho. No capítulo 4 é exibido o primeiro resultado deste trabalho através da revisão de literatura. No capítulo 5 é apresentado o processo proposto, a documentação sugerida e a descrição da aplicação prática do processo em um ambiente real. No capítulo 6 são realizadas as considerações finais. Após isso são apresentadas as referências utilizadas neste trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 Engenharia de Requisitos

Os requisitos constituem a primeira fase do ciclo de vida do desenvolvimento de software [28]. São descrições sobre as implementações que devem ser feitas no software, refletindo as necessidades dos clientes para sistemas que servem para finalidades específicas. Em alguns casos, requisito é apenas uma declaração em alto nível de um serviço ou uma restrição que o sistema deve oferecer [48]. Entender os requisitos de um projeto é uma tarefa difícil [35].

A engenharia de requisitos é definida como um processo de descobrir, analisar, documentar e verificar as funções e restrições do sistema [48]. Na visão de Pressman [35] a engenharia de requisitos permite que examinemos a situação do trabalho a ser realizado, as necessidades específicas que o projeto deve atender, a prioridade das atividades que devem ser realizadas e as funções e comportamentos que terão um impacto profundo no projeto. O objetivo principal do processo de engenharia de requisitos é fornecer um modelo da necessidade do cliente, em uma declaração clara, consistente, precisa e inequívoca do problema a ser resolvido [28].

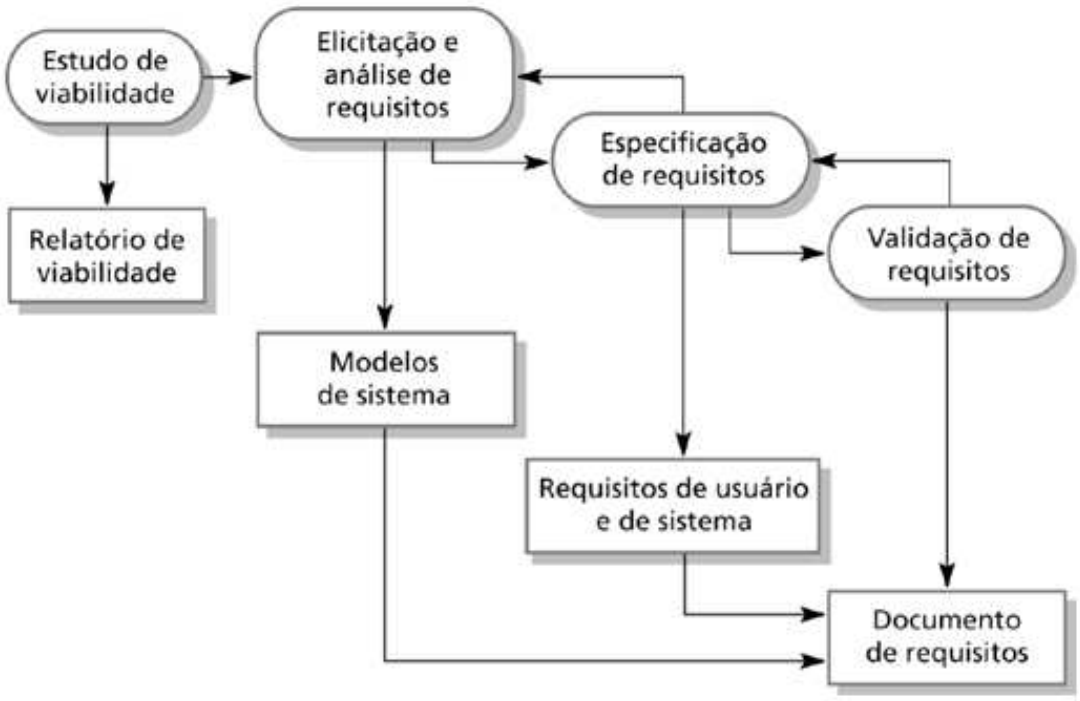

Figura 2.1 – Processo de engenharia de requisitos [47]

A figura 2.1 ilustra o processo como uma atividade de três estágios e as principais atividades da engenharia de requisitos. Cada atividade possui as seguintes características:

- Estudos de viabilidade: Visa avaliar a viabilidade do projeto, consiste de um conjunto preliminar de requisitos de negócios, um esboço da descrição do sistema e como o sistema pretende apoiar os processos do negócio. O estudo de viabilidade foca em descobrir se o sistema a ser implementado contribui para os objetivos gerais da organização, se o sistema pode ser desenvolvido com a tecnologia disponível e dentro das restrições de prazo e custo e se o sistema pode ser integrado com outros sistemas já implantados. Esse estudo pode ser realizado através do contato com gerentes de departamento em que o sistema será usado, especialistas em tecnologia, ou usuários finais do sistema [47].

- Elicitação e análise de requisitos: Nesta fase os engenheiros de software trabalham com o cliente e com o usuário final do sistema na tentativa de obter um domínio sobre o que a aplicação deve fazer, quais serviços o sistema deve fornecer, o desempenho esperado e as restrições. Neste processo são realizadas algumas atividades como a coleta dos requisitos, classificação dos requisitos, priorização dos requisitos e documentação dos requisitos [47].

- Especificação de requisitos: A especificação de requisitos reúne informações sobre o sistema proposto e os existentes para obter os requisitos de usuário e de sistema, tem o objetivo de comunicar os requisitos entre os desenvolvedores e os stakeholders. As fontes de informações incluem documentação e interação com os stakeholders que pode ser por meio de entrevistas, observações, cenários e protótipos [47].

- Validação de requisitos: A validação de requisitos tem o objetivo de mostrar que os requisitos realmente definem o sistema desejado pelo usuário. Durante esta etapa é verificado se o requisito é válido, se o requisito é consistente, se o requisito está completo, se o requisito pode ser implementado pela arquitetura disponível e dentro do prazo e custo estipulado, e se o requisito pode ser verificável através de testes [47].

2.2 Métodos Ágeis

Os métodos ágeis são abordagens contemporâneas para o desenvolvimento de software, baseando-se na colaboração com o cliente, no trabalho em equipe, no desenvolvimento iterativo incremental e na adaptação às mudanças [40]. Surgiram como uma maneira inovadora e diferenciada para auxiliar na minimização dos problemas de desenvolvimento. A definição

oficial do desenvolvimento ágil surgiu através de um manifesto denominado de manifesto ágil, que foi publicado em 2001 por um grupo de 17 especialistas de software que se reuniram com a intenção de encontrar maneiras de minimizar os problemas de desenvolvimento de software [1].

Na tentativa de atender as dificuldades dos desenvolvedores, os métodos ágeis usam uma abordagem baseada sobre a rápida iteração de todo o desenvolvimento de software, desde a elicitação de requisitos até a liberação de uma nova versão do software [37]. Os métodos ágeis são mais adequados ao desenvolvimento onde os requisitos mudam rapidamente durante o processo de desenvolvimento, pois comprometem-se a entregar o software rapidamente e em funcionamento, assim o cliente por sua vez pode propor mudanças e novos requisitos podem ser adicionados ou removidos do projeto. Um desenvolvimento ágil tem por objetivo reduzir a burocracia do projeto, evitando o excesso de documentação e possíveis retrabalhos [48]. O manifesto ágil possui quatro valores fundamentais [1]:

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

Embora os itens à direita sejam considerados importantes, deve-se valorizar mais os itens à esquerda.

2.3 BDD - Behavior Driven Development

O Behavior Driven Development (BDD) é uma técnica de desenvolvimento orientada a comportamento que surgiu na tentativa de auxiliar as equipes de desenvolvimento a construir e entregar um software de qualidade e de forma rápida, descrevendo como o sistema deve se comportar com uma linguagem natural [46]. A linguagem natural garante uma compreensão comum do sistema a ser desenvolvido entre todos os envolvidos do projeto [17].

Dan North [34] declara que ao usar e tentar ensinar a prática do TDD (Test Driven Development) em diferentes projetos encontrava problemas em relação ao entendimento da técnica, os desenvolvedores não tinham a noção de como começar a realização dos testes, o que

deveriam testar, quando testar, o que chamar de testes e como entender por que um teste falha. Em resposta a esses problemas North apresentou o BDD.

O TDD (Test Driven Development) é uma técnica de desenvolvimento orientada a testes, baseada em pequenos ciclos de desenvolvimento e práticas de escrita de testes automatizados que são realizados antes da codificação [27]. Um código de uma funcionalidade é desenvolvido juntamente com um teste para esse incremento, você só codifica uma próxima etapa do desenvolvimento depois que o código inicial passar no teste [48].

O BDD ajuda as equipes a concentrar seus esforços na identificação, compreensão e construção das histórias de usuários, garantindo que as mesmas sejam bem projetadas e implementadas visando e incentivando a colaboração entre os envolvidos no projeto, permitindo que eles expressem os requisitos de uma maneira mais testável para se obter um melhor entendimento dos requisitos [46]. O BDD orienta o desenvolvimento do sistema e oferece a oportunidade de realizar testes a partir de um conjunto de cenários que descrevem como a aplicação ou a unidade do código deverão se comportar em determinada situação[45]. A figura 2.2 ilustra cenários baseados em uma história de usuário.

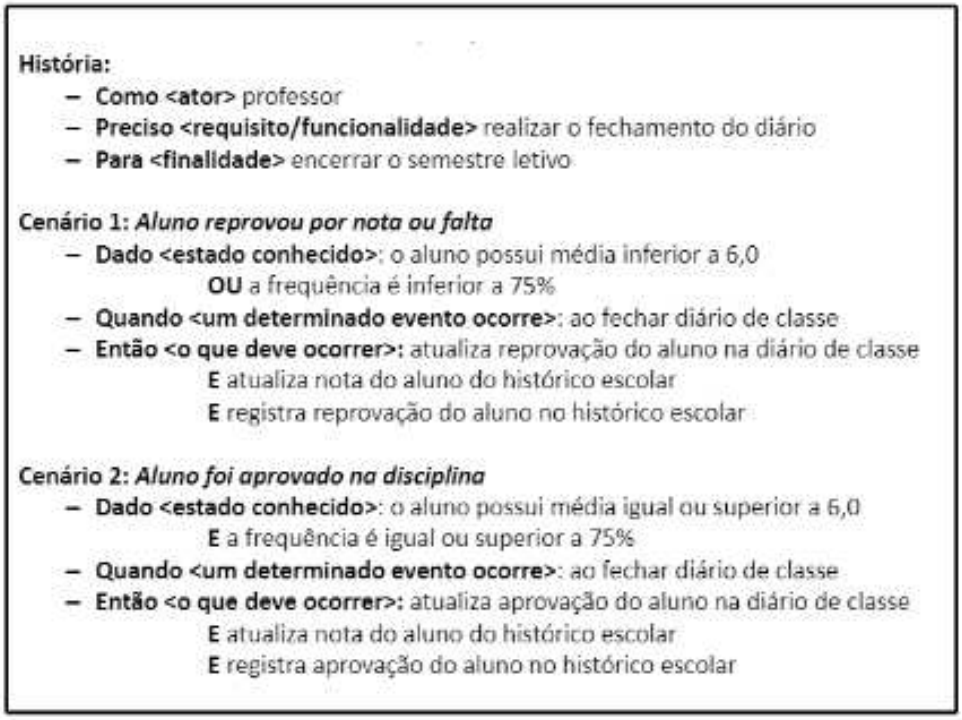

Figura 2.2 – Representação de cenários no BDD

Segundo North [34], o BDD evoluiu a partir de práticas ágeis estabelecidas e é projetado para torná-los mais acessíveis e eficazes para equipes novas para a entrega de software ágil. Ao

longo do tempo, BDD cresceu para abranger a imagem mais ampla de análise ágil e testes de aceitação.

2.3.1 Princípios do BDD

De acordo com Smart [46], o BDD segue alguns princípios, como a etapa de identificação dos objetivos do negócio, auxiliando na busca de recursos que ajudem a alcançar esses objetivos, onde usa exemplos concretos para tentar ilustrar o funcionamento do sistema. Na etapa de codificação o BDD ajuda os desenvolvedores a escrever um código de maior qualidade. Mostraremos como esses princípios funcionam de uma forma mais detalhada:

- **Foco nas características que oferecem valor ao negócio**

Em vez de tentar reunir todos os requisitos de uma só vez, as equipes que utilizam o BDD se envolvem em conversas contínuas com usuários finais e outros stakeholders para construir progressivamente a compreensão sobre quais funcionalidades eles devem desenvolver. Ao invés de aceitar uma lista de solicitações dos usuários, as equipes tentam entender os principais objetivos do projeto, propondo apenas recursos que podem ser demonstrados [46].

- **Trabalho em equipe para especificar as funcionalidades**

O BDD é uma prática altamente colaborativa, analistas de negócios, desenvolvedores e testadores trabalham em conjunto com os usuários finais para definir e especificar as funcionalidades do sistema, e os membros da equipe arquitetam ideias a partir de sua experiência individual. Esta abordagem é altamente eficiente para a interpretação dos requisitos [46].

- **Abraçar a incerteza**

Os profissionais que usam o BDD assumem que os requisitos ou mais precisamente a compreensão dos requisitos evoluirão e mudarão no decorrer do desenvolvimento do projeto. Eles tentam obter um parecer antecipado dos usuários e das partes interessadas para garantir que o desenvolvimento esteja no caminho certo e para que possam alterar o que for necessário durante o decorrer do desenvolvimento e não somente no final do projeto. Muitas vezes, a maneira mais eficaz de ver se os usuários gostam de um recurso é construí-lo e mostrá-lo o mais cedo possível [46].

- **Ilustrar as funcionalidades com exemplos concretos**

Quando uma funcionalidade é implementada, a equipe que pratica o BDD trabalha em conjunto com os usuários e outros stakeholders para definir histórias e cenários, na tentativa de entender o que será implementado. Os usuários ajudam a definir um conjunto de exemplos que ilustram os principais resultados esperados da funcionalidade. Esses exemplos são representados em um vocabulário comum e podem ser facilmente entendidos por todos os envolvidos no projeto [46].

- **Não escreva testes automatizados, escreva especificações executáveis**

Uma especificação executável é um teste automatizado que ilustra e verifica como a aplicação oferece um requisito de negócio específico. Esses testes automatizados são executados como parte do processo de compilação e são executados sempre que uma alteração é feita. Desta forma, eles servem tanto como testes de aceitação, determinando quais novos recursos são completos e, como testes de regressão, garantindo que novas alterações não quebraram quaisquer recursos existentes. Você pode automatizar uma especificação executável escrevendo o código de teste correspondente a cada etapa [46].

- **Não escreva testes de unidade, escreva especificações de baixo nível**

O BDD ajuda os desenvolvedores a escrever um código de qualidade, confiável, sustentável e melhor documentado. O desenvolvedor deve expressar o que o código deve realmente fazer em forma de uma especificação executável de baixo nível, descrevendo especificações técnicas de como uma determinada funcionalidade deve se comportar. Escrever especificações de baixo nível é escrever a documentação de forma detalhada [46].

- **Entregar documentação viva**

Os relatórios produzidos a partir de uma aplicação específica não são somente relatórios técnicos para desenvolvedores, mas também uma forma de documentação para toda a equipe, expressando um vocabulário familiar para os usuários. A documentação está sempre atualizada e requer pouca ou nenhuma manutenção manual.

Equipes experientes organizam esta documentação para que seja de fácil entendimento entre todos os stakeholders. Os desenvolvedores podem ver como funcionam as funcionalidades existentes, testadores e analistas podem ver como as funcionalidades foram implementadas e os usuários podem avaliar o estado atual do projeto [46].

- **Uso da documentação viva para apoiar o trabalho de manutenção**

Um projeto desenvolvido usando as práticas do BDD é mais fácil e menos caro de manter, pelo fato da documentação estar sempre atualizada. As especificações executáveis de alto nível ajudam os desenvolvedores a entender os objetivos de negócios e o fluxo da aplicação. As especificações executáveis no nível de teste de unidade fornecem exemplos detalhados de como as funcionalidades foram implementadas. Os desenvolvedores de manutenção que trabalham em um projeto BDD acham mais fácil saber por onde começar quando precisam fazer uma mudança. As boas especificações executáveis fornecem uma grande variedade de exemplos de como testar o sistema corretamente, e as mudanças de manutenção geralmente envolverão a escrita de uma nova especificação executável em linhas semelhantes ou a modificação de uma existente [46].

Poucos trabalhos apresentam como alternativa o uso do BDD para especificação e validação de requisitos. Neste trabalho o uso do BDD será uma das documentações a serem propostas para a utilização.

2.4 Modelo V

O modelo cascata é o modelo de desenvolvimento mais antigo da engenharia de software, nas últimas décadas este modelo sofreu muitas críticas fazendo com que sua eficácia fosse questionada até mesmo por seus fiéis defensores [35]. Este modelo apresenta um ciclo de vida sequencial e linear, onde cada etapa deve ser concluída antes que a próxima etapa possa ser realizada [49].

O modelo V é um modelo conceitual de desenvolvimento visto como uma melhoria ao modelo cascata, sugerindo uma abordagem sequencial e sistemática do início ao fim do desenvolvimento do software [35]. Esse modelo identifica que existem diferentes tipos de testes a serem realizados durante o desenvolvimento, como teste de unidade e teste de integração [41].

O modelo representa a relação entre os dois lados do processo de desenvolvimento, este relacionamento é usado para determinar se a fase foi concluída com sucesso. Neste modelo existe dependência entre as unidades, em cada unidade é necessário verificar e aprovar antes de avançar para a unidade seguinte, se ocorrer um problema durante a verificação ou validação de qualquer estágio, então o estágio oposto do "V"deve ser revisado e, se necessário, reiterado [5]. A figura a 2.3 representa o modelo V.

