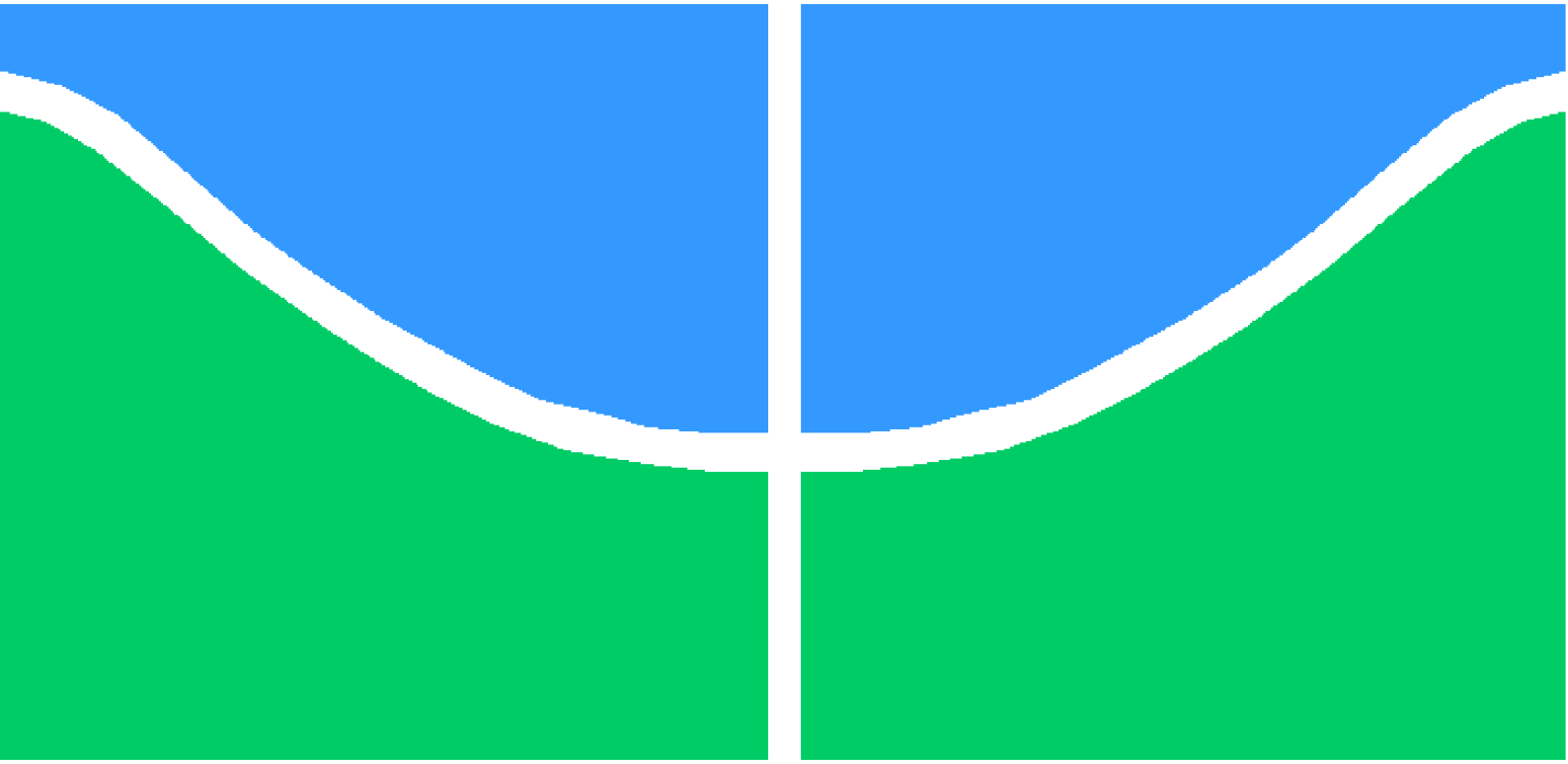

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Relátorio de Projeto

Grupo 4: Danilo Barros, Henrique Lopes, Thiago Moreira, Vitor Bertulucci

Orientador: Elaine Venson, MSc

Brasília, DF

2016

Danilo Barros, Henrique Lopes, Thiago Moreira, Vitor Bertulucci

Relátorio de Projeto

Relatório submetido à disciplina de Engenharia de Requisitos, do curso de graduação em Engenharia de Softwareda Universidade de Brasília.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Elaine Venson, MSc