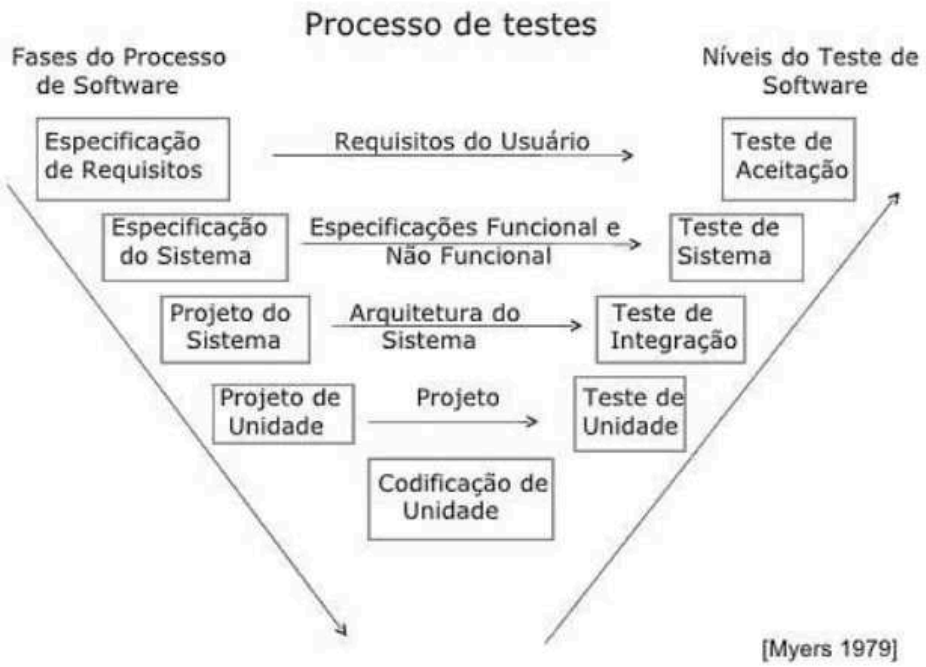

Figura 2.3 – Modelo V proposto por Myers [35]

No modelo V, as fases do lado esquerdo representam a fase de verificação, enquanto as fases de testes do lado direito representam a fase de validação. Esta ligação entre as etapas auxilia na identificação de erros, permitindo que sejam identificados logo nas etapas iniciais, evitando que os problemas sejam encontrados somente quando o sistema estiver todo codificado. O modelo V encapsula as atividades de verificação e validação , portanto, fornece uma estrutura útil para o desenvolvimento de software de alta qualidade [31].

Este modelo está sendo apresentado, pois será utilizado como base para a melhoria do processo a ser proposto.

2.5 Kanban

O método kanban foi criado na década de 1960 pela empresa Toyota, que tinha como objetivo manter um eficaz funcionamento do sistema de produção em série, o kanban foi inicialmente aplicado em empresas japonesas de fabricação em série e está estreitamente ligado ao conceito de "just in time". A utilização do sistema kanban permite um controle de produção sobre quando, quanto e o que produzir [2].

Kanban é um mecanismo de controle de fluxo, em que as atividades de processamento são desencadeadas pelos sinais de demanda do processo. Em um desenvolvimento de software

o kanban impulsiona as equipes de projetos a visualizar o fluxo de trabalho, limitar os trabalhos em andamento e medir o tempo de cada iteração, fornecendo visibilidade ao processo mostrando o trabalho atribuído e realizado por cada desenvolvedor [3].

O kanban geralmente é implementado com cartões de índice físico, os cartões agem como cartões de controle de fluxo entre as diferentes etapas de trabalho [24]. O quadro é a principal ferramenta utilizada para visualizar e coordenar o trabalho em equipe. As colunas representam a sequência de atividades, onde os cartões que representam as tarefas são colocados [14]. A figura 2.4 representa o quadro Kanban.

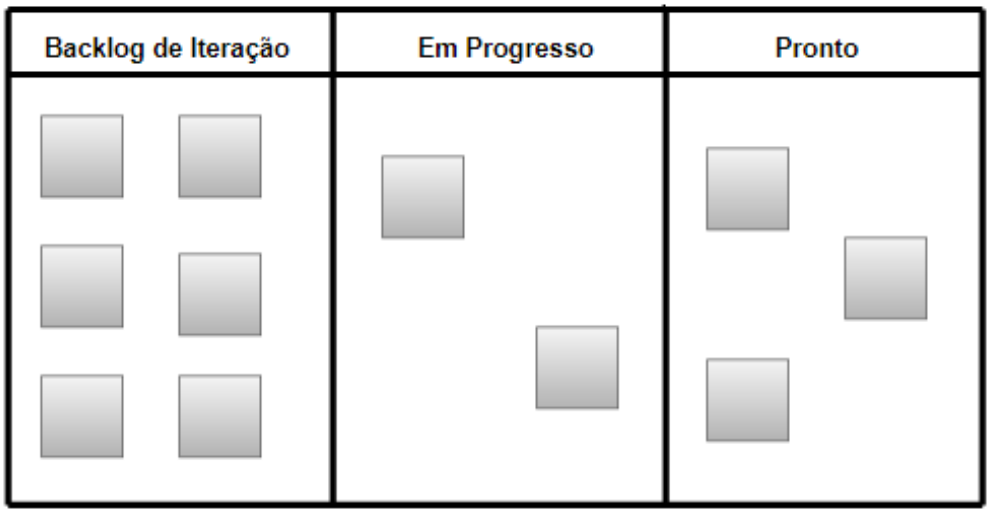

Figura 2.4 – Quadro Kanban (Adaptado de Prikladnick) [36]

2.6 Trabalhos Relacionados

Diogo G. Reali [39] apresentou através de um estudo de caso do desenvolvimento de uma aplicação web, um framework de avaliação baseado em mensuração, realizando uma investigação comparativa do desenvolvimento usando o Processo Unificado e o BDD (Behavior Driven Development) no processo de elicitação e validação de requisitos, analisando os resultados obtidos e ressaltando as vantagens e desvantagens da utilização de cada um dos métodos na tentativa de indicar qual processo exigia menor esforço dos desenvolvedores. Os resultados obtidos indicam que o processo usando o BDD necessita de um menor número de horas trabalhadas para a entrega de uma funcionalidade com complexidade semelhante.

Moraes [32] realizou um estudo empírico baseado em pesquisas semi-estruturadas que

buscou identificar como o BDD (Behavior Driven Development) é adotado na prática e como o mesmo apoia a engenharia de requisitos. Inicialmente foi realizado o levantamento de problemas em relação a engenharia de requisitos, após isso identificado como o BDD é utilizado na prática e como o BDD pode auxiliar na engenharia de requisitos na minimização desses problemas, também apresentou benefícios e dificuldades do uso da prática do BDD.

Cezerino [13] relata sobre o levantamento de requisitos utilizando o BDD (Behavior Driven Development), apontando os benefícios da técnica através de pesquisas realizadas por outros estudos. Destaque para o comparativo do levantamento de requisitos utilizando métodos tradicionais versus BDD e o estudo realizado por [39] que aponta que o uso do BDD requer menor esforço em relação ao Processo unificado. Através disso o artigo concluiu que embora seja uma técnica de desenvolvimento nova e que pouco se sabe sobre sua usabilidade o BDD se mostra eficiente em comparação a outras técnicas tradicionais já conhecidas.

Conforme observado os trabalhos relacionados focaram seus estudos no funcionamento do BDD, fazendo um comparativo com outras abordagens de desenvolvimento a fim de comprovar a sua eficácia. Com base no que foi constatado nos estudos relacionados, este trabalho pretende propor melhorias do processo de especificação e validação de requisitos nos métodos ágeis sugerindo uma documentação adicional através do BDD e outras documentações que possam auxiliar no processo.

3 METODOLOGIA

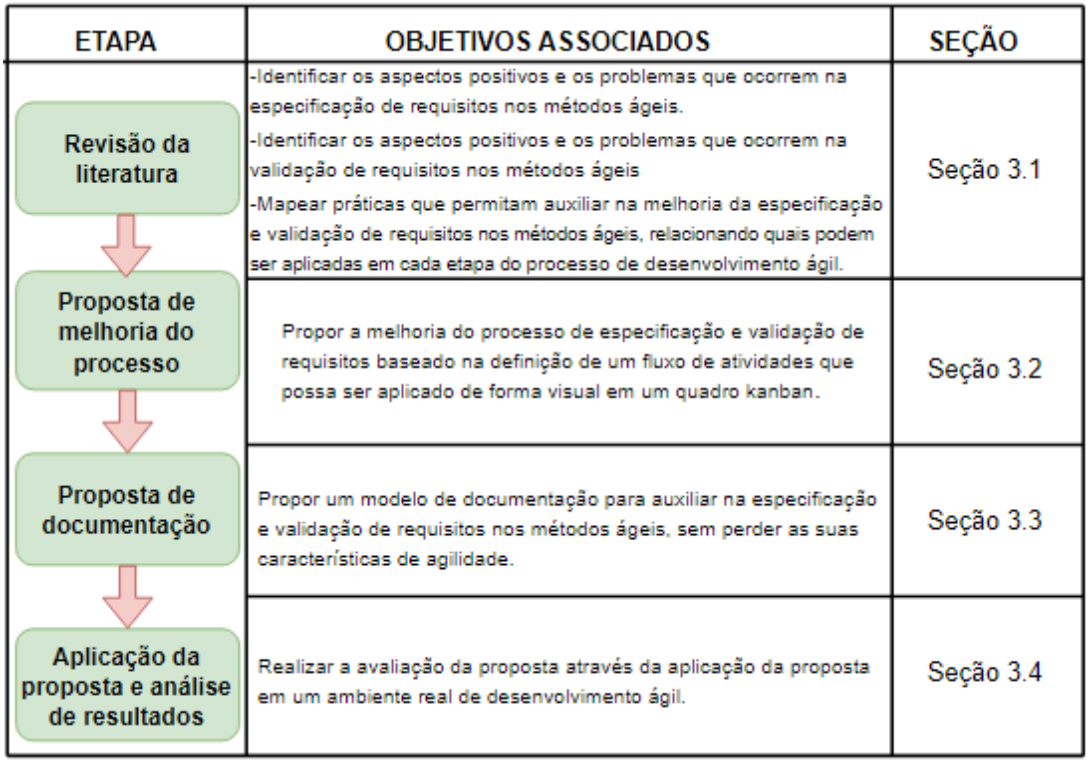

Figura 3.1 – Etapas de metodologia do trabalho

3.1 Revisão da literatura

Com o propósito de atingir o objetivo da pesquisa, primeiramente foi realizada a revisão de literatura por trabalhos que tratam sobre a engenharia de requisitos nos métodos ágeis, especificamente sobre as etapas de especificação e validação de requisitos. A revisão buscou identificar lacunas no processo de especificação e validação de requisitos, práticas utilizadas, os benefícios e os desafios enfrentados pelas equipes de desenvolvimento para posteriormente propor melhorias neste processo. Para realizar a pesquisa primeiramente foi definido palavras chaves para compor a string de busca a ser utilizada para pesquisa nas principais bases científicas. As bases pesquisadas foram: IEEE, ACM, Scopus, Springer Link. Os objetivos de pesquisa associados a esta fase de pesquisa são :

- Identificar os aspectos positivos e os problemas que ocorrem na especificação de requisitos nos métodos ágeis

- Identificar os aspectos positivos e os problemas que ocorrem na validação de requisitos nos métodos ágeis
- Mapear práticas que permitam auxiliar na melhoria do processo de especificação e validação de requisitos.

A representação das informações dos objetivos acima foram representados através de tabelas e descrição textual. Os resultados desta fase de pesquisa serviram como base para a próxima fase do trabalho. O resultado desta etapa é descrito no capítulo 4.

3.2 Proposta de melhoria do processo

Para a definição do processo foi utilizado como base o modelo V que propõe realizar testes em todas as fases do projeto, onde os testes tem maior efetividade evitando problemas futuros. O modelo representa a relação entre os dois lados do processo de desenvolvimento, este relacionamento é usado para determinar se a fase foi concluída com sucesso. Neste modelo os defeitos são detectados precocemente, os testes são realizados desde o início do projeto influenciando diretamente na qualidade do software.

A importância de adotar o modelo V se deve ao fato do mesmo associar atividades de testes em todas as fases do processo de desenvolvimento de software, que devem ser executados para garantir a entrega de um produto de qualidade.

Foi proposta a sequência de atividades necessárias para a melhoria do processo de desenvolvimento, cada uma das atividades estão detalhadas na seção 5.1. Posteriormente foi organizado essas atividades em um quadro Kanban para possibilitar a aplicação do processo na prática das empresas, onde foi definido o fluxo de atividades adequado ao processo onde possui colunas adicionais que representam a sequência de etapas do processo conforme é apresentado na seção 5.2.

Para a representação do processo foi utilizado a ferramenta BPMN Modeler, essa ferramenta possui uma breve descrição no capítulo 5. O processo proposto está representado graficamente no apêndice A deste trabalho.

3.3 Proposta de documentação

Através da revisão de literatura sobre engenharia de requisitos nos métodos ágeis, foi proposto um modelo de documentação para auxiliar na melhoria do processo de especificação

e validação de requisitos sem perder as características de agilidade, procurando manter uma documentação mínima, simples e eficaz para o entendimento dos requisitos e o desenvolvimento do projeto.

Na tentativa de melhorar a documentação em relação ao projeto a ser desenvolvido foi proposto modelos de documentações, dentre eles a documentação RN, que é uma documentação adicional as histórias de usuários que tem como objetivo complementar as informações contidas nas histórias de usuários com informações voltadas a regras de negócio utilizadas pelo cliente.

Outros modelos de documentações foram apresentados com objetivo propor uma maior visão sobre o produto a ser desenvolvido. Com o propósito de melhorar a visão de produto em relação a uma história de usuário foi utilizado o BDD. Também foi utilizado uma documentação referente aos RNF (Requisitos não funcionais), que visa tratar os RNF logo no início do projeto para diminuir os problemas relacionados.

As documentações criadas foram incorporadas sobre as histórias de usuários, criando assim um novo bloco de informações. As estruturas das documentações estão detalhadas no capítulo 5 deste trabalho.

3.4 Aplicação da proposta e análise de resultados

Na terceira e última etapa desta pesquisa foi realizada a avaliação do processo proposto, onde aplicou-se a proposta de melhoria em um ambiente simulado de desenvolvimento ágil. Após a aplicação foi realizado a coleta de dados assim como foi feita a análise dos resultados obtidos.

O ambiente simulado de desenvolvimento ágil utilizado para o estudo foi no componente curricular Planejamento e Gestão de Projetos do Curso de Ciência da Computação da Universidade Federal da Fronteira Sul (UFFS) que tem como objetivo ensinar os alunos a gerenciar projetos de software e desenvolverem um software no decorrer do semestre.

Neste ambiente existiam três equipes de desenvolvimento cada uma delas formada por quatro alunos. Cada equipe desenvolveu um projeto separado, porém todos foram orientados a seguir a mesma metodologia proposta neste trabalho, assim como o mesmo conjunto de práticas da metodologia Scrum. Devido a limitação de tempo para a realização deste trabalho e pelo fato dos times desenvolverem os projetos durante as aulas, a aplicação e a análise foram realizadas com base em duas iterações.

A primeira iteração teve 3 dias de desenvolvimento, a equipe seguiu a metodologia

scrum porém não utilizou a melhoria proposta neste trabalho. Já na iteração secundária que também foi de 3 dias a equipe seguiu o processo de melhoria proposto neste trabalho, assim como as documentações criadas para auxiliar no entendimento dos requisitos. No capítulo 5 foram detalhadas as duas iterações assim como apresentado um comparativo entre ambas.

4 RESULTADOS DA REVISÃO DA LITERATURA

Este capítulo apresenta os resultados obtidos através da revisão de literatura, realizada para atender os seguintes objetivos específicos:

- Identificar os aspectos positivos e os problemas que ocorrem na especificação de requisitos nos métodos ágeis.
- Identificar os aspectos positivos e os problemas que ocorrem na validação de requisitos nos métodos ágeis.
- Mapear práticas que permitam auxiliar na melhoria da especificação e validação de requisitos nos métodos ágeis, relacionando quais podem ser aplicadas em cada etapa do processo de desenvolvimento ágil.

4.1 Buscas por revisões de literatura relacionadas ao trabalho

4.1.1 Estratégia de busca

As buscas foram realizadas em artigos publicados entre os anos de 2005 a 2017, período após o início da utilização do BDD e a ocorrência do manifesto ágil que aconteceu em 2001. Foram realizadas buscas nas seguintes bases científicas: ACM Digital Library, IEEEXplore Digital Library, Science Direct e Springer Link. Para a busca foram definidas 3 strings de busca:

- **STRING 1 (busca por especificação e validação de requisitos nos métodos ágeis):**

("agile method"or "agile methodologies"or "scrum"or "agile software development"or "kanban software") and ("requirements engineering"or "specification requirements"or "validation requirements"or "practices engineering requirements"or "challenges engineering requirements")

- **STRING 2 (busca pelo uso do BDD nos métodos ágeis):**

("agile method"or "agile methodologies"or "scrum"or "agile software development"or "kanban software") and ("bdd"or "behavior driven development")

- **STRING 3 (busca pelo uso do modelo V nos métodos ágeis):**

("agile method"or "agile methodologies"or "scrum"or "agile software development"or "kanban software") and ("v-model"or "model v")

A busca utilizando a string1 retornou 21 artigos na base ACM, 54 artigos na base IEEE, 141 artigos na base Springer Link e 210 artigos na base Science Direct, totalizando 426 estudos. A string 2 retornou 2 artigos na base ACM, 6 artigos na IEE, 10 artigos no Science Direct e 6 artigos no Springer Link, totalizando 24 estudos. A string 3 retornou 4 artigos na base ACM. 4 artigos na IEEE, 26 artigos no Science Direct e 20 artigos no Sringer Link, totalizando 54 artigos. No total as 3 strings de busca retornaram 504 estudos.

4.1.2 Seleção dos Estudos

A seleção dos estudos foi dividida em três etapas. Na primeira etapa, foi realizada a leitura do título e do resumo do artigo, descartando estudos que não tratavam sobre especificação e validação de requisitos nos métodos ágeis, e estudos que não eram relacionados ao tema desta pesquisa. Na segunda etapa foi realizada a leitura da introdução e realizada uma análise geral do artigo. Na última etapa foi realizada uma análise detalhada dos artigos que passaram nas etapas anteriores. Os critérios de inclusão foram:

- O artigo menciona aspectos positivos que ocorrem na especificação e validação de requisitos nos métodos ágeis;
- O artigo menciona desafios enfrentados na especificação e validação de requisitos nos métodos ágeis;
- O artigo menciona práticas utilizadas para a especificação e validação de requisitos nos métodos ágeis;
- O artigo menciona o uso do BDD juntamente com métodos ágeis;
- O artigo menciona o uso do Modelo V juntamente com métodos ágeis;

Dos 504 artigos iniciais, 473 foram descartados na primeira etapa, resultando 31 artigos. Na segunda etapa dos 31 artigos foram descartados 12 artigos, resultando em 19 artigos. Finalmente após a última etapa de seleção foram descartados 6 artigos e resultando em um total de 13 artigos selecionados para este estudo.

Inicialmente foram descartados um grande número de artigos devido as strings serem gerais, para encontrar o maior número de artigos possíveis sobre o assunto. A tabela 4.1 apresenta os artigos selecionados a partir das strings de busca.

Tabela 4.1: Estudos selecionados pelas strings de busca

ID	Referência	Trabalhos Relacionados
E1	A Case Study on Benefits and Side-Effects of Agile Practices in Large-Scale Requirements Engineering	BJARNASON, Elizabeth; WNUK, Krzysztof; REGNELL, Björn. A case study on benefits and side-effects of agile practices in large-scale requirements engineering. In: Proceedings of the 1st Workshop on Agile Requirements Engineering. ACM, 2011. p. 3.
E2	An empirical study on the requirements engineering practices for agile software development	KASSAB, Mohamad. An empirical study on the requirements engineering practices for agile software development. In: Software Engineering and Advanced Applications (SEAA), 2014 40th EUROMICRO Conference on. IEEE, 2014. p. 254-261.
E3	A Multi-Case Study of Agile Requirements Engineering and the Use of Test Cases as Requirements	BJARNASON, Elizabeth et al. A multi-case study of agile requirements engineering and the use of test cases as requirements. Information and Software Technology, v. 77, p. 61-79, 2016.
E4	A Reflection on Agile Requirements Engineering: Solutions Brought And Challenges Posed	INAYAT, Irum et al. A reflection on agile requirements engineering: solutions brought and challenges posed. In: Scientific Workshop Proceedings of the XP2015. ACM, 2015. p. 6.
E5	Agile Requirements Engineering Practices: An Empirical Study	CAO, Lan; RAMESH, Balasubramaniam. Agile requirements engineering practices: An empirical study. IEEE software, v. 25, n. 1, 2008.
E6	Naming the pain in requirements engineering: A design for a global family of surveys and first results from Germany	FERNÁNDEZ, Daniel Méndez; WAGNER, Stefan. Naming the pain in requirements engineering: A design for a global family of surveys and first results from Germany. Information and Software Technology, v. 57, p. 616-643, 2015.
E7	Challenges and practices in aligning requirements with verification And validation: a case study of six companies	BJARNASON, Elizabeth et al. Challenges and practices in aligning requirements with verification and validation: a case study of six companies. Empirical Software Engineering, v. 19, n. 6, p. 1809-1855, 2014.
E8	A Mapping Study on Requirements Engineeringin Agile Software Development	HEIKKILÄ, Ville T. et al. A mapping study on requirements engineering in agile software development. In: Software Engineering and Advanced Applications (SEAA), 2015 41st Euromicro Conference on. IEEE, 2015. p. 199-207.
E9	A systematic literature review on agile requirements engineering practices and challenges	INAYAT, Irum et al. A systematic literature review on agile requirements engineering practices and challenges. Computers in human behavior, v. 51, p. 915-929, 2015.

Continua na próxima página

Tabela 4.1 Continuação da tabela 4.1

ID	Referência	Trabalhos Relacionados
E10	A systematic literature review on quality criteria for agile requirements specifications	HECK, Petra; ZAIDMAN, Andy. A systematic literature review on quality criteria for agile requirements specifications. Software Quality Journal, p. 1-34, 2016.
E11	Agile Requirements Engineering: A systematic literature review	SCHÖN, Eva-Maria; THOMASCHEWSKI, Jörg; ESCALONA, María José. Agile requirements engineering: a systematic literature review. Computer Standards and Interfaces, v. 49, p. 79-91, 2017.
E12	Review on Agile Requirements Engineering Challenges	ELGHARIANI, Kaiss; KAMA, Nazri. Review on Agile requirements engineering challenges. In: Computer and Information Sciences (ICCOINS), 2016 3rd International Conference on. IEEE, 2016. p. 507-512.
E13	State of practice in requirements engineering: contemporary data	KASSAB, Mohamad; NEILL, Colin; LAPLANTE, Phillip. State of practice in requirements engineering: contemporary data. Innovations in Systems and Software Engineering, v. 10, n. 4, p. 235-241, 2014.
E14	Engenharia de requisitos em projetos ágeis: Um mapeamento sistemático baseado em evidências da indústria	ALVES, Daniela de Castro Pereira. Engenharia de requisitos em projetos ágeis: um mapeamento sistemático baseado em evidências da indústria. 2015.

Após a seleção dos estudos foram identificados aspectos positivos e problemas relacionados a especificação e validação de requisitos, e as práticas utilizadas para especificar e validar requisitos. As tabelas 4.2 a 4.6 ilustram os resultados obtidos, assim como a frequência das mesmas. O cálculo da % de frequência foi realizado através da divisão entre o número de citações de um estudo pelo número total de artigos selecionados pela string de busca.

4.1.3 Aspectos positivos e problemas relacionados a especificação de requisitos

Procurando atender o objetivo específico: Identificar os aspectos positivos e os problemas que ocorrem na especificação de requisitos nos métodos ágeis, as tabelas 4.2 e 4.3 apresentam os resultados obtidos neste contexto.

Tabela 4.2: Aspectos positivos relacionados a especificação de requisitos

ID	Aspectos positivos	Estudos	Nº de citações	% Citação
1	Equipes multifuncionais aumentam a clareza dos requisitos	E1-E5	2	14.28%
2	A colaboração ativa do cliente permite resolver problemas no início do projeto, resultando em requisitos de qualidade superior	E1-E3-E5-E8-E9	5	35.71%
3	O detalhamento dos requisitos permite que o requisito seja mais estável e menos propenso a mudar resultando em uma especificação de requisitos mais confiável	E1-E10	2	14.28%
4	Foco nos requisitos mais importantes através da priorização	E1-E3-E5-E10	4	28.57%
5	Histórias de usuários aumentam a probabilidade de capturar e atender as expectativas dos clientes de forma curta, simples e direta.	E1-E4-E9-E12	4	28.57%
6	O alto nível da descrição dos requisitos proporciona a liberdade de projetar soluções inovadoras	E3	1	7.14%
7	As histórias de usuários grandes e complexas são divididas em menores, com base na facilidade de implementação	E4	1	7.14%
8	As histórias de entrega, que estendem histórias de usuários com especificações funcionais, cenários de teste e design de alto nível podem mitigar a insuficiência do formato da história do usuário	E8	1	7.14 %

A seguir será detalhado cada um dos aspectos positivos relacionados a especificação de requisitos encontrados na pesquisa:

Equipes Multifuncionais (ID 1): Equipes multifuncionais incluem membros de diferentes grupos funcionais que possuem objetivos semelhantes. No caso dos métodos ágeis desenvolvedores, testadores, designers e gerentes trabalham juntos e compartilham seu conhecimento. Esta prática ajuda a reduzir as falhas de comunicação e problemas com requisitos sobre o escopo, aumentando a clareza dos requisitos para uma melhor especificação [10].

Colaboração do cliente (ID 2): A participação ativa do cliente permite um consenso entre as partes interessadas e promove discussões em relação aos requisitos. Ao trabalhar em estreita colaboração com o cliente, as exigências podem ser resolvidas no início no projeto resultando em requisitos de qualidade superior e maior qualidade de software (menos erros), bem como, menos desperdício devido ao retrabalho [10]. O envolvimento ativo do cliente fortalece a consistência dos requisitos [9]. O cliente trabalha em conjunto com a equipe de desenvolvimento para alcançar a definição dos requisitos. Este processo é dinâmico e aberto a mudanças que podem ser realizadas a qualquer momento [26].

Detalhamento dos Requisitos (ID 3):

Os requisitos são primeiramente definidos em alto nível e então refinados iterativamente pela equipe de desenvolvimento, em requisitos mais detalhados a implementação progride. O detalhamento gradual dos requisitos auxilia no desafio de monitorar o desenvolvimento de uma perspectiva de requisitos e lacunas de comunicação no desenvolvimento, acarretando em um escopo mais viável. Finalizar e documentar os requisitos detalhados apenas quando são necessários para implementação significa que os requisitos são até então mais estáveis e menos propensos a mudar [10].

Foco nos requisitos através da priorização (ID 4): A priorização de requisitos permite aos desenvolvedores voltarem o seu foco para os requisitos mais importantes e de maior relevância em uma determinada etapa do projeto. Os requisitos primeiramente são definidos em alto nível e priorizados, em seguida refinados em requisitos mais detalhados onde podem ser priorizados novamente [10]. A equipe de desenvolvimento seleciona itens de trabalho com base na prioridade, custo e viabilidade [9]. Em cada iteração, os requisitos com a maior prioridade são detalhados e posteriormente desenvolvidos. A prioridade pode

mudar enquanto o requisito estiver aberto. Desta forma, o desenvolvimento ágil garante que o cliente recebe o que mais precisa em qualquer momento. Isso também permite que o cliente possa mudar de ideia ou alterar algum requisito durante o projeto e alterar suas prioridades [22].

Uso de histórias de usuários (ID 5): As histórias de usuários especificam os requisitos do cliente de forma breve e concisa, são curtas, diretas e compreensíveis pelas partes interessadas [25]. Conforme Bjarnason [10] as histórias de usuários facilitam a comunicação entre as funções de negócios e engenharia (ao expressar o ponto de vista dos usuários) e aumentam a probabilidade de capturar e atender às expectativas dos clientes. As histórias de usuários deslocam a concentração da documentação escrita para a comunicação [18].

Alto nível de descrição dos requisitos (ID 6):

A participação do cliente é procurada principalmente para funcionalidades complexas, através de entrevistas e comunicação direta entre stakeholders e engenheiros. O alto nível de requisitos descritos pelo cliente fornece aos engenheiros de desenvolvimento liberdade para projetar soluções inovadoras que permite produzir produtos muito criativos [9].

Divisão de histórias de usuários grandes e complexas (ID 7):

As histórias de usuário especificam os requisitos do cliente de forma breve e concisa. As histórias de usuários não são detalhadas como as especificações de requisitos tradicionais. Eles são curtos, diretos e compreensíveis pelas partes interessadas. As histórias de usuários grandes e complexas são divididas em menores, com base na facilidade de implementação.

Uso das histórias de entrega (ID 8):

As histórias de entrega que são histórias de usuários estendidas com especificações funcionais, cenários de design e teste de alto nível podem mitigar a insuficiência do formato da história do usuário em grandes redes de provedores de clientes. As histórias de entrega melhoram a compreensão dos requisitos e ajudam na concepção do sistema.

Conforme apresentado na tabela 4.2 foram encontrados 8 aspectos positivos relacionados a especificação de requisitos nos métodos ágeis. Destaque para a colaboração ativa do cliente (ID 2) que teve o maior índice de citações com 35.71 %, outros aspectos positivos como

o benefício do uso das histórias de usuários (ID 5) e foco nos requisitos através da priorização de requisitos (ID 4) também tiveram um alto índice de citações.

Tabela 4.3: Problemas relacionados a especificação de requisitos

ID	Problemas	Estudos	Nº de citações	% Citações
1	Informações insuficientes contidas nas histórias de usuários podem levar a uma especificação de requisitos incorreta	E1-E4-E5-E6-E8-E9-E12-E14	8	57.14 %
2	Mudanças nos requisitos devido a má especificação no início do projeto	E5-E6-E7-E12	4	28.57%
3	Muito tempo e esforço são gastos para discutir e identificar requisitos reais ou completos	E3-E7-E12	3	21.43%
4	Problemas pela falta de conhecimentos técnicos do cliente	E3-E5-E6	3	21.43%
5	Na falta de especificação de um requisito, os testadores precisam supor e compilar as informações faltantes	E5-E7	2	14.29%
6	Insuficiência dos requisitos para escrever o software e seus testes	E6-E7	2	14.29%
7	Os clientes podem não estar dispostos ou são incapazes de priorizar diretamente os requisitos	E1-E5-E8	3	21.43%

A tabela 4.3 ilustra os problemas relacionados a especificação de requisitos. A insuficiencia de informações contidas nas histórias de usuários (ID 1) lidera a lista de problemas com 57.14 %, requisitos incompletos ou mau especificados (ID 2), tempo e esforço gasto para identificar requisitos completos (ID 3) e indisposição ou incapacidade do cliente em priorizar requisitos (ID 7) foram outros problemas mencionados. Alguns aspectos positivos da especificação de requisitos em métodos ágeis solucionam problemas da engenharia tradicional, mas consequentemente ocasionam novos problemas. Esses problemas serão detalhados a seguir.

Informações insuficientes contidas nas histórias de usuário (ID 1):

Caso houver falta de comunicação em determinado momento entre os membros da equipe, a documentação é necessária para manter o acompanhamento de quaisquer mudanças [19]. Os métodos ágeis substituem a documentação convencional por uma documentação mínima [11], que possui um objetivo de usuário específico descrito nas histórias de usuários. Quando o software a ser desenvolvido é grande ou complexo as histórias de usuário não transmitem as informações suficientes para o projeto de software [15].

Segundo Haugset [21], a falta de documentação pode trazer riscos para a mudança de código existente. Quando ocorre um problema de comunicação a falta de documentação pode causar uma variedade de problemas. Estes incluem a incapacidade de dimensionar o software, evoluir o sistema ao longo do tempo e incorporar novos membros na equipe de desenvolvimento [38]. O artigo [26] relata que essa documentação mínima é um desafio vital que os métodos ágeis representam para as equipes de desenvolvimento. O conteúdo contido nas histórias de usuários pode ser escasso para uma boa especificação de requisitos.

Mudanças nos requisitos devido a má especificação no início do projeto (ID 2):

Mudanças nos requisitos são comuns em projetos que usam métodos ágeis. De acordo com Baskerville et al. [6], as necessidades de alteração que surgem no desenvolvimento não acontecem somente pela falta de conhecimento ou experiência dos usuários, mas também pela evolução constante da tecnologia. Bjarnason et al. [10], relatou que os clientes querem realizar mudanças a todo instante não compreendendo a disciplina de envolvimento do cliente.

Alguns dos entrevistados nos estudos de [10] relataram que a falta de uma definição clara de requisitos no início do desenvolvimento do projeto resultou em um grande número de mudanças durante o decorrer do processo, ocasionando retrabalho e causando frustração dentro da equipe de desenvolvimento.

Tempo e esforço gasto para identificar requisitos completos (ID 3):

Os requisitos são inicialmente especificados em alto nível , durante o processo esses requisitos são detalhados, porém identificar o requisito completo não é uma tarefa fácil. O artigo [9] relata que é necessário gastar muito tempo e esforço para discutir e identificar os requisitos reais. Quando os testadores são obrigados a trabalhar com requisitos incompletos,

informações adicionais são adquiridas de outras fontes, o que requer tempo e esforço adicionais para localizar essas informações. A indisponibilidade e a falta de conhecimentos técnicos do cliente também dificultam a identificação de um requisito completo.

Falta de conhecimento do cliente (ID 4):

O conhecimento técnico limitado afeta a capacidade do cliente para discutir os requisitos de qualidade. Isso pode levar à negligência para especificá-los completamente [38]. As partes interessadas no projeto muitas vezes não possuem um entendimento claro do que o sistema deve fazer ou como o sistema deve executar, ocasionando mudanças futuras [44].

Suposição de requisitos por parte dos testadores (ID 5):

Quando houver falta a especificação dos requisitos, os testadores precisam supor os requisitos com base na sua experiência e compilar as informações faltantes, já que os requisitos não são suficientes para escrever software e teste do software [8].

Insuficiência dos requisitos para escrever o software e seus testes (ID 6):

Quando os requisitos não são suficientes para escrever o software e testes do software, aumenta-se o esforço necessário para testar e o risco de má interpretação dos requisitos do cliente [8].

Incapacidade ou indisposição do cliente em priorizar requisitos (ID 7):

A priorização dos requisitos é baseada no valor comercial, isso consequentemente pode acarretar que a arquitetura do sistema ou requisitos relacionados ao sistema sejam ignorados [10]. Quando uma nova priorização não é praticada com prudência, pode ocasionar instabilidade no projeto [11].

Realizar uma entrega rápida de requisitos simples que são visíveis pelo cliente pode criar expectativas irrealistas [11]. Clientes podem ter necessidades conflitantes e podem ter dificuldades para chegar em uma consenso sobre prioridades [29], assim como podem não estar dispostos ou são incapazes de piorizar requisitos [37].

4.1.4 Aspectos positivos e problemas na validação de requisitos

Procurando atender o objetivo específico: Identificar os aspectos positivos e os problemas que ocorrem na validação de requisitos nos métodos ágeis, as tabelas 4.4 e 4.5 apresentam os resultados obtidos neste contexto.

Tabela 4.4: Aspectos positivos relacionados a validação de requisitos

ID	Aspectos positivos	Estudos	Nº de citações	% Citações
1	A grande quantidade de testes logo no começo do desenvolvimento permite evitar problemas maiores	E3-E5-E7-E14	4	28.57%
2	Requisitos completos são validados logo pela proximidade de interação entre as etapas.	E7-E8	2	14.29%
3	A priorização de requisitos feita pelo cliente e a prototipagem contribuem para a validação de requisitos	E4	1	7.14%
4	Os testes de aceitação ajudam a corrigir as histórias ,aumentam a usabilidade, melhoram a funcionalidade e determinam a integridade da implementação das histórias	E4-E5	2	14.29%
5	Práticas para validar o sistema em desenvolvimento garantem atender as expectativas do cliente	E7	1	7.14%
6	A rápida validação de protótipos pelos clientes simplifica o gerenciamento de mudanças posteriores	E9-E14	2	14.29%
7	A iteração com o cliente cria um relacionamento mais satisfatório e aumento da confiança entre cliente e equipe	E5-E8-E14	3	21.43%

A tabela 4.4 ilustra os aspectos positivos relacionados a validação de requisitos. A grande quantidade de testes realizados no começo do desenvolvimento (ID 1) e a iteração com

o cliente (ID 7) foram os aspectos que receberam maior destaque.