Brasília, DF

2016

Resumo

O presente documento foi criado por alunos da disciplina de Engenharia de Requisitos e tem como objetivo determinar o processo de engenharia de requisitos para o contexto da empresa real Azura - Soluções em Software. Para tanto este documento aborda todas as vertentes necessárias para uma boa gerencia e execução da engenharia de requisitos, definindo a melhor metodologia a ser utilizada dado o contexto da empresa, o processo de engenharia de requisitos e todas as suas atividades, incluindo atividades inerentes do modelo de maturidade de processos utilizados. Este documento também trata de ações como elicitação de requisitos e suas técnicas mais apropriadas para o contexto, assim como a ferramenta de gestão de requisitos utilizada e a política de gerência utilizada, descrevendo seus atributos e rastreabilidade.

Palavras-chaves: engenharia de requisitos. requisitos. azura. software.

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

RUP	Rational Unified Process
SAFe	Scaled Agile Framework
PO	Product Owner
PM	Product Manager
MPS.BR	Melhoria de Processos de Software Brasileiro

Sumário

1 Introdução

1.1 Finalidade

Este documento tem como objetivo determinar o processo de engenharia de requisitos que será utilizado para a criação dos requisitos do software que solucionará problemas de uma empresa real: Azura - Soluções em Software.

1.2 Escopo

Requisitos de software são descrições das funções, propriedades e restrições que um produto ou sistema a ser desenvolvido deve possuir. A construção dos requisitos deve atender as condições necessárias para satisfazer um objetivo específico.

Paralelamente, a Engenharia de Requisitos engloba o conjunto de todas atividades que contribuem de forma concisa as condições necessárias que o software deve atender. Ganha-se notoriedade cinco das atividades de todo o processo: elicitação de requisitos, análise e negociação, documentação, verificação e validação e gerência. Dessa forma, este documento irá tratar de todo o processo para a solução dos problemas da empresa em questão.

1.3 Visão Geral

Esse documento possui os seguinte capítulos, que estruturam todo o trabalho e representam:

- Capitulo 2: Contexto

  Descrição da empresa Azura, em relação a sua estrutura, contexto de trabalho, filosofia e objetivos, bem como a visão geral de seu problema.

- Capitulo 3: Metodologia

- Capitulo 4: Processo de Engenharia de Requisitos

  Engloba a justificativa da escolha de metodologia para desenvolvimento dos requisitos e as etapas definidas do processo de Especificação de Software.

- Capitulo 5: Elicitação de Requisitos

  Aborda as técnicas de Elicitação de Requisitos escolhidas com o cliente, abordando cada técnica individualmente.

- Capitulo 6: Gerência de Requisitos

  Descreve dados sobre gerenciamento de requisitos e seus atributos para alcançar a rastreabilidade e gerenciamento do requisito.

- Capitulo 7: Ferramenta de Gerência de Requisitos

  Define as ferramentas utilizadas de acordo com critérios definidos e escolhidos para avaliar um conjunto de três ferramentas.

- Capitulo 8: Planejamento do Projeto

- Capitulo 9: Considerações Finais

2 Contexto

2.1 Azura Soluções em Software

Azura Software é uma empresa de desenvolvimento que vai além da construção de *sites* e aplicativos, atuando também na área de *marketing* digital e identidade visual, focando em um processo e produto inovador e acima da curva em relação a qualidade do mercado.

- Objetivos: o principal objetivo da Azura é ampliar as fronteiras de alcance das empresas clientes a partir do trabalho realizado pela organização, transmitindo confiabilidade e, acima de tudo, compromisso com os projetos. A ideia é conciliar as técnicas de cada especialidade da Azura com o fluxo da empresa, criando o melhor conteúdo possível e desenvolvendo um produto de qualidade com um bom custo benefício.
- Missão: Facilitar a vida das pessoas e o processo das empresas clientes utilizando conceitos de engenharia de *software*, publicidade e computação gráfica.

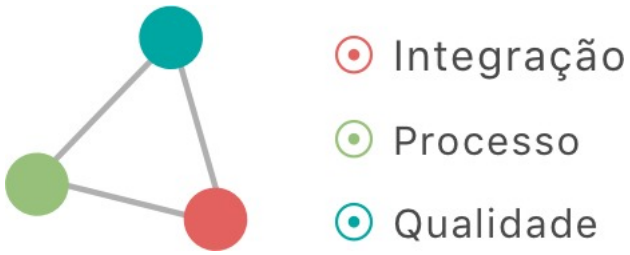

Figura 1 – Significado da logo Azura

A logomarca da empresa reflete diretamente no proposito de seus produtos e serviços. Os 3 propósitos são bem difundidos entre os membros da empresa, e eles precisam necessariamente de sincronia e excelência.