Grande quantidade de testes no início do desenvolvimento (ID 1): A realização de testes desde o início do desenvolvimento permite identificar problemas logo nas fases iniciais do projeto e não somente no final, assim facilitando a correção desses problemas que poderiam ocasionar problemas maiores futuramente. Um teste unitário visa avaliar pequenas unidades que compõem um software proporcionando assim uma maior probabilidade de sucesso nos testes de aceitação.

Rápida validação de requisitos completos (ID 2): Nos métodos ágeis os requisitos são validados rapidamente, assim que um requisito é codificado o mesmo é testado e validado caso estiver correto, ou corrigido caso estiver incompleto. O artigo [30] apresenta um modelo concentrado na validação precoce para aliviar vários problemas associados ao desenvolvimento de software e minimizando o custo do produto. As abordagens ágeis permitem uma entrega mais rápida de valor e validação do sistema [23].

A priorização de requisitos feita pelo cliente (ID 3): A validação de requisitos é outro problema dos métodos tradicionais que é resolvido através da priorização de requisitos constantes feito pelo cliente usando várias técnicas. Além disso, a prototipagem também ajuda o cliente a visualizar suas demandas e sugira mudanças se necessário em estados iniciais de desenvolvimento [25].

Testes de aceitação ajudam a corrigir as histórias de usuários melhorando a usabilidade e funcionalidades (ID 4):

No final de cada ciclo, os testes avaliam os recursos implementados. Os testes de aceitação que o cliente desenvolve são meios para validação de requisitos [11]. Os testes são criados e aplicados para cada uma das histórias de usuários definidas. Esses testes confirmam a correção das histórias dos usuários aumentando a usabilidade e a funcionalidade da história do usuário [25].

Práticas para validar o sistema garantem atender as expectativas do cliente (ID 5):

Um bom alinhamento de práticas de validação é essencial para permitir um desenvolvi-

mento de software eficiente, contribuindo na produção de software que atende às necessidades e expectativas dos clientes. Os testes de requisitos são um aspecto importante para garantir que o produto final atenda aos requisitos e às expectativas dos clientes[8].

A rápida validação de protótipos simplifica o gerenciamento de mudanças (ID 6):

Os protótipos permitem que os usuários e as partes interessadas possam trabalhar com a versão inicial do produto objetivando entender o sistema e discutir requisitos adicionais e/ou requisitos não atendidos [4]. É uma maneira simples e direta de revisar as especificações de requisitos com os clientes e obter feedback oportuno antes de mover para iterações subsequentes, promovendo feedback mais rápido e melhora a antecipação do cliente do produto [26].

Iteração com o cliente cria um relacionamento mais satisfatório (ID 7):

A iteração com o cliente permite um relacionamento mais satisfatório com o cliente. Além disso, foi percebido que a utilização de reuniões validação permitem a identificação mais rápidas de problemas no processo de desenvolvimento e também possibilita a verificação se o projeto está no alvo, o que aumenta a confiança do cliente e confiança na equipe [4].

Tabela 4.5: Problemas relacionados a validação de requisitos

ID	Problemas	Estudos	Nº de citações	% Citações
1	Representante do cliente nem sempre está disponível	E1-E3-E4-E5-E6-E8-E9-E12-E14	9	64.28%
2	A definição de casos de testes para validação / Casos de testes mau estruturados	E3-E5-E7	3	21.43%
3	Dificuldade em entender o que o cliente realmente precisa	E3-E5-E6	3	21.43%
4	Motivar os engenheiros a escrever casos de teste de nível de aceitação não é fácil.	E3	1	7.14%
5	Capturar requisitos complexos com casos de teste de aceitação é um desafio.	E3	1	7.14%
6	Uma má priorização de requisitos pode levar a um atraso do projeto, elevação de custo e retrabalho.	E4-E5-E14	3	21.43%

Na tabela 4.5 é apresentado os problemas relacionados a validação de requisitos. A indisponibilidade do representante do cliente (ID 1) foi o problema mais citado, definição de casos de teste para validação(ID 2), e má priorização de requisitos (ID 6) também receberam destaque. Os principais problemas serão detalhados abaixo.

Representante do cliente nem sempre está disponível (ID 1):

A participação ativa do cliente é incentivada pelas abordagens ágeis, porém a disponibilidade local do cliente é um problema frequente no desenvolvimento de software [38]. Nos estudos de [10] foi mencionado que o cliente ou seu representante nem sempre está suficientemente disponível e está envolvido na equipe de desenvolvimento. Grandes projetos dependem de representantes do cliente, que devem dividir sua atenção entre o cliente e os desenvolvedores do produto [29].

Embora nenhum estudo tenha afirmado que as mudanças nos requisitos podem ser determinadas diretamente pelo cliente para o andamento do projeto, a disponibilidade do cliente é considerada desafiadora e necessária, pois pode influenciar diretamente na perspectiva do negócio [37]. Os estudos mostram que quase todas as pesquisas realizadas relataram problemas referentes a participação do cliente e apontaram a necessidade de haver uma representação frequente para um bom andamento do projeto. Por esse motivo muitas empresas acabam optando por utilizar um desenvolvedor como representante do cliente, que muitas vezes podem não conhecer o mercado [33].

Definição de casos de testes (ID 2): Definir o que testar para validar um requisito não é uma tarefa fácil, é necessário ter o total domínio sobre o o requisito, saber o que deve se testado e porque testar. Todos os entrevistados de [9] mencionaram o desafio de motivar os engenheiros a escrever casos de teste e que o cliente necessita de qualidades técnicas para escrever um bom teste e saber o que testar.

Dificuldade em entender o que o cliente precisa (ID 3):

Os clientes expressam os requisitos a um nível de abstração mais alto ao invés de se concentrar em detalhes técnicos[9]. A eficácia no entendimento do requisito depende fortemente da interação intensiva entre clientes e desenvolvedores. Para projetos que não conseguem al-

cançar essa interação de alta qualidade, essa abordagem coloca riscos como requisitos que estão inadequadamente desenvolvidos ou, pior ainda, errados [11].

Segundo os estudos realizados por Rudorfer [42], as relações entre o problema e a solução não são transparentes. As partes interessadas no projeto muitas vezes não possuem um entendimento claro do que o sistema deve fazer ou como o sistema deve executar, então as partes interessadas frequentemente mudam de ideia sobre as funcionalidades [44].

Dificuldade em escrever casos de testes (ID 4):

O cliente que não possui conhecimentos técnicos não fornece requisitos de qualidade. Os engenheiros acham difícil combinar os requisitos de alto nível com o código necessário para os casos de teste, motivar os engenheiros a escrever casos de teste de nível de aceitação é considerado um desafio [9].

Capturar requisitos complexos com casos de testes de aceitação (ID 5):

O conhecimento técnico limitado afeta a capacidade do cliente para discutir os requisitos de qualidade. Isso pode levar à negligência para construir os requisitos corretamente. Da mesma forma, capturar requisitos complexos com casos de teste de aceitação é um desafio [9].

Problemas na priorização de requisitos (ID 6): Requisitos são priorizados pela sua importância e por seu valor comercial, uma má priorização pode acarretar problemas no projeto. Estudos encontrados em [4] relatam que uma nova priorização quando não praticada com cautela leva a instabilidade. A constante redefinição de prioridades dos requisitos é considerada um desafio. Uma má priorização de requisitos consequentemente ocasionará em problemas na validação de um requisito, onde seus testes de validação podem ter sido realizados no momento errado.

4.1.5 Práticas utilizadas para especificação e validação de requisitos

Procurando atender o objetivo específico: Mapear práticas que permitam auxiliar na melhoria da especificação e validação de requisitos nos métodos ágeis, relacionando quais podem ser aplicadas em cada etapa do processo de desenvolvimento ágil, a tabela 4.6 apresenta as práticas identificadas neste contexto.

Tabela 4.6: Práticas utilizadas para especificação e validação de requisitos

ID	Nome da prática	Descrição	Estudos	% Citações
1	Equipes de desenvolvimento multifuncional	Equipes trabalham em conjunto nas tarefas de especificação e validação de requisitos.	E1-E4-E7-E9-E12	35.71%
2	Refatoração de código	Modificar um sistema de software para melhorar a estrutura interna do código sem alterar seu comportamento.	E8-E11	14.29%
3	Workshop	Reunião de grupos de pessoas interessados em determinado projeto ou atividade.	E14	7.14%
4	Requisitos iterativos	Requisitos que são alterados/evoluem com as etapas do desenvolvimento.	E1-E4-E5-E9-E12	35.71%
5	Histórias de usuários	Histórias curtas, simples e diretas para especificar os requisitos do usuário.	E1-E2-E4-E9-E11-E12-E13-E14	57.14%
6	Priorização de requisitos	Requisitos são priorizados conforme sua importância.	E4-E5-E9-E10-E12	35.71%
7	Reunião de revisão da sprint	Revisão da sprint realizada onde o PO aprova o que foi produzido pela equipe de desenvolvedores.	E4-E5-E9-E12	28.57%
8	Sprint retrospectiva	Retrospectiva da sprint realizada com o objetivo de avaliar o que foi realizado na sprint.	E4-E9-E12	21.43%
9	Participação ativa do cliente/ Comunicação cara-cara	Cliente participa a todo momento do projeto através de reuniões presenciais.	E2-E4-E5-E7-E9-E12-E13	50.00%
10	Testar antes de codificar/-Testes unitários	Teste de cada funcionalidade é criado antes da codificação normalmente utilizando o TDD.	E4-E5-E9-E12	28.57%
11	Prototipagem	Desenvolvimento de um protótipo com base no conhecimento dos requisitos iniciais.	E2-E4-E5-E7-E9-E11-E12-E13-E14	64.29%
12	Testes de aceitação	Teste para avaliar a qualidade do produto.	E4-E7-E10	21.43%

Continua na próxima página

Tabela 4.6 Continuação da tabela 4.6

ID	Nome da prática	Descrição	Estudos	% Citações
13	Brainstorming	Técnica de dinâmica de grupo desenvolvida para mapear a potencialidade criativa de uma determinada equipe.	E2-E13-E14	21.43%
14	Questionários	Técnica para coletar informações de composta por um número grande ou pequeno de questões.	E14	7.14%
15	Gestão de mudanças	Priorizar e avaliar as mudanças solicitadas por clientes nos requisitos.	E4-E9-E12	21.43%
16	Gerenciamento de requisitos	Encontrar, documentar, organizar e rastrear os requisitos variáveis de um sistema.	E4-E5-E9-E12	28.57%
17	BDD	Técnica de desenvolvimento orientada a comportamento que oferece a oportunidade de realizar testes a partir de um conjunto de cenários.	E2-E11-E13-E14	28.57%
18	Grupo Focal	Forma de entrevistas que coleta informações por meio das interações grupais.	E2-E13-E14	21.43%

- **Equipes multifuncionais (ID 1):**

Equipes multifuncionais incluem membros de diferentes grupos funcionais que possuem objetivos semelhantes. No caso dos métodos ágeis desenvolvedores, testadores, designers e gerentes trabalham juntos e compartilham seu conhecimento. Esta prática ajuda a reduzir as falhas de comunicação e problemas com requisitos sobre o escopo [10].

- **Refatoração de código (ID 2):**

A refatoração de código é a prática fundamental para manter o código limpo [29]. A prática destina-se a acomodar mudanças nos requisitos e atender a instabilidade de requisitos

em iterações posteriores [7].

- **Workshop (ID 3):**

Workshop é uma reunião em grupo, onde devem fazer parte do grupo a equipe de analistas e os stakeholders. A reunião deve ser conduzida por uma pessoa neutra e observadora que deve evitar de influenciar na tomada de decisões das pessoas envolvidas na reunião [4].

- **Requisitos Iterativos (ID 4):**

Nos métodos ágeis, os requisitos aparecem também durante o decorrer do desenvolvimento do projeto [38]. A interação frequente entre as partes interessadas no projeto, levam a essa abordagem de exigências iterativas, que torna os requisitos mais claros durante o processo e permite que os requisitos evoluam com menos investimento de tempo [11].

- **Histórias de Usuário (ID 5):**

Histórias de usuário são uma forma simples de especificar as necessidades do cliente. As histórias descritas pelo cliente facilitam a comunicação e proporcionam uma maior compreensão entre os envolvidos no projeto [15]. As histórias de usuários são conceituadas para serem capazes de solucionar o problema da constante atualização de documentos de especificação de requisitos que acontece na engenharia de requisitos tradicional [10].

- **Priorização de requisitos (ID 6):**

Diferente da metodologia tradicional onde os requisitos são priorizados no início do projeto, em um desenvolvimento ágil os requisitos são priorizados constantemente em cada iteração do desenvolvimento [11]. Segundo estudos realizados por Daneva et al. [15], quem define as prioridades dos requisitos geralmente é o cliente.

- **Reunião de revisão da sprint (ID 7):**

Reuniões de revisão são realizadas frequentemente para validação de requisitos. No final de cada ciclo de desenvolvimento, é agendada uma reunião que envolve desenvolvedores, clientes e outras partes interessadas. Durante a reunião, os recursos entregues são demonstrados, os

clientes e controle de qualidade fazem perguntas e fornecem feedback e comentários. As reuniões de revisão mostram que o projeto está dentro do prazo e dentro do cronograma, aumenta a confiança e a confiança do cliente na equipe e destaca os problemas no início [11].

- **Retrospectivas (ID 8):**

Retrospectivas são reuniões realizadas após o término de uma iteração [12]. Nessas reuniões é visto o trabalho realizado até o momento e é determinada as etapas futuras de retrabalho caso preciso [26].

- **Comunicação cara a cara (ID 9):**

A comunicação cara a cara é uma característica relevante da engenharia de requisitos nos métodos ágeis, proporciona uma comunicação ambiental entre os representantes do cliente e os membros da equipe de desenvolvedores. Os métodos ágeis defendem a documentação mínima [11] como histórias de usuário. A comunicação cara a cara frequente ajuda o cliente a dirigir o projeto em uma direção de acordo com a sua própria compreensão do projeto, uma comunicação repentina entre as partes interessadas no projeto ajuda na evolução dos requisitos. A frequência de comunicação depende principalmente da disponibilidade e disposição dos membros da equipe [26].

- **Testar antes de codificar (ID 10):**

Essa técnica propõe escrever testes antes de começar codificar os requisitos funcionais, na tentativa de identificar falhas antes da fase da codificação [26].

- **Prototipagem (ID 11):**

Um protótipo é uma versão inicial do sistema, baseado nos requisitos que ainda não estão totalmente definidos, representa o produto real tanto no sentido funcional como no sentido gráfico [47]. A prototipagem começa com requisitos simples que são completamente compreendidos e possuem uma prioridade alta [16]. A prototipagem esclarece o entendimento do problema e pode identificar falhas iniciais no projeto.

- **Revisão de reuniões e testes de aceitação (ID 12):**

Nesta prática é apresentado os requisitos desenvolvidos e o atraso de produtos que são revistos constantemente nas reuniões [12]. É uma forma de saber quais histórias foram concluidas e quais ainda estão por fazer, assim como quais foram "aprovadas"e "rejeitadas"[23].

- **Brainstorm (ID 13):**

É uma técnica usada para explorar a criatividade dos indivíduos em busca de soluções para questões específicas. O Brainstorm pode ser dividido em duas fases, a primeira propõe que cada indivíduo exponha suas ideias, a segunda é onde essas ideias são discutidas. O objetivo da técnica é que seja levantado o maior número de ideias possíveis e levar os indivíduos a um denominador comum. [4].

- **Questionários (ID 14):**

Questionários são muito utilizados quando é necessário coletar as mesmas informações de vários usuários ao mesmo tempo. Cada usuário responde o questionário individualmente, através da resposta dos entrevistados é que os requisitos são identificados. É importante que as questões do questionário sejam simples e de fácil entendimento para os usuários [4].

- **Gestão de mudanças (ID 15):**

As abordagens ágeis possuem maior facilidade em lidar com mudanças em relação a abordagens tradicionais. As principais mudanças relatadas são adicionar ou retirar recursos do projeto [11]. Com uma comunicação frequente entre cliente e desenvolvedor diminuem as necessidades de mudança nas etapas seguintes [18].

- **Gerenciamento de Requisitos (ID 16):**

O conjunto de atividades relacionadas ao processo de acompanhamento de mudanças é denominada gerência de requisitos [43]. O gerenciamento de requisitos é realizado a partir do backlog do produto, lista de recursos e cartões de índice [11]. No Scrum o backlog do produto pode ser usado para acompanhar as alterações de requisitos [23].

- **BDD (ID 17):**

O BDD é uma técnica de desenvolvimento orientada a comportamento que surgiu na tentativa de auxiliar as equipes de desenvolvimento a construir e entregar um software de qualidade e de forma rápida, descrevendo como o sistema deve se comportar com uma linguagem natural. O BDD orienta o desenvolvimento do sistema e oferece a oportunidade de realizar testes a partir de um conjunto de cenários.

Cenário é uma forma de tentar demonstrar ao usuário como será o comportamento do sistema, através de exemplos práticos e descritivos os usuários podem demonstrar o que esperam do sistema, como querem que o sistema funcione e interrogar o funcionamento de determinada funcionalidade. O objetivo dessa prática é proporcionar uma melhor compreensão do sistema para os stakeholders [4].

- **Grupo Focal (ID 18):**

É uma técnica onde um grupo de usuários discutem questões livremente, propiciando um ambiente mais natural para o diálogo pois todos os participantes expõem sua opinião e ouvem a opinião dos demais. Nesta discussão são identificadas necessidades do usuário e assuntos relacionados a um tópico específico [4].

A tabela 4.6 ilustra as práticas utilizadas para especificação e validação de requisitos nos métodos ágeis. Conforme o levantamento um total de 18 práticas estão sendo utilizadas. As práticas mais citadas foram prototipagem, histórias de usuários e participação ativa do cliente com 64.29 %, 57.14% e 50.00% de ocorrências respectivamente. Outras práticas mencionadas tiveram um número considerado de ocorrências, isso mostra o uso comum de práticas em diferentes estudos.

Como podemos observar existem muitos estudos relacionados ao desenvolvimento de software ágil e que existe também um grande esforço para a melhoria da qualidade do desenvolvimento de software através de práticas de especificação e validação de requisitos, porém muitos problemas continuam sendo relatados, ou seja mesmo com a melhora de qualidade e com o uso de práticas ágeis, vários problemas ainda persistem e outros novos surgem, não conseguindo assim minimizar as dificuldades enfrentadas em um desenvolvimento de software ágil.

Muitos estudos relatam um esforço por parte da validação dos requisitos, o TDD (Test Driven Development) é uma técnica de desenvolvimento orientada a testes, baseada em pequenos ciclos de desenvolvimento e práticas de escrita de testes realizados antes da codificação, na tentativa de descobrir problemas nas fases iniciais do projeto, mas conforme North [34] os desenvolvedores encontravam muitas dificuldades em relação ao detalhamento dos testes a serem realizados, então North propôs o BDD (Behavior Driven Development) que orienta o desenvolvimento do sistema e oferece a oportunidade de realizar testes a partir de um conjunto de cenários. Este trabalho irá apresentar soluções viáveis utilizando o BDD na tentativa de minimizar os problemas enfrentados na especificação e validação de requisitos nos métodos ágeis

5 MAPEAMENTO DO PROCESSO

Neste capítulo será respondido a seguinte questão de pesquisa:

QP. Como melhorar o processo de especificação e validação de requisitos nos métodos ágeis?

Os objetivos específicos relacionados a questão de pesquisa e atendidos neste capítulo são:

- Propor a melhoria do processo de especificação e validação de requisitos baseando-se no modelo V, definindo um fluxo de atividades que possa ser aplicado de forma visual em um quadro kanban.
- Propor documentações que possam auxiliar na especificação e validação de requisitos nos métodos ágeis através de BDD e outras documentações adicionais, sem perder as suas características de agilidade.

Na primeira etapa deste trabalho, apresentado no capítulo 4, foi buscado identificar os aspectos positivos e problemas relacionados a especificação e validação de requisitos, assim como mapear as práticas que permitiam auxiliar na melhoria dos desafios enfrentados. Essas informações serão utilizadas para embasar a proposta deste capítulo.

Este capítulo está estruturado em 3 seções. A seção 5.1 apresenta o processo de melhoria proposto e as documentações relacionadas a cada etapa. A seção 5.2 apresenta o quadro kanban aderente ao processo e demonstra como será organizada as documentações no quadro. A seção 5.3 apresenta a aplicação da proposta apresentada neste trabalho num ambiente real, assim como os resultados obtidos.

5.1 Processo Proposto

Para representar o processo de melhoria foi criado um diagrama com o auxílio da ferramenta Bizagi BPMN Modeler que permite a modelagem de diagramas, mapas e processos com o intuito de contribuir para que o processo proposto fique claro, organizado e de forma que seja facilmente compreendido. A figura apresentada no apêndice A deste trabalho, ilustra o processo proposto.

O processo foi criado tendo como base o modelo V, que sugere uma abordagem sequencial e sistemática do início ao fim do desenvolvimento do software [35]. O modelo representa a relação entre os dois lados do processo de desenvolvimento, este relacionamento é usado para determinar se a fase foi concluída com sucesso.

Neste modelo existe dependência entre as atividades, em cada atividade é necessário verificar e aprovar antes de avançar para a atividade seguinte, se ocorrer um problema durante a verificação ou validação de qualquer estágio, então o estágio anterior do V deve ser revisado e, se necessário, reiterado [5]. A importância de adotar o modelo V se deve ao fato do mesmo associar atividades de testes em todas as fases do processo de desenvolvimento de software, que devem ser executados para garantir a entrega de um produto de qualidade.

Para possibilitar a aplicação do processo na prática das empresas foi criado um quadro kanban que contempla todas as fases do processo conforme descrito na seção 5.2. Na sequência explicaremos cada fase proposta no processo, qual a ligação entre as fases, o objetivo de cada uma delas e a documentação sugerida para utilizar.

a) Visão de Negócio:

Conforme apresentado no apêndice A, a primeira etapa do processo ocorre na fase de especificação de requisitos e tem como objetivo levar a um entendimento sobre a visão de negócio em relação ao projeto a ser desenvolvido. Nesta fase os requisitos são descritos através das histórias de usuários. Como visto na fundamentação teórica deste trabalho a insuficiência de documentação é um desafio comum em um desenvolvimento ágil.

A falta de participação constante do cliente é um problema recorrente no desenvolvimento de software, sendo assim a documentação é fundamental para o manter o acompanhamento do projeto e de possíveis mudanças [19].

Conforme Cao e Ramesh os métodos ágeis substituem a documentação convencional por uma documentação mínima, que possui um objetivo de usuário específico descrito nas histórias de usuários [11]. Quando o software a ser desenvolvido é grande ou complexo as histórias de usuário não transmitem as informações suficientes para o projeto de software [15]. A figura 5.1 ilustra um modelo de história de usuário.

DESCRIÇÃO HISTÓRIA (PRODUCT BACKLOG): Login no sistema
Como: <papel/função exercida pelo usuário> Usuário de sistema **Preciso:** <descrição do requisito> Realizar login no sistema **Para:** <descrição do objetivo/valor do negócio> Acessar minha área de trabalho

Figura 5.1 – Modelo de História de usuário

Na tentativa de amenizar este problema de insuficiência de documentação será apresentado um modelo de documentação adicional as histórias de usuários, que tem como objetivo complementar as informações contidas nas histórias de usuários com informações voltadas a regras de negócio utilizadas pelo cliente. Essas informações devem ser preenchidas juntamente com o PO e com os principais stakeholders. Essa documentação será baseada no modelo RN, que é uma documentação que segue a estrutura ilustrada na figura 5.2. A figura ilustra um exemplo da aplicação do modelo RN em relação a história de usuário 5.1.

DESCRIÇÃO HISTÓRIA (PRODUCT BACKLOG): Login no sistema
Como: <papel/função exercida pelo usuário> Usuário de sistema **Preciso:** <descrição do requisito> Realizar login no sistema **Para:** <descrição do objetivo/valor do negócio> Acessar minha área de trabalho
REGRAS DE NEGÓCIO RELACIONADAS A HISTÓRIA
- O QUE, QUANDO, PORQUE, QUEM (ATORES ENVOLVIDOS), EFEITO/CONSEQUÊNCIA, COMPORTAMENTO ESPERADO DO SISTEMA a)Se o usuário errar a senha mais de 3 vezes enquanto fizer login, o sistema deve bloquear o acesso por 10 minutos b)Se o usuário estiver logado no sistema , quando for acessar o sistema em outra sessão, o acesso do usuário deve ser bloqueado em outras máquinas c) Quando o usuário logado ficar sem mexer no sistema por mais de 30 minutos, a sessão de login deve expirar

Figura 5.2 – Modelo de documentação RN

Conforme pode ser observado, a documentação sobre a história vai sendo incorporada ao documento anterior da história com um novo bloco de informações. Os demais documentos serão adicionados seguindo o mesmo padrão.

b) Visão de Produto:

A segunda etapa do processo tem como objetivo propor uma maior visão sobre o produto a ser desenvolvido. Com o propósito de melhorar a visão de produto em relação a uma história de usuário será utilizado o BDD (Behavior Driven Development) que é uma abordagem que nos permite interpretar o comportamento do sistema através de um conjunto de cenários que descrevem como a aplicação ou a unidade do código deverão se comportar em determinada situação [45]. Esta abordagem é apresentada na seção 2.3 deste trabalho.

Além de contribuir no complemento da documentação o BDD permite que tenhamos uma maior clareza sobre a necessidade do cliente. Segundo [42] muitas vezes não existe transparência entre o problema e a solução. Os stakeholders muitas vezes não possuem um entendimento claro do que o sistema deve fazer ou como o sistema deve executar. O uso do BDD permite minimizar este problema evitando que as partes interessadas ocultem informações importantes sobre o comportamento desejado pelo sistema. A imagem 5.3 representa o uso do BDD.

DESCRIÇÃO HISTÓRIA (PRODUCT BACKLOG): Login no sistema
Como: <papel/função exercida pelo usuário> Usuário de sistema **Preciso:** <descrição do requisito> Realizar login no sistema **Para:** <descrição do objetivo/valor do negócio> Acessar minha área de trabalho
REGRAS DE NEGÓCIO RELACIONADAS A HISTÓRIA
- O QUE, QUANDO, PORQUE, QUEM (ATORES ENVOLVIDOS), EFEITO/CONSEQUÊNCIA, COMPORTAMENTO ESPERADO DO SISTEMA a)Se o usuário errar a senha mais de 3 vezes enquanto fizer login, o sistema deve bloquear o acesso por 10 minutos b)Se o usuário estiver logado no sistema , quando for acessar o sistema em outra sessão, o acesso do usuário deve ser bloqueado em outras máquinas c) Quando o usuário logado ficar sem mexer no sistema por mais de 30 minutos, a sessão de login deve expirar
BDDs RELACIONADOS A HISTÓRIA
Cenário 1: Login realizado com sucesso **Dado: <um estado conhecido>** O usuário informou usuário e senha incorretos **Quando: <um determinado evento ocorre>** Tentar realizar o login no sistema **Então: <o que deve ocorrer>** Acessar a área de trabalho do usuário no sistema **Cenário 2: Falha no login** **Dado: <um estado conhecido>** O usuário informou usuário e senha incorretos **Quando: <um determinado evento ocorre>** Tentar realizar o login no sistema **Então: <o que deve ocorrer>** Informar que o usuário e senha estão incorretos e pedir para informar novamente

Figura 5.3 – Exemplo utilizando o BDD

Ainda nesta etapa de visão do produto porém na fase do modelo V de especificação do sistema será adicionado uma documentação referente aos RNF (Requisitos não funcionais). Esta documentação tem o intuito de diminuir os problemas relacionados aos RNF que foi relatado por Ramesh et al [38]. Conforme os autores, muitas organizações não dedicam atenção para os requisitos não funcionais no início do projeto, essa falta de atenção resulta posteriormente em gargalos ocasionando redesenvolvimento. Ignorar requisitos não funcionais como segurança, manutenção, testabilidade e usabilidade durante o projeto pode resultar em um retrabalho futuro.

Neste processo serão definidos possíveis ocorrências para cada ameaça identificada em relação aos RNF. As possíveis ameaças são escritas e em paralelo a isso também são escritas as soluções para cada ameaça. Este documento deve ser preenchido junto com o PO e os stakeholders e é anexado ao bloco sequencial juntamente com as outras documentações relatadas acima. A figura 5.4 representa o documento relacionado aos RNFs que é adicionado a sequencia de documentação.

DESCRIÇÃO HISTÓRIA (PRODUCT BACKLOG): Login no sistema

Como: <papel/função exercida pelo usuário> Usuário de sistema
Preciso: <descrição do requisito> Realizar login no sistema
Para: <descrição do objetivo/valor do negócio> Acessar minha área de trabalho

REGRAS DE NEGÓCIO RELACIONADAS A HISTÓRIA

- O QUE, QUANDO, PORQUE, QUEM (ATORES ENVOLVIDOS), EFEITO/CONSEQUÊNCIA, COMPORTAMENTO ESPERADO DO SISTEMA
a)Se o usuário errar a senha mais de 3 vezes enquanto fizer login, o sistema deve bloquear o acesso por 10 minutos
b)Se o usuário estiver logado no sistema , quando for acessar o sistema em outra sessão, o acesso do usuário deve ser bloqueado em outras máquinas
c) Quando o usuário logado ficar sem mexer no sistema por mais de 30 minutos, a sessão de login deve expirar

BDDs RELACIONADOS A HISTÓRIA

Cenário 1: Login realizado com sucesso
Dado: <um estado conhecido> O usuário informou usuário e senha incorretos
Quando: <um determinado evento ocorre> Tentar realizar o login no sistema
Então: <o que deve ocorrer> Acessar a área de trabalho do usuário no sistema

Cenário 2: Falha no login
Dado: <um estado conhecido> O usuário informou usuário e senha incorretos
Quando: <um determinado evento ocorre> Tentar realizar o login no sistema
Então: <o que deve ocorrer> Informar que o usuário e senha estão incorretos e pedir para informar novamente

REQUISITOS NÃO FUNCIONAIS RELACIONADO A HISTÓRIA

PROBLEMAS/NECESSIDADES: **<descrição do problema, necessidade ou ameaça relacionado ao RNF>** Segurança
SOLUÇÃO: **<descrição das soluções encontradas>** Caracteres da senha devem estar ocultos

PROBLEMAS/NECESSIDADES: **<descrição do problema, necessidade ou ameaça relacionado ao RNF>** Usabilidade
SOLUÇÃO: **<descrição das soluções encontradas>** Execução da rotina deve ser amigável e ser feita em poucos passos

PROBLEMAS/NECESSIDADES: **<descrição do problema, necessidade ou ameaça relacionado ao RNF>** Desempenho
SOLUÇÃO: **<descrição das soluções encontradas>** Sistema deve realizar o login rapidamente

PROBLEMAS/NECESSIDADES: **<descrição do problema, necessidade ou ameaça relacionado ao RNF>** Portabilidade
SOLUÇÃO: **<descrição das soluções encontradas>** O sistema deve permitir o acesso o login de diferentes tipos de plataformas e SO

Figura 5.4 – Exemplo utilizando o modelo de documentação dos RNF

c) Visão de construção do produto:

A etapa de visão de construção do produto é realizada na fase do modelo V de projeto de sistema. Nesta etapa temos como objetivo detalhar todas as tarefas necessárias para projetar e desenvolver o sistema (arquitetura, banco de dados, projeto de interface, codificação, entre outras). Para esta etapa será vinculado um documento que possua a informação sobre cada tarefa, qual categoria de artefato pertence, qual o artefato vinculado e uma descrição auxiliar.

Este documento tem como objetivo de identificar os artefatos para cada unidade assim como permitir analisar o impacto no projeto no decorrer do desenvolvimento, pois conforme dito neste trabalho a arquitetura escolhida pela equipe de desenvolvedores no ínicio do projeto pode se tornar inadequada em fases posteriores com novos requisitos. [38]. Segundo os estudos de Cao e Ramesh [11], rever a arquitetura no decorrer do projeto pode ocasionar um aumento do custo, sendo que a única outra alternativa seria de jogar fora o código e reescreve-lo.

Este documento será anexado juntamente com os documentos descritos nas etapas acima formando assim uma sequência de blocos de arquivos que são associados a cada história de usuário. A figura 5.5 representa o modelo de documentação referente as tarefas para projetar e desenvolver o sistema anexada juntamente as documentações anteriores.

DESCRIÇÃO HISTÓRIA (PRODUCT BACKLOG): Login no sistema

Como: <papel/função exercida pelo usuário> Usuário de sistema
Preciso: <descrição do requisito> Realizar login no sistema
Para: <descrição do objetivo/valor do negócio> Acessar minha área de trabalho

REGRAS DE NEGÓCIO RELACIONADAS A HISTÓRIA

- O QUE, QUANDO, PORQUE, QUEM (ATORES ENVOLVIDOS), EFEITO/CONSEQUÊNCIA, COMPORTAMENTO ESPERADO DO SISTEMA
a)Se o usuário errar a senha mais de 3 vezes enquanto fizer login, o sistema deve bloquear o acesso por 10 minutos
b)Se o usuário estiver logado no sistema , quando for acessar o sistema em outra sessão, o acesso do usuário deve ser bloqueado em outras máquinas
c) Quando o usuário logado ficar sem mexer no sistema por mais de 30 minutos, a sessão de login deve expirar

BDDs RELACIONADOS A HISTÓRIA

Cenário 1: Login realizado com sucesso
Dado: <um estado conhecido> O usuário informou usuário e senha incorretos
Quando: <um determinado evento ocorre> Tentar realizar o login no sistema
Então: <o que deve ocorrer> Acessar a área de trabalho do usuário no sistema

Cenário 2: Falha no login
Dado: <um estado conhecido> O usuário informou usuário e senha incorretos
Quando: <um determinado evento ocorre> Tentar realizar o login no sistema
Então: <o que deve ocorrer> Informar que o usuário e senha estão incorretos e pedir para informar novamente

REQUISITOS NÃO FUNCIONAIS RELACIONADO A HISTÓRIA

PROBLEMAS/NECESSIDADES: **<descrição do problema, necessidade ou ameaça relacionado ao RNF>** Segurança
SOLUÇÃO: **<descrição das soluções encontradas>** Caracteres da senha devem estar ocultos

PROBLEMAS/NECESSIDADES: **<descrição do problema, necessidade ou ameaça relacionado ao RNF>** Usabilidade
SOLUÇÃO: **<descrição das soluções encontradas>** Execução da rotina deve ser amigável e ser feita em poucos passos

PROBLEMAS/NECESSIDADES: **<descrição do problema, necessidade ou ameaça relacionado ao RNF>** Desempenho
SOLUÇÃO: **<descrição das soluções encontradas>** Sistema deve realizar o login rapidamente

PROBLEMAS/NECESSIDADES: **<descrição do problema, necessidade ou ameaça relacionado ao RNF>** Portabilidade
SOLUÇÃO: **<descrição das soluções encontradas>** O sistema deve permitir o acesso o login de diferentes tipos de plataformas e SO

ID TAREFA	DESCRIÇÃO TAREFA PARA PROJETAR E DESENVOLVER A HISTÓRIA (SPRINT BACKLOG)	CATEGORIA DO ARTEFATO	ARTEFATO AUXILIAR VINCULADA	DESCRIÇÃO
1	Criação das tabelas no banco de dados	Banco de dados		Tabela de usuário deverá ser utilizada como chave primária de relacionamento em todas as tabelas.
2	Diagrama de classes com o projeto de arquitetura	Arquitetura		O diagrama de classes deve ser mantido sempre que forem criadas novas tabelas e novas funcionalidades.
3	Projeto da interface	Interface	RNF1 - Usabilidade	
4	Definição de botões padrão a ser utilizado no sistema	Imagem	RNF1 - Usabilidade	
5	Programação da funcionalidade conforme especificado nos documentos (História, RN e BDDs)	Código	RN1, BDD1, BDD2	
6	Reaproveitamento de componente relacionado a registro de logs do sistema. Gravar IP, usuário, data, hora, número de tentativas	Componente	RNF2 - Segurança	

Figura 5.5 – Documento referente as tarefas para projetar e desenvolver o sistema

Os documentos criados em relação a visão de negócio e visão de produto foram anexados em um quadro juntamente com as histórias de usuários e fazem parte da etapa de planejamento da iteração, conforme representado na figura.