2.2 Organização da Empresa

Atualmente, com exceção dos cargos de direção, a Azura Software é dividida em duas hierarquias, sendo elas: Desenvolvedores e Publicitários. O cargo de desenvolvedor é destinado ao time de desenvolvimento, que tem o papel de codificar o software. Os publicitários são destinados ao *marketing* digital e identidade visual dos produtos, trabalhando principalmente com design. Apesar da hierarquia presente, existem funcionários que tem

capacitação pra duas áreas, sendo híbridos em relação aos cargos. Já os cargos executivos estão divididos em presidência e diretor, sendo um pra cada especialidade da empresa (Identidade visual, *marketing* digital, WEB e *mobile*). O cargo da presidência lida com funções referentes a análise de custos e controle de clientes, admissão e demissão de membros, promoções de cargos e controle de custos. Os gerentes lidam com supervisionamento e planejamento dos projetos ativos, lidando com possíveis problemas relacionados aos processos (AZURA, 2016).

Figura 2 – Organograma da empresa Azura Software

2.3 Descrição do problema

Como a própria marca da empresa ressalta, o processo é fundamental e afeta diretamente nos produtos, serviços e tarefas da empresa. Porém, o o processo é o principal problema a espera de uma solução efetiva por meio de *software*.

Atualmente, diversas empresas trabalham com mais de um projeto sendo produzido em conjunto, o que exige maturidade da equipe de gerência para conseguir conciliar e alocar os recursos, tanto materiais quanto humanos, disponíveis para dar andamento e cumprir com prazos. A Azura Software, por se tratar de uma empresa recente e ainda com poucos integrantes, possui um problema no que diz respeito à gerência dos projetos, pois, como adotam a metodologia ágil em seu processo, aloca, em cada um deles, um responsável pela gerência dos mesmos. A ideia é ter um gerente para cada projeto, o qual ficará responsável também por desenvolver, mas que terá mais peso na parte da organização e se tentar ainda mais nas realizações das atividades, controle de custos e organização de dados.

Neste contexto, a companhia conta com o contato constante de clientes e realiza o desenvolvimento dos projetos todos em paralelo, o que tem causado uma certa desorganização no que se refere ao controle de clientes e seus respectivos projetos e prazos, por falta de um sistema de gerência dos mesmos.

3 Metodologia

3.1 Metodologia de desenvolvimento

O avanço nas tecnologias trouxe consigo o aumento na complexidade nos projetos e consequentemente a necessidade de avançar também nas metodologias adotadas dentro dos mesmos, visando melhorar tanto a produtividade quanto qualidade. É de extrema importância o estudo sobre qual metodologia se aplica mais corretamente para cada projeto e é papel do engenheiro decidir a que melhor se adequa, para diminuir os riscos e aumentar as chances de sucesso do mesmo (IEEE, 2014). Seguindo como base esta linha de raciocínio, será definido qual abordagem melhor se aplica diante do projeto da Azura Software.

3.1.1 Rational Unified Process (RUP)

Em busca de um processo unificado, em 1990, foi criado uma metodologia que combinaria as melhores características de cada um dos modelos convencionais de um processo de *software*, sendo eles: casos de uso, centrado na arquitetura, iterativo e incremental (PRESSMAN, 2016). O RUP adotou essas características reconhecendo a importância da comunicação com o cliente e descrever sua visão acerca do sistema, o importante papel da arquitetura em compreender o *software*, e por fim, um fluxo de processo iterativo e incremental, que seria uma resposta aos pontos fracos do modelo tradicional (SOMMERVILLE, 2007).

3.1.1.1 Estrutura do RUP

O RUP utiliza em seu processo Disciplinas e Fases. Como pode ser observado na Figura 3, as disciplinas estão na parte vertical do gráfico, que representam as atividades que ocorrem no processo. As Fases estão representadas na parte horizontal do gráfico, onde estão relacionadas mais estritamente aos negócios do que a assuntos técnicos no processo de *software* (SOMMERVILLE, 2007).