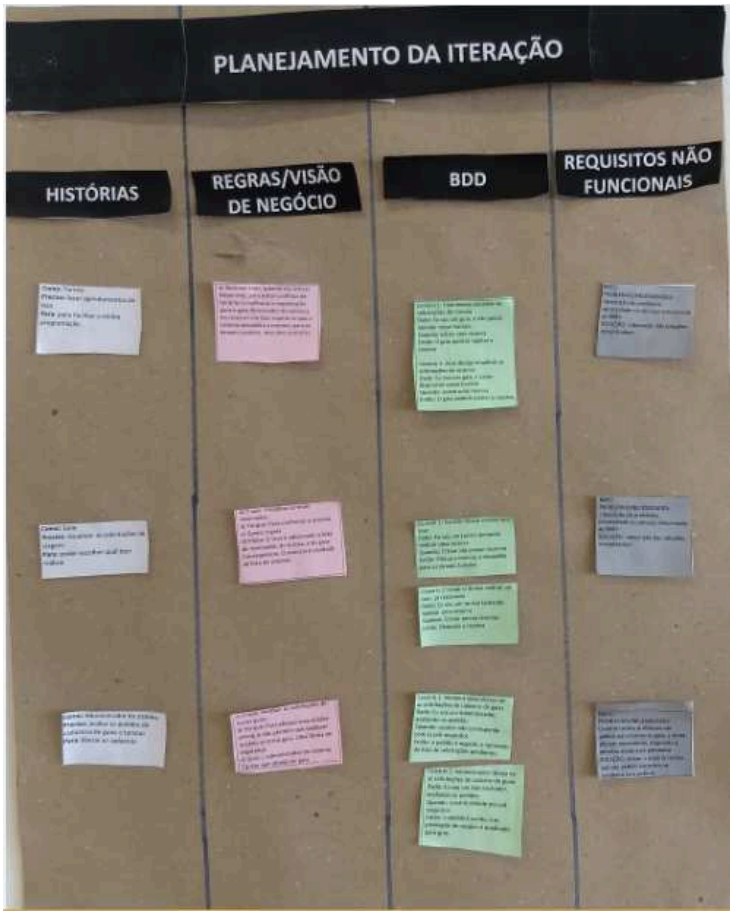

Figura 5.6 – Imagem que ilustra a etapa de planejamento da iteração

d) Desenvolvimento

A próxima etapa do processo, conforme apresentado no apêndice A, é a etapa de desenvolvimento que está subdividido em: tarefas a fazer, fazendo e pronto.

Tarefas a fazer: Nesta fase é onde a equipe irá trabalhar no desenvolvimento do projeto. Após o processo de levantamento das histórias e da documentação adicional relacionada a elas, as histórias são quebradas em tarefas conforme mostra a figura 5.7, com o objetivo de agilizar e facilitar o desenvolvimento. Nesta coluna estão todas as tarefas que estão para serem feitas. Quando um componente da equipe vai realizar a tarefa , o mesmo deve deslocar a atividade para o quadro ao lado "Fazendo".

Fazendo: Nesta coluna do processo é onde ficam todas as atividades que estão em andamento. Quando um membro da equipe vai desenvolver uma tarefa, ele deve deslocar a mesma para essa coluna. Isto permite ter um controle de quem está realizando determinada tarefa. Ao

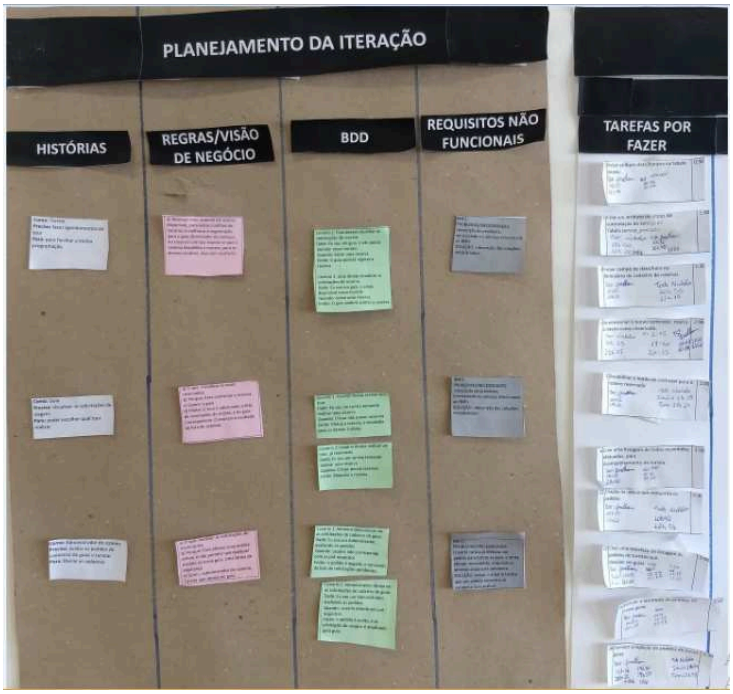

Figura 5.7 – Para seguir para a próxima etapa as histórias são quebradas em tarefas

deslocar a tarefa para esta coluna deve-se anexar os documentos que foram planejados anteriormente a cada tarefa, conforme mostra a imagem 5.8. Após a finalização da tarefa deve-se deslocá-la para a coluna ao lado "Pronto".

Pronto: Após a execução da tarefa, a mesma deve ser deslocada para a coluna de "Pronto", todas as atividades que foram desenvolvidas porém ainda não testadas aguardam nessa coluna para passar por uma bateria de testes. O primeiro teste a ser realizado pelo desenvolvedor é o teste de code review.

e) Teste de code review

Testando: Na etapa de teste de code review é onde o modelo V começa a avançar para a execução dos testes a serem realizados. Neste primeiro momento o foco é a identificação de problemas de bug no código, por isso denominado de code review. Caso reprovar no teste a mesma volta para coluna "A Fazer", e é retornada até ser aprovada. O teste de code review nos permite identificar uma possível falha logo no inicío do projeto.

Pronto: Se a tarefa testada for aprovada no teste de code review a mesma é deslocada para a coluna de "Pronta", indicando que foi aprovada e está no aguardo para passar para a

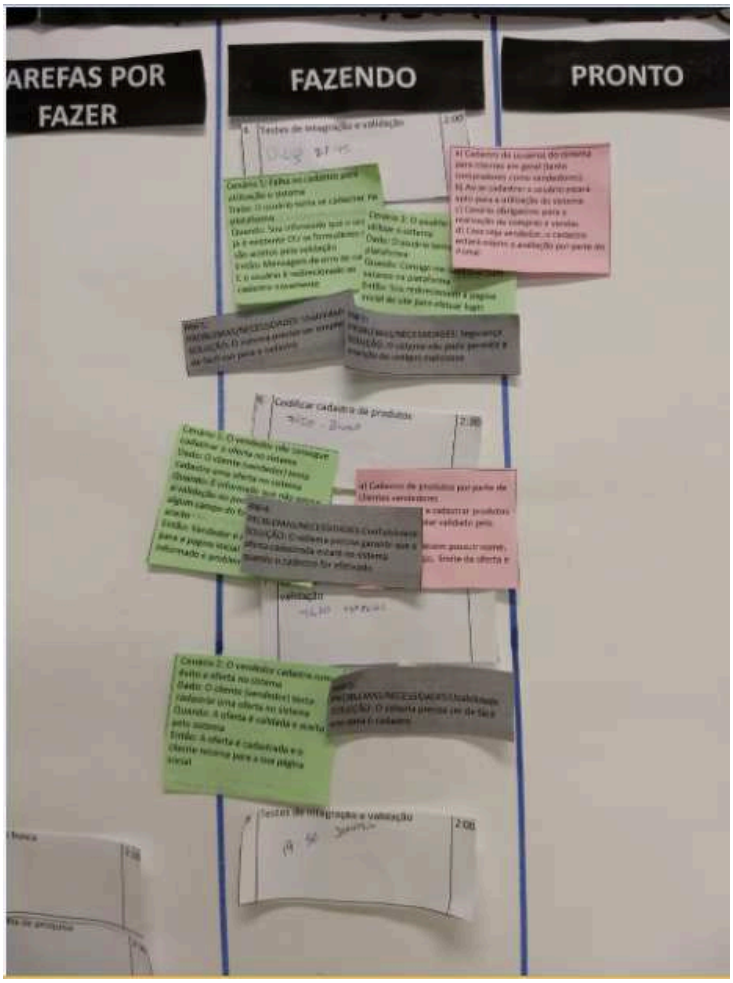

Figura 5.8 – Imagem que ilustra a associação das documentações as tarefas

próxima etapa de testes.

f) Teste de documentos complementares

Na etapa anterior foram realizados os testes no código procurando identificar problemas neste aspecto. Nesta etapa o foco são os testes de sistema, procurando validar a visão do produto e do negócio, identificados nos documentos conforme descrito a seguir:.

Validando BDDs, RNFs e RNs: Após a tarefa ser aprovada no teste de code review , a mesma é deslocada para a coluna "Validando BDDs, RNFs e RNs". Nesta etapa são realizados todos os testes referentes a documentações associadas a tarefa e é onde é simulado as operações de rotina do sistema de modo de verificar se seu comportamento está de acordo com o solicitado pelo cliente.

Os documentos que serão validados incluem documentos RN (regra de negócios), e documentos relacionados a produto como BDD e RNF (Requisitos não funcionais). Caso a tarefa for aprovada , passa para a próxima etapa caso contrário volta para "a fazer", onde é refeita e testada novamente até ser aprovada.

Tarefas Concluídas: Toda tarefa que passar pelo teste de unidade e tiver as documentações associadas a ela aprovada será classificada como "Concluída".

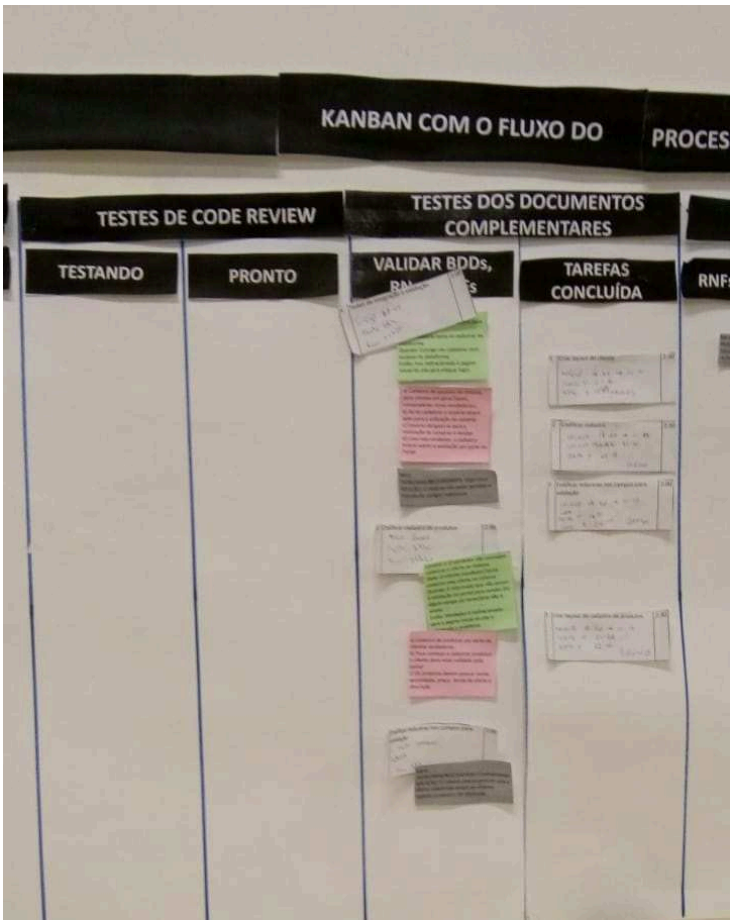

Figura 5.9 – Imagem que ilustra os testes de code review e testes dos documentos complementares

g) Testes concluídos de documentos

Após passar por todos os testes os documentos são deslocados para a coluna correspondente ao seu tipo de documento:

RNF testados: Coluna onde são associados todos os documentos de RNF testados.

BDDs testados: Coluna onde são associados todos os documentos de BDD testados.

Regras de negócio testadas: Coluna onde são associados todos os documentos de regra de negócio testados.

História testadas: Coluna onde são associadas todas as histórias de usuários testados.

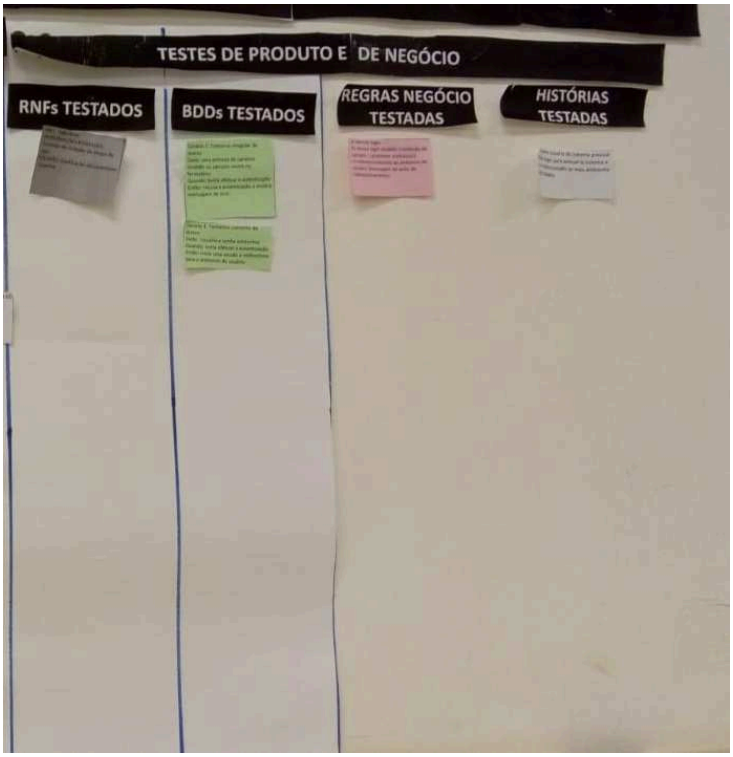

Figura 5.10 – Imagem que ilustra os testes concluídos de documentos

- **Review/Retrospectiva**

Ao final da sprint foi realizado a sprint review, onde o projeto foi avaliado em relação aos objetivos do sprint, quais foram alcançados e quais as falhas ocorridas. Durante essa reunião apresentada pelo scrum master participaram a equipe e o PO. Após a review foi realizado a retrospectiva onde foi discutido e identificado o que funcionou bem, o que pode ser melhorado e que ações serão tomadas para melhorar.

5.2 Quadro Kanban aderente ao processo e documentos propostos

O Kanban é um mecanismo de controle de fluxo, em que as atividades de processamento são desencadeadas pelos sinais de demanda do processo. O quadro é a principal ferramenta utilizada para visualizar e coordenar o trabalho em equipe. As colunas representam a sequência de atividades, onde os cartões que representam as tarefas são colocados [14]. O capítulo 2.5 traz uma explicação mais detalhada sobre o kanban, assim como apresenta um modelo genérico de quadro.

Baseado na proposta de melhoria do processo proposta na seção 5.1 adaptamos dois quadros kanban que estão representados nas figura 5.11 e 5.12, os mesmos foram estruturados de acordo com o processo apresentado no apêndice A. O primeiro quadro está dividido em 5 colunas, que demonstram o planejamento da iteração que representa a etapa antes do desenvolvimento onde são estruturadas as histórias de usuários, adaptada as documentações adicionais junto a essas histórias e onde as histórias são divididas em tarefas caso necessário. O segundo quadro representa o fluxo de desenvolvimento dos requisitos, desde a parte de definição das tarefas até os testes.

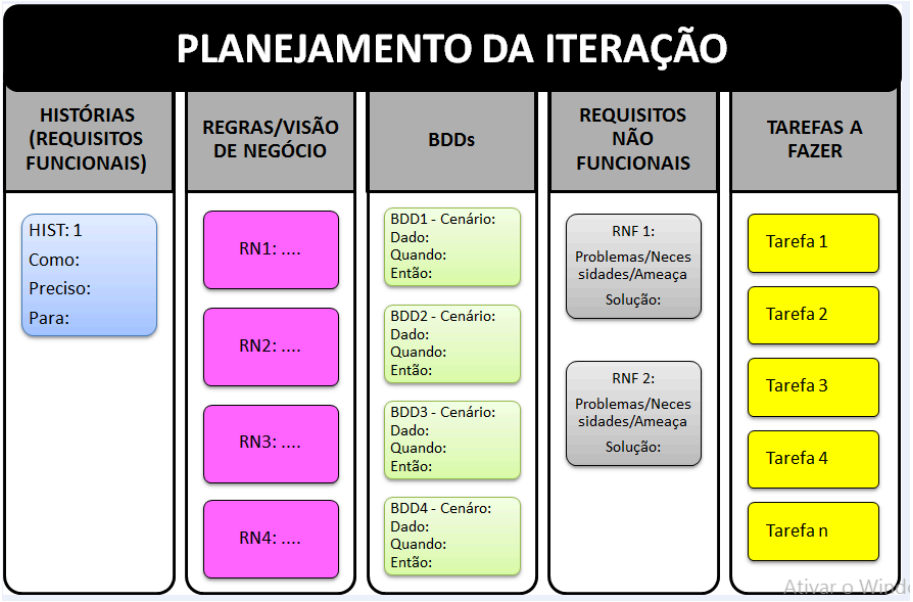

Figura 5.11 – Quadro Kanban referente ao planejamento da iteração

No quadro de planejamento da iteração representada na figura 5.11 é onde será associado as documentações necessárias a partir das histórias de usuário. Na primeira coluna do quadro representa as histórias de usuários construídas junto com o cliente. A segunda coluna representa documentos relacionados a regras de negócios que permitem auxiliar no entendi-

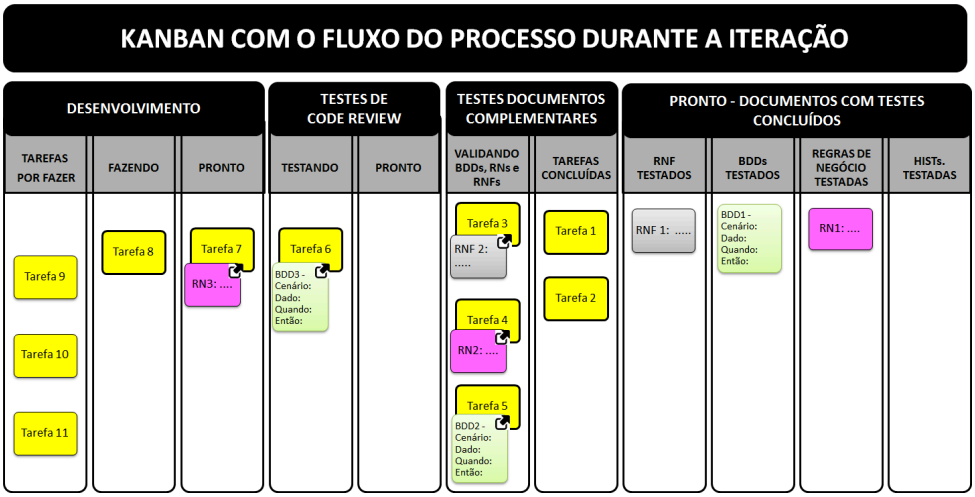

Figura 5.12 – Quadro Kanban que representa o fluxo do desenvolvimento

mento das regras de negócio. Na terceira coluna a documentação é baseada no BDD que nos permite construir os cenários de testes. A quarta coluna é associado os requisitos não funcionais. A última coluna é onde são apresentadas as tarefas que derivam das histórias de usuários, necessárias para projetar e desenvolver as funcionalidades da história.

O quadro kanban que representa o fluxo do processo de desenvolvimento durante a iteração, está divido em 4 etapas do processo que são divididas em 11 colunas, cada uma dessas colunas foram explicadas na seção 5.1. A primeira etapa representa o desenvolvimento, onde as tarefas do quadro anterior são deslocadas para a coluna de tarefas a fazer e se movem dentro desta coluna conforme o andamento da construção da tarefa, as documentações criadas no quadro anterior são associadas a cada tarefa representando assim uma sequência de arquivos conforme podemos ver na imagem 5.8.

Após a etapa de desenvolvimento são realizados os testes de code review na tentativa de encontrar bugs. Na terceira coluna é onde realizamos os testes referente as documentações associadas a história. Na última coluna é onde é associamos os testes realizados com as histórias de usuários e as documentações associadas a elas.

5.3 Aplicação da proposta e análise dos resultados

Na última etapa desta pesquisa foi realizada a avaliação do processo proposto, onde aplicou-se a proposta de melhoria em um ambiente simulado de desenvolvimento ágil. Após a aplicação foi realizado a coleta de dados assim como foi feita a análise dos resultados obtidos.

O ambiente simulado de densenvolvimento ágil utilizado para o estudo foi no compo-

nente curricular Planejamento e Gestão de Projetos do Curso de Ciência da Computação da Universidade Federal da Fronteira Sul (UFFS) que tem como objetivo ensinar os alunos a gerenciar projetos de software e desenvolvem um software no decorrer do semestre.

Neste ambiente existiam três equipes de desenvolvimento, cada uma delas formada por quatro alunos. Cada equipe desenvolveu projetos diferentes, porém todos foram orientados a seguir a mesma metodologia proposta neste trabalho, assim como o mesmo conjunto de práticas da metodologia Scrum.

Devido a limitação de tempo para a realização deste trabalho e pelo fato dos times desenvolverem os projetos durante as aulas, a aplicação e a análise foram realizadas com base em duas iterações. A primeira iteração teve 3 dias de desenvolvimento, a equipe seguiu a metodologia scrum porém não utilizou a melhoria proposta neste trabalho. Já na iteração secundária que também foi de 3 dias a equipe seguiu o processo de melhoria proposto neste trabalho, assim como as documentações criadas para auxiliar no entendimento dos requisitos.

Na sequência serão apresentados os resultados de uma das equipes que foram acompanhadas. As seções 5.3.1 e 5.3.2 explicam as iterações 1 e 2 respectivamente, e a seção 5.3.3 apresenta um comparativo entre as duas iterações.

5.3.1 Primeira iteração

Na primeira iteração as equipes não utilizaram a proposta de melhoria apresentada neste trabalho. O desenvolvimento baseou-se na metodologia ágil scrum, onde as equipes utilizaram várias práticas ágeis, dentre elas: Planejamento da sprint, backlog do produto, reuniões diárias, história de usuários, planning poker, priorização de requisitos, uso de ferramenta visual (quadro kanban) e gráfico burndown.

Nesta etapa a equipe iniciou o processo desevolvendo parte das histórias de usuários levantadas no início do projeto. As histórias de usuário eram a única documentação disponível nesta iteração, ficando notório a dificuldade dos membros da equipe em entender alguns requisitos e o que o cliente realmente precisava. Os testes eram exploratórios e sem documentação formal e a validação do produto foi baseada na descrição das histórias de usuários. A figura 5.13 representa o processo da sprint 1.

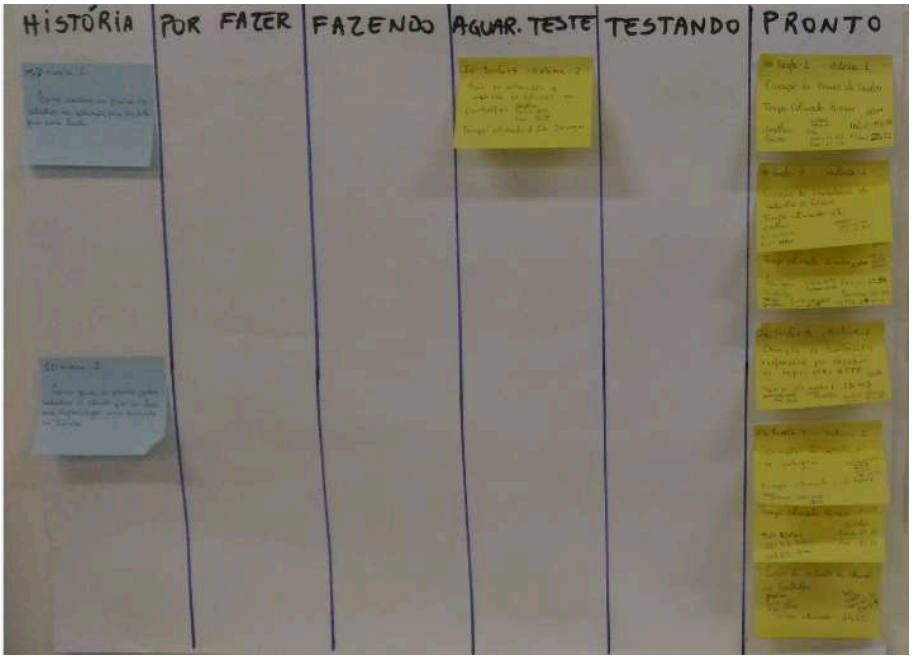

Figura 5.13 – Quadro Kanban referente a primeira sprint

5.3.2 Segunda iteração

Nesta seção será descrita a segunda iteração realizada pelas equipes. Inicialmente, cabe destacar que nesta iteração os times utilizaram a abordagem proposta neste trabalho, tendo anexado documentações relacionadas a visão de negócio, visão de produto e visão de construção do produto, assim como foram realizados testes em diferentes etapas do projeto.

O objetivo desta iteração era de melhorar a qualidade dos requisitos desenvolvidos, proporcionando um melhor entendimento das necessidades do cliente por parte dos desenvolvedores. As documentações associadas foram simples e objetivas para que o projeto não perdesse suas caracteristicas de um projeto ágil.

Inicialmente as equipes descreveram as histórias de usuários, e a partir dela as regras de negócios, os possíveis cenários através do BDD, assim como a as ameaças e soluções referentes aos requisitos não funcionais através dos documentos propostos neste trabalho. Após isso associaram as documentações a cada tarefa derivada das histórias de usuários.

Em seguida as equipes começaram o processo de desenvolvimento, após finalizar uma tarefa foi realizado o teste de code review com a finalidade de encontrar bugs relacionados a problemas de código logo no início da iteração, as tarefas que não passaram inicialmente no teste foram corrigidas e reiteradas, até serem aprovadas.

Após o teste de code review foi realizado os testes referentes as documentações asso-

ciadas onde foi simulado as operações de rotina do sistema de modo de verificar se o comportamento da rotina está de acordo com o esperado pelo cliente. As tarefas reprovadas no teste foram corrigidas e reiteradas até serem aprovadas. As tarefas aprovadas foram classificadas como concluídas. A imagem 5.14 representa este processo.

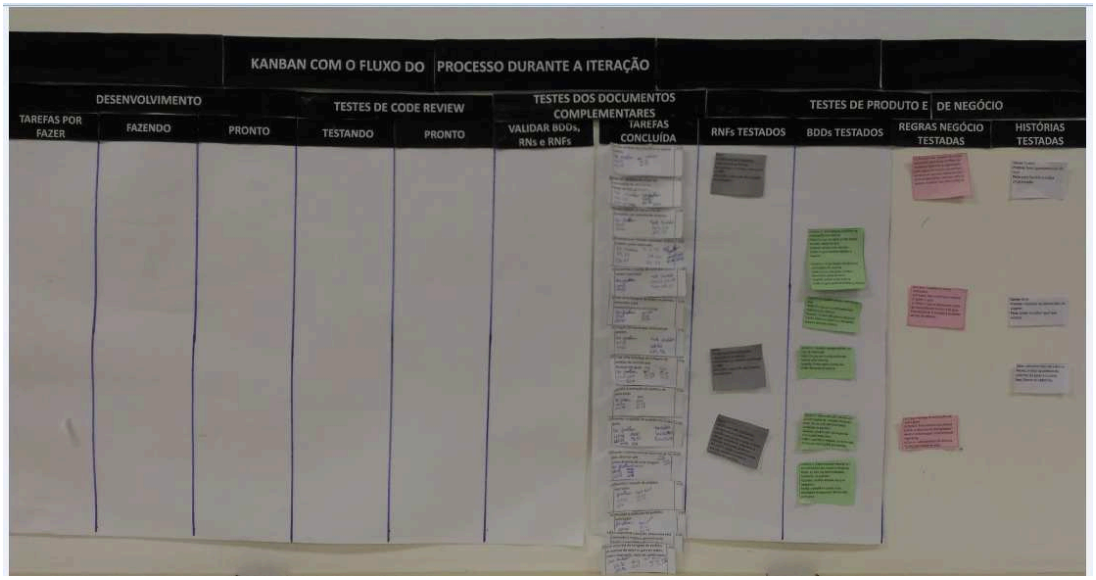

Figura 5.14 – Quadro Kanban após a conclusão da segunda sprint

5.3.3 Análise comparativa dos resultados das iterações

Nesta seção serão comparados os resultados das iterações, bem como será avaliada a abordagem proposta neste trabalho com base nos resultados obtidos.

Um dos fatores analisados na comparação entre as iterações foi em relação ao tempo gasto para realizar cada sprint. Na primeira iteração as equipes de modo geral apresentaram uma pequena margem de atraso. Na segunda sprint o time iniciou com atraso no primeiro dia, porém conseguiu avançar significativamente a partir do segundo dia, e concluiu todos as tarefas no prazo estabelecido.

As imagens 5.15 e 5.16 ilustram os gráficos burndown das sprints, que nos permite visualizar o tempo gasto e o tempo estimado de uma das equipes para cada dia de desenvolvimento das sprints.

Em relação a qualidade dos requisitos desenvolvidos cabe destacar que na segunda iteração todos os times apresentaram uma melhora significativa comparado a sprint 1. Os requisitos estavam mais detalhados e completos, proporcionando assim um melhor entendimento por parte dos desenvolvedores.

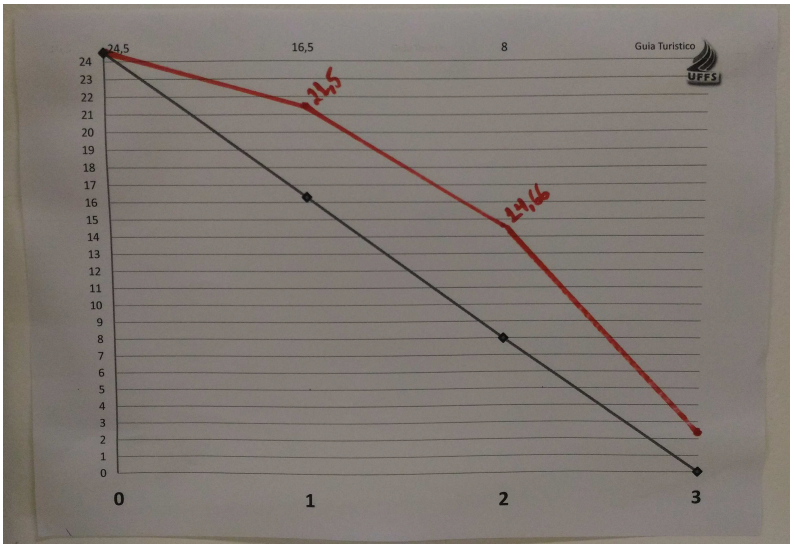

Figura 5.15 – Gráfico Burndown referente a primeira sprint

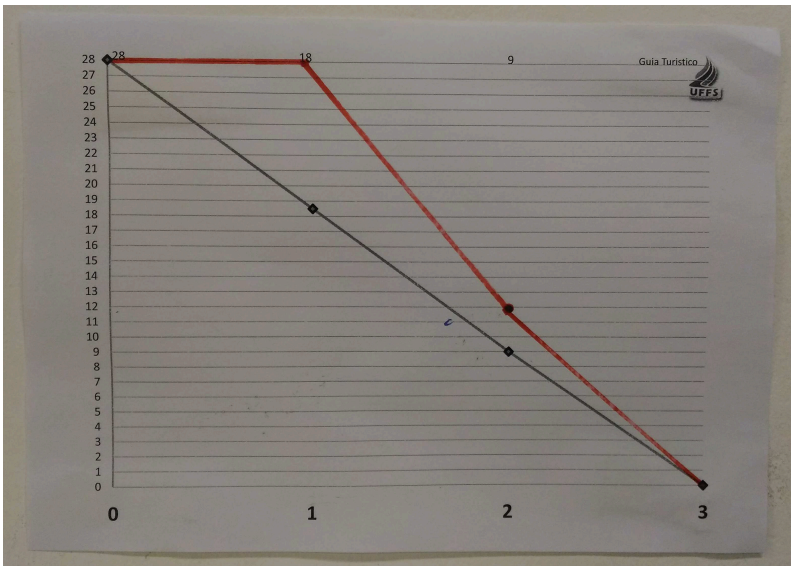

Figura 5.16 – Gráfico Burndown referente a segunda sprint

Outro fator comparativo analisado foi a quantidade de histórias aprovadas pelo PO. Na segunda iteração os resultados foram iguais ou maiores em relação aos da primeira iteração. O nível de satisfação do cliente melhorou pelo fato de que a documentação adicional permitiu ao o cliente expressar melhor a sua necessidade, melhorando a comunicação com a equipe, assim como proporcionou ao cliente uma visão mais detalhada sobre como se comportaria o sistema, evitando expectativas irrealistas.

Após da conclusão da segunda sprint foram aplicados questionários com as pessoas das equipes, com o objetivo de saber o quanto o novo processo melhorou o desenvolvimento. Foram feitas as seguintes perguntas:

a) O quanto a nova documentação auxiliou para compreender e desenvolver um produto aderente às necessidades do cliente:

( ) Não auxiliou em nada

( ) Auxiliou razoavelmente

( ) Auxiliou significativamente

b) O quanto a novo processo auxiliou a melhorar o processo dos testes nas etapas de especificação e validação de requisitos:

( ) Não auxiliou em nada

( ) Auxiliou razoavelmente

( ) Auxiliou significativamente

Das respostas obtidas para a pergunta ’a’ 87.50% responderam "Auxiliou significativamente"e 12.50% responderam "Auxiliou razoavelmente".

Das respostas obtidas para a pergunta ’b’ 87.50% responderam "Auxiliou significativamente"e 12.50 % "Auxiliou razoavelmente".

Relatos de membros das equipes extraído dos comentários:

> "O processo auxiliou bastante, foi possível compreender melhor o funcionamento da ferramenta (regra de negócio), e avaliar o comportamento em determinadas situações. Além de permitir a visualização de todo o andamento do projeto, por todos os membros da equipe."
>
> "A melhoria nos testes foi significativa, aumentou a comunicação e aproximação entre quem desenvolvia e o responsável pelo teste."
>
> "Os BDDs contribuem bastante para o processo como um todo. Ficou muito mais fácil elaborar testes e definir tarefas. Tornou o processo significativamente mais transparente."

A partir da comparação realizada entre os dados das 2 sprints e através dos dados das entrevistas, é possível concluir que a abordagem apresentada neste trabalho atende aos objetivos aos quais se propõe, uma vez que de sua aplicação prática foram obtidos resultados satisfatórios.

6 CONSIDERAÇÕES FINAIS

Neste trabalho foi apresentada uma abordagem que teve como objetivo melhorar o processo de especificação e validação de requisitos em projetos que utilizam métodos ageis. Para que os objetivos deste trabalho fossem alcançados foi estruturado a metodologia de pesquisa apresentada em 4 etapas: (i) Revisão da Literatura; (ii) Prosposta de melhoria do processo; (iii) Proposta de documentação; e (iv) Aplicação da proposta e análise de resultados;

Na Etapa 1 foram atingidos os seguintes objetivos específicos: (a) "Identificar os aspectos positivos e os problemas que ocorrem na especificação de requisi-tos nos métodos ágeis."; (b) "Identificar os aspectos positivos e os problemas que ocorrem na validação de requisitosnos métodos ágeis."; e (c) "Mapear práticas que permitam auxiliar na melhoria da especificação e validação de requisitos nos métodos ágeis, relacionando quais podem ser aplicadas em cada etapa doprocesso de desenvolvimento ágil."