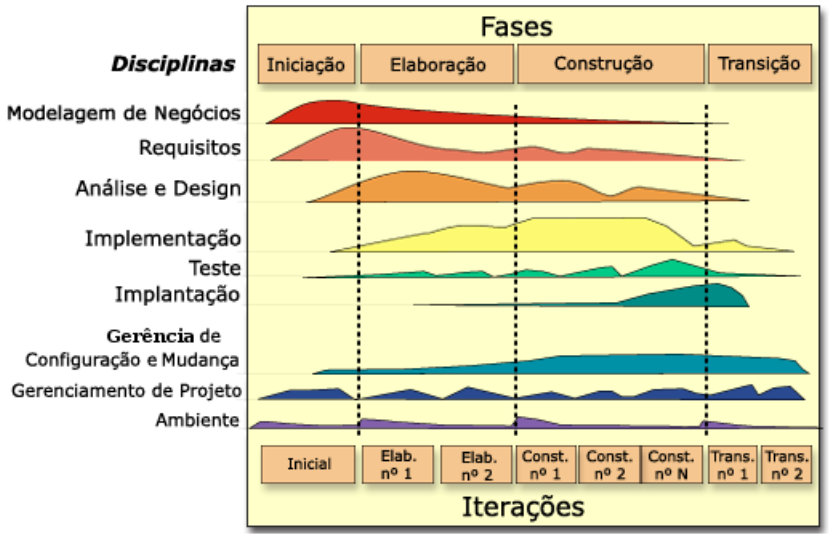

Figura 3 – Gráfico do workflow do RUP

As quatro fases técnicas do RUP são:

- Iniciação: devem ser identificadas todos os *stakeholders* que irão interagir com o sistema e definir suas interações com o *software*, com o intuito de avaliar a contribuição de cada envolvido dentro do projeto.

- Elaboração: a implementação do código já ocupa grande parte do esforço da equipe, visto que essa fase tem o foco na garantia de uma arquitetura instável. Visando mitigar os riscos, são implementados os requisitos mais significativos do sistema.

- Construção: esta fase está essencialmente relacionada ao projeto, programação e teste do sistema. As partes do sistema são desenvolvidas paralelamente e integradas durante esta fase. Ao final deve-se ter um sistema de *software* em funcionamento e a documentação associada pronta para ser liberada para os usuários.

- Transição: nesta fase ocorre a passagem do sistema que foi desenvolvido para os usuários/clientes do sistema, já em funcionamento e pronto para uso.

3.1.2 Metodologia Ágil

Com os avanços nas tecnologias e também nos modelos de processos, que não estavam sendo tão eficazes, percebeu-se a necessidade de inovar com o que diz respeito aos processos de desenvolvimento de *software*. Com isso, por volta da década de 90, começaram a produzir um novo conceito sobre produção de *softwares*, que estava mais focada no produto em si do que em documentações ou arquitetura do *software*, conhecida como metodologia ágil.

Os métodos tradicionais de desenvolvimento são focados na geração de documentos sobre o projeto em questão e está interessado em processos e ferramentas. Já os métodos ágeis focam as atenções na constante entrega do produto e nas interações entre os indivíduos.

A metodologia ágil tem como principais objetivos promover a interação contínua entre o cliente e a equipe do projeto e adaptação contínua a mudanças, com o intuito de tornar a comunicação mais rápida e a troca de informações mais concisas, visando a melhoria constante no processo, para alcançar os objetivos mais rapidamente, evitar falhas no meio do processo e aumentar a produtividade.

3.1.2.1 SAFe

SAFe ou *Scaled Agile Framework* é uma base de conhecimento criada por Dean Lefingwell, baseada em princípios do Ágil e *Lean*, utilizado com sucesso em grandes empresas de software e que vem crescendo a cada dia. O SAFe tenta unir o que os métodos ágeis - em geral - têm de melhor, levando conhecimentos do *Scrum*, *Extreme Programming* (XP) e do *Lean*, anteriormente mencionado. Isso proporciona uma orientação compreensível para o trabalho relacionado ao portfólio, fluxo de valor, programa e o nível dos times de uma companhia (LEFFINGWELL, 2016).

Pode-se resumir o processo e abordagens do SAFe de acordo com a Figura 4, a qual é conhecida dentro do *framework* como *Big Picture*, onde a metodologia é disposta dentro de seus três existentes subníveis.

3.1.3 Análise de metodologia com a problemática

De acordo com (BOEHM; TURNER, 2009), existem quatro principais considerações que devem ser tomadas diante da natureza do desenvolvimento de *software*, que são: Características da aplicação; Características de gerenciamento; Características técnicas; Características das pessoas.