Os dados encontrados nesta esta da pesquisa proporcionaram uma base de compreensão sobre o comportamento de um projeto que utiliza métodos ágeis, em sequência foi analisado a frequência de citação de cada aspecto positivo, problema e prática encontrada para que fosse identificado quais relacionavam uma quantidade maior de recorrência, consistindo assim em boas práticas para serem seguidas, principais problemas a serem resolvidos e práticas mais eficientes para solucionar esses problemas.

Sobre a etapa de especificação de requisitos, foi possível identificar que os aspectos positivos mais citados foram a colaboração ativa do cliente, a priorização de requisitos e o uso de histórias de usuários. Os problemas mais recorrentes foram as informações insuficientes contidas nas histórias de usuários, mudança de requisitos e a falta de conhecimentos técnicos do cliente.

Sobre a etapa de validação de requisitos, foi possível identificar que os aspectos positivos mais citados foram a grande quantidade de testes no início do desenvolvimento e a iteração com o cliente. Os problemas mais recorrentes foram a insdisponibilidade do cliente e dificuldade em entender o que o cliente realmente precisa.

Apesar de muitas práticas serem adotadas, observa-se que não se consegue resolver de forma efetiva os problemas relatados.

Na Etapa 2 foi proposta a melhoria do processo para atender o objetivo específico (d) "Propor a melhoria do processo de especificação e validação de requisitos baseando-se no mo-

delo V, definindo um fluxo de atividades que possa ser aplicado de forma visual em um quadro kanban", onde foram definidas as etapas deste processo e quais atividades seriam realizadas em cada uma destas etapas.

Para a apresentação desta abordagem foram considerados os problemas identificados na revisão de literatura e especificados no capítulo 4 deste trabalho. A fim de auxiliar na construção e visualização do funcionamento da abordagem proposta, foi elaborado um diagrama de processo de negócios na notação BPMN (apêndice A) que mapeia todo o processo proposto a ser seguido, permitindo o acompanhamento passo a passo das etapas e atividades a serem realizadas.

Na etapa 3 deste trabalho foram criadas documentações adicionais para atender o objetivo especifico (e) "Propor um modelo de documentação para auxiliar na especificação e validação de requisitos nos métodos ágeis através de BDD e outras documentações adicionais, sem perder as suas características de agilidade".

Para atingir este objetivo foram definidos novos documentos que deveriam ser associados as histórias de usuários. Os documentos criados foram simples e objetivos com a intenção de resolver problemas de insuficiência de documentos, porém mantendo as características de documentação dos métodos ágeis.

O primeiro documento criado em anexo as histórias de usuário foi o documento RN (Regras de Negócio) que melhora a visão do negócio relacionado a história do usuário. Na sequência foram associados outros dois documentos referente a visão de produtos, sendo o BDD voltado aos requisitos funcionais e também uma documentação voltada aos Requisitos não funcionais. Por último foi proposto um novo documento para melhorar as informações sobre a construção do produto e do projeto do sistema.

Desta forma foi possível contribuir para a melhora da documentação nas várias etapas de projeto do sistema, primeiro melhorando a visão de negócio, posteriormente melhorando a visão do produto a ser desenvolvido e por último auxiliando no projeto de construção do produto.

Na etapa 4, buscando avaliar a proposta, foi realizada a aplicação desta através de um ambiente simulado de desenvolvimento de software ágil, realizada a coleta de dados e interpretação dos resultados. A partir do que foi apresentado através da análise e comparação entre as duas iterações realizadas pelas equipes pode-se afirmar que os objetivos propostos neste trabalho foram atingidos.

6.1 Trabalhos Futuros

Neste trabalho foi apresentada uma abordagem elaborada para a melhoria do processo de especificação e validação de requisitos em projetos ágeis de software, baseado no modelo V. No entanto este trabalho teve o foco voltado especificamente para as duas primeiras fases do modelo V, etapas de especificação de requisitos e especificação de sistema. Assim recomenda-se os seguinte trabalhos futuros:

- Continuidade na realização de estudos nas fases de especificação de requisitos e especificação de sistema, adotando métricas para avaliação e comprovando a eficiência do projeto de forma quantitativa.

- Apresentar uma proposta de melhoria nas fases de arquitetura de sistema e projeto de unidade do modelo V.

- Realizar estudos de caso em projetos reais, aplicando a abordagem proposta em empresas que adotam métodos ágeis (como empresas privadas e outro setor público).

REFERÊNCIAS

[1] Manifesto ágil. Disponível em: http://agilemanifesto.org, 2001. acessado em 13/06/2017.

[2] Significado de kanban. Disponível em: https://www.significados.com.br/kanban/, 2011. acessado em 15/08/2017.

[3] M. O. Ahmad, J. Markkula, and M. Oivo. Kanban in software development: A systematic literature review. In *Software Engineering and Advanced Applications (SEAA), 2013 39th EUROMICRO Conference on*, pages 9–16. IEEE, 2013.

[4] D. d. C. P. ALVES. Engenharia de requisitos em projetos ágeis: um mapeamento sistemático baseado em evidências da indústria. 2015.

[5] S. Balaji and M. S. Murugaiyan. Waterfall vs. v-model vs. agile: A comparative study on sdlc. *International Journal of Information Technology and Business Management*, 2(1):26–30, 2012.

[6] R. Baskerville, B. Ramesh, L. Levine, and J. Pries-Heje. High-speed software development practices: What works, what doesn't. *IT professional*, 8(4):29–36, 2006.

[7] D. M. Berry. The inevitable pain of software development, including of extreme programming, caused by requirements volatility. *Eberlein and Leite*, 2002.

[8] E. Bjarnason, P. Runeson, M. Borg, M. Unterkalmsteiner, E. Engström, B. Regnell, G. Sabaliauskaite, A. Loconsole, T. Gorschek, and R. Feldt. Challenges and practices in aligning requirements with verification and validation: a case study of six companies. *Empirical Software Engineering*, 19(6):1809–1855, 2014.

[9] E. Bjarnason, M. Unterkalmsteiner, M. Borg, and E. Engström. A multi-case study of agile requirements engineering and the use of test cases as requirements. *Information and Software Technology*, 77:61–79, 2016.

[10] E. Bjarnason, K. Wnuk, and B. Regnell. A case study on benefits and side-effects of agile practices in large-scale requirements engineering. In *Proceedings of the 1st Workshop on Agile Requirements Engineering*, page 3. ACM, 2011.

[11] L. Cao and B. Ramesh. Agile requirements engineering practices: An empirical study. *Software, 25(1):60–67*, 2008.

[12] R. Carlson et al. Applying scrum to stabilize systems engineering execution. *CrossTalk*, 2012.

[13] A. Cezerino and F. P. Nascimento. Utilização da técnica de desenvolvimento orientado por comportamento (bdd) no levantamento de requisitos. *Revista Interdisciplinar Científica Aplicada*, 10(3):40–51, 2016.

[14] E. Corona and F. E. Pani. A review of lean-kanban approaches in the software development. *WSEAS Transactions on Information Science and Applications*, 10(1):1–13, 2013.

[15] Daneva et al. Agile requirements prioritization in large-scale outsourced system projects: An empirical study. *Journal of Systems and Software, 86(5):1333–1353,*, 2013.

[16] A. De Lucia and A. Qusef. Requirements engineering in agile software development. *Journal of emerging technologies in web intelligence*, 2(3):212–220, 2010.

[17] M. Diepenbeck, M. Soeken, D. Große, and R. Drechsler. Behavior driven development for circuit design and verification. In *High Level Design Validation and Test Workshop (HLDVT), 2012 IEEE International*, pages 9–16. IEEE, 2012.

[18] K. Elghariani and N. Kama. Review on agile requirements engineering challenges. In *Computer and Information Sciences (ICCOINS), 2016 3rd International Conference on*, pages 507–512. IEEE, 2016.

[19] R. Goetz. How agile processes can help in time-constrained requirements engineering. *In Proceedings of the international workshop on time constrained requirements engineering.*, 2002.

[20] D. D. Gregorio. How the business analyst supports and encourages collaboration on agile projects. In *Systems Conference (SysCon), 2012 IEEE International*, pages 1–4. IEEE, 2012.

[21] B. Haugset and T. Stalhane. Automated acceptance testing as an agile requirements engineering practice. In *Proceedings of the 2012 45th Hawaii International Conference*

on System Sciences, HICSS '12, pages 5289–5298, Washington, DC, USA, 2012. IEEE Computer Society.

[22] P. Heck and A. Zaidman. A systematic literature review on quality criteria for agile requirements specifications. *Software Quality Journal*, pages 1–34, 2016.

[23] V. T. Heikkilä, D. Damian, C. Lassenius, and M. Paasivaara. A mapping study on requirements engineering in agile software development. In *Software Engineering and Advanced Applications (SEAA), 2015 41st Euromicro Conference on*, pages 199–207. IEEE, 2015.

[24] M. Ikonen, E. Pirinen, F. Fagerholm, P. Kettunen, and P. Abrahamsson. On the impact of kanban on software project work: An empirical case study investigation. In *Engineering of Complex Computer Systems (ICECCS), 2011 16th IEEE International Conference on*, pages 305–314. IEEE, 2011.

[25] I. Inayat, L. Moraes, M. Daneva, and S. S. Salim. A reflection on agile requirements engineering: solutions brought and challenges posed. In *Scientific Workshop Proceedings of the XP2015*, page 6. ACM, 2015.

[26] I. Inayat, S. S. Salim, S. Marczak, M. Daneva, and S. Shamshirband. A systematic literature review on agile requirements engineering practices and challenges. *Computers in human behavior*, 51:915–929, 2015.

[27] D. Janzen and H. Saiedian. Test-driven development concepts, taxonomy, and future direction. *Computer*, 38(9):43–50, 2005.

[28] G. Kotonya and I. Sommerville. Requirements engineering with viewpoints. *Software Engineering Journal*, 11(1):5–18, 1996.

[29] O. Ktata and G. Lévesque. Agile development: Issues and avenues requiring a substantial enhancement of the business perspective in large projects. In *proceedings of the 2nd Canadian conference on computer science and software engineering*, pages 59–66. ACM, 2009.

[30] L. O. Lobo and J. D. Arthur. Local and global analysis: Complementary activities for increasing the effectiveness of requirements verification and validation. In *Proceedings of the 43rd annual Southeast regional conference-Volume 2*, pages 256–261. ACM, 2005.

[31] M. Mc Hugh, O. Cawley, F. McCaffcry, I. Richardson, and X. Wang. An agile v-model for medical device software development to overcome the challenges with plan-driven software development lifecycles. In *Software Engineering in Health Care (SEHC), 2013 5th International Workshop on*, pages 12–19. IEEE, 2013.

[32] L. C. P. Moraes. Um estudo empírico sobre o uso do bdd e seu apoio a engenharia de requisitos. Master's thesis, Pontifícia Universidade Católica do Rio Grande do Sul, 2016.

[33] N.A.Ernst and G. C. Murphy. Case studies in just-intime requirements analysis. *In Proc. Second International Workshop on Empirical Requirements Engineering (EmpiRE), pages 25–32. IEEE*, 2012.

[34] D. North. Introducing bdd. https://dannorth.net/introducing-bdd/. acessado em 13/06/2017.

[35] Pressman. In *Engenharia de Software: Uma abordagem profissional, sétima edição*. Porto Alegre: AMGH, 2011.

[36] R. Prikladnicki, R. Willi, and F. Milani. *Métodos ágeis para desenvolvimento de software*. Bookman Editora, 2014.

[37] Z. Racheva and M. Daneva. How do real options concepts fit in agile requirements engineering? In *Software Engineering Research, Management and Applications (SERA), 2010 Eighth ACIS International Conference on*, pages 231–238. IEEE, 2010.

[38] B. Ramesh, L. Cao, and R. Baskerville. Agile requirements engineering practices and challenges: an empirical study. *Information Systems Journal,20(5), 449–480.http://dx.doi.org/10.1111/j.1365-2575.2007.00259.x*, 2010.

[39] D. G. Reali. Análise e experiência no desenvolvimento guiado ao comportamento. 2015.

[40] D. F. Rico, H. H. Sayani, and S. Sone. *The business value of agile software methods: maximizing ROI with just-in-time processes and documentation*. J. Ross Publishing, 2009.

[41] P. Rook. Controlling software projects. *Software Engineering Journal*, 1(1):7–16, 1986.

[42] A. Rudorfer, T. Stenzel, and G. Herold. A business case for feature-oriented requirements engineering. *IEEE software*, 29(5):54–59, 2012.

[43] M. Sayão and J. C. S. do Prado Leite. Rastreabilidade de requisitos. *RITA*, 13(1):57–86, 2006.

[44] A. M. Sen and K. Hemachandran. Elicitation of goals in requirements engineering using agile methods. In *2010 IEEE 34th Annual Computer Software and Applications Conference Workshops*, pages 263–268, July 2010.

[45] T. R. Silva. Definition of a behavior-driven model for requirements specification and testing of interactive systems. In *Requirements Engineering Conference (RE), 2016 IEEE 24th International*, pages 444–449. IEEE, 2016.

[46] J. F. SMART. In *BDD in Action*. Manning Publications, 2015.

[47] SOMMERVILLE. Engenharia de software, oitava edição. *7. ed. Boston: Addison Wesley*, 2007.

[48] Sommerville. In *Engenharia de Software, nona edição*. São Paulo: Pearson Prentice Hall, 2011.

[49] M. Stoica, M. Mircea, and B. Ghilic-Micu. Software development: Agile vs. traditional. *Informatica Economica*, 17(4):64, 2013.

APÊNDICES

APÊNDICE A – Mapeamento do processo de melhoria proposto neste trabalho

Neste apêndice é apresentado o mapeamento do processo a ser seguido para a implementação prática da abordagem de melhoria do processo de especificação e validação de requisitos nos métodos ágeis proposto neste trabalho, o qual está representado através de um diagrama criado com a notação BPMN (Business Process Modeling Notation).

Para a criação do diagrama foi utilizada a ferramenta Bizagi BPMN Modeler versão 3.1.0.011 (https://www.bizagi.com/pt/produtos/bpm-suite/modeler), que permite a modelagem de processos na notação BPMN. Os elementos gráficos da notação BPMN que foram utilizados neste trabalho e seus significados, de acordo com as explicações providas pelo software utilizado, são apresentados na figura A1.

Elemento Gráfico	Nome do elemento	Grupo	Descrição
	Evento de início	Fluxo	O evento de início indica onde o processo começará
	Evento de fim	Fluxo	O evento de fim indica onde um o processo terminará
	Tarefa	Fluxo	Uma tarefa é uma atividade atômica que está incluída dentro de um processo
	Gateway	Fluxo	Locais dentro de um processo de negócios onde o fluxo de sequência pode tomar dois ou mais caminhos alternativos
	Objeto de Dados	Dados	Fornece informações sobre como documentos, dados e outros objetos são usados e atualizados durante o processo.
	Fluxo de sequência	Conectores	É usado para mostrar a ordem em que as atividades serão executadas em um processo. Cada fluxo tem uma só origem e um só destino

Figura A.1 – Elementos gráficos da notação BPMN

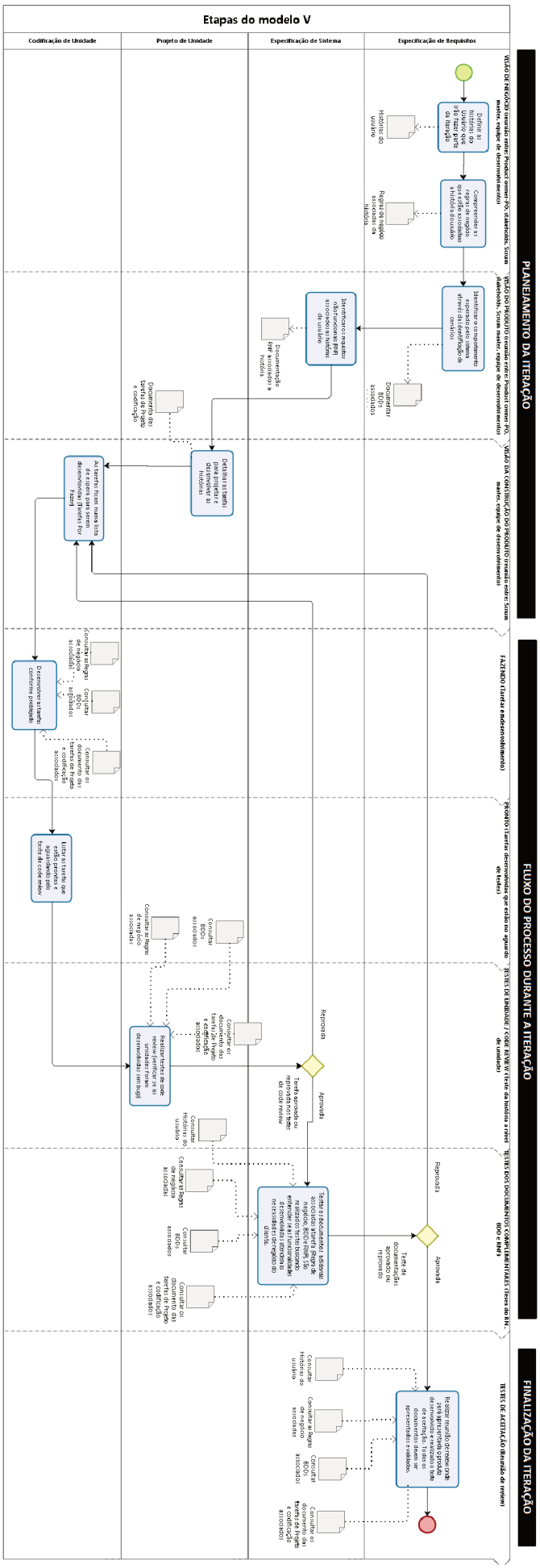

Figura A.2 – Diagrama que representa o processo proposto