3.1.3.1 Característica da aplicação

3.1.3.1.1 Objetivos

Dentro das metodologias de processos, o processo unificado defende que os requisitos são mais estáveis e melhor definidos, ou seja, que não sofrem tanta alteração e, se sofrerem, é algo previsível (BOEHM; TURNER, 2009). Já o manifesto ágil diz que em um projeto de *software*, os requisitos podem sofrer alterações constantes, e em muitos casos, os mesmos não são definidos totalmente ou corretamente antes do projeto se dar início (PRESSMAN, 2016), e o time deve ser "ágil"o suficiente para conseguir dar continuidade ao projeto sem deixá-lo ser afetado.

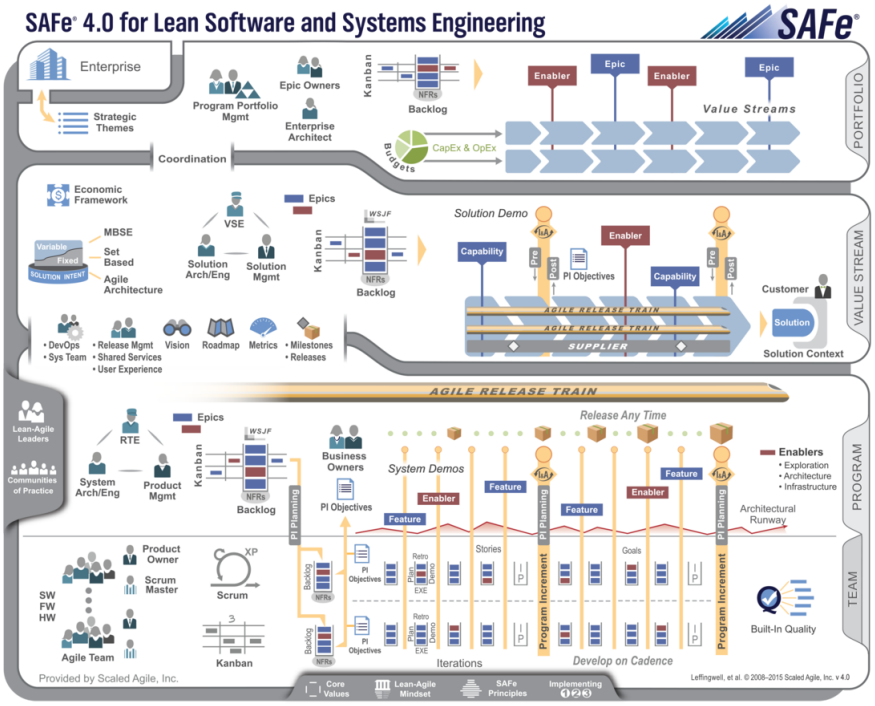

Figura 4 – Processo do SAFe 4.0

Dentre o problema encontrado na empresa Azura, o cliente necessita um sistema de controle de clientes, como o número de clientes, tempo para término dos projetos, custos, atrasos, mão-de-obra necessária, investimentos, manutenção, o que auxilia no levantamento de requisitos, mas que ainda não é certeza do cliente se é somente isso que o mesmo necessita. Com isso, a instabilidade e a incerteza no levantamento desses requisitos nessa fase inicial do projeto aponta mais para as características do manifesto ágil do que o modelo de processo unificado.

3.1.3.1.2 Tamanho do projeto

De acordo com (PRESSMAN, 2016), os processos ágeis funcionam melhor em projetos com o menor volume de pessoas, com a comunicação e colaboração constante e com pessoas motivadas. Já de acordo com (BOEHM; TURNER, 2009), o RUP necessita de um número mais significativo de artefatos e formalizações.

Neste contexto, o projeto em questão conta com a participação de cinco membros, sendo que um deles é o cliente, responsável pela Azura Software, que realizará as devidas validações dos requisitos juntamente com a equipe, e o mesmo não está interessado em artefatos mais sofisticados, por se tratar de um projeto de pequeno porte. Diante das informações propostas, pode-se perceber que o projeto aponta para os métodos ágeis, em relação ao tamanho do projeto.

3.1.3.2 Característica de gerenciamento

Na metodologia ágil, os clientes devem estar intimamente envolvidos no processo de desenvolvimento, tendo o papel de fornecer e priorizar novos requisitos do sistema e avaliar suas iterações de maneira rápida, sempre visando uma boa evolução do projeto e constante *feedback*, sem necessidade de formalidades em artefatos (SOMMERVILLE, 2007). Diferente da Metodologia Ágil, a Metodologia Tradicional utiliza um processo com diversos artefatos, que servem como produtos tangíveis do projeto, sendo eles coisas que o projeto produz ou usa enquanto trabalha para o produto final. Iterações e incrementos no projeto são marcados por várias documentações, concisos e compreensíveis por uma ampla gama de usuários (KRUCHTEN, 2000).

Tendo como referencia a definição de SOMMERVILLIE e as características do cliente, que tem uma postura mais informal e não está interessado em artefatos, e sim em interação contínua com a equipe e *feedbacks* do andamento do projeto, a Metodologia Ágil seria a adequada para o projeto.

3.1.3.3 Características técnicas

Características técnicas tem sido o foco de muitos dos debates sobre efetividade de metodologias como ágil e *plan-driven*. Essa secção discute sobre como cada uma das abordagens atingem a elicitação e gerenciamento de requisitos, atividades de desenvolvimento e teste (BOEHM; TURNER, 2009).

3.1.3.3.1 Requisitos

Métodos ágeis contam com ciclos rápidos de interação para determinar mudanças nas capacidades e ajustar elas na próxima interação, o que define a área de requisitos como instável, lidando com adaptação constante, rápida e eficiente (BOEHM; TURNER, 2009). Encontrar os requisitos que são prioridade dentro do conjunto de atividades a serem realizadas é um trabalho em conjunto do time de desenvolvimento com o cliente, onde o time deve ter a capacidade de definir se a atividade a ser incluída na próxima interação é factível (BOEHM; TURNER, 2009).

Metodologias tradicionais geralmente preferem um ponto de partida mais formal, completa e consistente, ou seja, com requisitos melhor definidos e que não sofram muitas modificações - com uma base mais sólida (BOEHM; TURNER, 2009).

Como dito anteriormente, de acordo com o cliente, os requisitos não possuem essa base sólida como defendidos nos métodos tradicionais, onde os modelos ágeis como integração contínua, permitem a adaptação diante de uma base de requisitos mais instáveis, que podem ser modificados com o decorrer do processo.

Dentre as atividades citadas anteriormente, considerando o escopo da disciplina estão a elicitação e gerenciamento de requisitos. Já a área de desenvolvimento e testes não estão relacionados com o foco da disciplina de requisitos de *software*, e por isso não serão citados ou descritos.

3.1.3.4 Características das pessoas

3.1.3.4.1 Relação cliente-equipe

Na abordagem tradicional, a relação do cliente com a equipe se restringe a algumas fases, como a Gerência de Projetos e a Modelagem de negócio, onde o cliente deve entender suas necessidades e, a partir disso, as duas partes tenham convicção e entendimento comum a ser desenvolvido (KRUCHTEN, 2000).

Na metodologia ágil, onde a colaboração com o cliente é mais importante do que a negociação de contratos, a comunicação é livre e desimpedida de burocracias. As dúvidas que ocorrem e pendencias são resolvidas quase que imediatamente, impossibilitando desentendimentos. A clareza no projeto é sempre mantida com a comunicação constante com o cliente (AMBLER, 2002).

O cliente tem a característica de ser bem comprometido com o projeto em termos de presença continua e *feedbacks*, estando sempre disposto a ajudar a equipe em duvidas pontuais e buscando sempre melhorias, dispensando burocracias referentes a documentos. Essa característica ajuda equipe a ter provimento de informações a um curto prazo de tempo, sendo uma característica bem comum na metodologia ágil.

3.1.3.4.2 Time de desenvolvimento

Utilizada em diversos projetos pelo time de desenvolvimento, junto a experiência de alguns membros utilizando-a em estágios, a metodologia ágil mostra-se bastante eficaz sendo utilizada em uma equipe com características de boa comunicação, simplista e com bom relacionamento interpessoal.

3.2 Metodologia escolhida

3.2.1 Abordagem escolhida

A partir do estudo das metodologias propostas e apresentadas, juntamente com a definição do perfil do cliente, a equipe de engenharia de requisitos de *software* decidiu por adotar a metodologia ágil para o projeto, pois, de acordo com o que foi citado anteriormente, as características - aplicação, gerenciamento, técnicas e das pessoas - mostraram uma adequação muito focada nesta abordagem.

4 Processo de Engenharia de Requisitos

4.1 Scaled Agile Framework

Neste tópico será explicado do processo de Engenharia de Requisitos definido pela equipe, o qual será utilizado metodologia ágil - como citado anteriormente - e seguirá o processo do *Scaled Agile Framework* (SAFe).

4.1.1 Nível de Portfólio

O Portfólio do SAFe é o nível mais alto encontrado dentro do processo e fornece a construção básica para a organização do processo de organizações ágeis em torno do fluxo de valor para alcançar os objetivos estratégicos. Este nível encapsula esses elementos e fornece o orçamento básico e outros mecanismos necessários para assegurar que os investimentos e fluxos de valor promovam o retorno necessário para que a companhia conheça seus objetivos estratégicos (SAFE, 2015).

O nível de Portfólio possui duas ramificações relacionadas aos negócios. A primeira ramificação promove os temas estratégicos que irão guiar o portfólio para os grandes e variáveis objetivos de negócio. O outro ramo deste nível indica o fluxo constante do contexto do portfólio em relação a organização. Isso informa a empresa do atual estado do conjunto de soluções, assim como alguns indicadores de performance e outros fatores de negócio que afetam o portfólio (SAFE, 2015).

4.1.2 Nível de Fluxo de Valor

Este nível é aplicado, mais especificamente, em soluções complexas e sistemas críticos, que envolvem diversas equipes das mais diversas áreas de conhecimento, como *software, hardware*, elétrico e eletrônico, mecânica, etc (SAFE, 2015).

Em geral, o nível de Fluxo de Valor é aplicado a projetos de Engenharia de Requisitos que possuem um escopo muito grande, geralmente aplicado a empresas de grande porte. Por se tratar de um projeto curto, realizado em uma disciplina da Universidade de Brasília, este é o único nível que não será aplicado no contexto deste projeto.

4.1.3 Nível de Programa

Este nível é onde os times de desenvolvimento e outros recursos são aplicados em projetos de *software*. Os times, regras e atividades são organizadas seguindo a metáfora do *Agile Release Train* (ART), ou seja, um time ágil que aumenta o nível de entrega

de incrementos para o projeto aplicando os princípios e práticas do SAFe, tentando remover empecilhos do mesmo, como alguns passos - os menos relevantes - e intervenções desnecessárias (SAFE, 2015).

Mesmo sendo chamado de nível de programa, os ARTs são, em geral, mais duradouros e tem uma melhor estrutura, missão e organização própria se relacionado com "programas"tradicionais, os quais possuem datas de início e fim. Ou seja, as bases do ART são: possuir um ciclo de vida mais duradouro, com fluxo constante e organização própria. Essa natureza permite que os ART impulsionem os portfólio do SAFe (SAFE, 2015).

4.1.4 Nível de Time

Embora seja tratado como um tópico separado, o nível de Time do SAFe faz parte do nível de Programa. Todos os times do SAFe fazem parte do ART - nível centrado na construção do programa. O nível de Time promove a organização, regras, artefatos e modelo de processo para as atividades dos times ágeis. Cada time é responsável por definir, construir e testar as histórias selecionadas a partir do acúmulo de atividades ou backlog de atividades em interações com tamanhos fixos, utilizando uma cadência comum de interação e sincronização com os demais times relacionados ao projeto (SAFE, 2015).

Os times utilizam práticas do *ScrumXP* e fazem o uso do *Kanban* do time para, rotineiramente, entregar subsistemas de alta qualidade - incrementos. Isso assegura que todos os diferentes times presentes no ART criem um sistema integrado e testado que os clientes podem avaliar e dar retornos brevemente (SAFE, 2015).

4.1.5 Papéis no processo

Neste tópico definiremos alguns dos papéis presentes na metodologia adotada que serão utilizados no contexto do projeto e quais são as principais atividades que cada um destes papéis são responsáveis.

4.1.5.1 Product Owner

O *Product Owner* é, tipicamente, responsável por transmitir os interesses dos *stakeholders* para a Equipe *Scrum*, entendendo as necessidades e garantindo a solução correta para o problema. Com isso, o PO tem o papel de definir todos os itens priorizados no *Product Backlog* e garantir que toda funcionalidade implementada agregue o maior valor possível a aplicação, elaborando e validando histórias de usuário (SCHWABER, 2004).

O papel de *Product Owner* será exercido pelo cliente e gerente da Azura Software, Rafael Akiyoshi.

4.1.5.2 Product Manager

O *Product Manager* é a principal autoridade de conteúdo no processo, guiando o nível de programa, onde sua principal responsabilidade é o *backlog* de programa. Ele trabalha, assim como o *Product Owner*, juntamente com o cliente para entender as necessidades, definir requisitos e guiar o projeto. Como suas responsabilidades, encontra-se as atividades de: fornecer o rápido *feedback*, validar a resposta do cliente, criar da visão do projeto e realizar o planejamento de *releases* (SAFE, 2015).

Dentro da equipe, o papel de *Scrum Master* será revezado entre os membros ao final de cada *Sprint*.

4.1.5.3 Scrum Master

Esse papel é tipicamente designado ao gerente do projeto mas pode ser exercido por qualquer membro dentro da equipe, sendo responsável por ensinar o processo *Scrum* para a equipe, garantindo que todos sigam as regras e práticas do mesmo. Outra função do *Scrum Master* é manter a harmonia da equipe assegurando que quaisquer obstáculos no projeto sejam eliminados (SCHWABER, 2004).

Assim como o *Product Manager*, os papeis de *Scrum master* serão revesados dentro da equipe.

4.1.5.4 Scrum Team

Comumente chamado de time de desenvolvimento ou apenas time, neste papel não existe necessariamente uma divisão funcional, tendo todos trabalhando juntos para entregar uma parte funcional do produto ao final de cada *Sprint*. Sua função é, a partir do *Product Backlog*, selecionar os itens mais prioritários, se comprometendo a entregá-los ao final da iteração. Um time tipicamente tem de 6 a 10 pessoas (LIBARDI; BARBOSA, 2010).

Esse papel vai ser desempenhado em conjunto pelos alunos: Danilo Barros, Henrique Lopes, Thiago Moreira e Vitor Bertulucci.

4.1.6 Artefatos

4.1.6.1 Portfólio

- Temas de Investimento: São usados para guiar as prioridades de investimentos para a organização assegurando que o trabalho realizado esta alinhado a estrategia da organização. Esses temas direcionam a visão do portfólio que sera expressada através de diversas iniciativas épicas que são alocadas para *release* ao longo do tempo. (LEFFINGWELL, 2011)

- Épicos: Representam o mais alto nível da necessidade de um cliente, sendo iniciativas de desenvolvimento que tem como objetivo agregar valor ao tema de investimento. São identificados, priorizados, estimados e mantidos no *backlog* do portfólio. No planejamento de *releases* os épicos são decompostos em *features* especificas e posteriormente serão transformados em historias de usuário para implementação. (LEFFINGWELL, 2011)

- *Backlog* de Portfólio: É o artefato de mais alto nível presente na fase de Portfólio do SAFe. Detém e prioriza épicos que foram aprovados para a implementação (SAFE, 2015).

4.1.6.2 Programa

- Visão: Descreve uma visão futura da solução a ser desenvolvida, refletindo as necessidades do cliente e dos *stakeholders*, assim como as funcionalidades e as capacidades que estão propostas a solucionar tais necessidades. O documento visão também provê uma visão geral mais ampla e mais contextual da solução que está a ser desenvolvida (SAFE, 2015).

  Define o escopo de alto nível e o propósito de um programa, produto ou projeto, respondendo as grandes perguntas referentes a esses pontos. Uma instrução clara do problema, solução proposta e os recursos de alto nível do produto ajudam a estabelecer expectativas e a reduzir riscos. É essencial que todos os *stakeholders* envolvidos no projeto entendam esse artefato (IBM, 2015).

- Requisitos não funcionais: Documento contendo todos os requisitos não funcionais relacionados a necessidade do usuário. Descreve atributos do sistema, que englobam itens como: segurança, manutenção e usabilidade. É um documento de qualidades e restrições persistentes que é revisitado após cada definição de feito, para cada iteração, incremento de programa ou *release* (SAFE, 2015).

- *Roadmap*: Planos com alto nível de abstração que mostram a evolução pretendida do seu produto ao longo das próximas *releases*, dispostos em sequencia seguindo a priorização definida (SAFE, 2015).

- *Backlog* de programa: É o repositório definitivo para as próximas evoluções previstas para avançar o projeto, sendo um compilado de *features* que, no geral, atendem todas as necessidades do usuário e que irão agregar valor à aplicação (SAFE, 2015).

4.1.6.3 Time

- *Backlog* de Time: Representa uma coleção contendo tudo que o time precisa ou necessita fazer para avançar e incrementar o sistema. Ele contém histórias de usuá-

rio que são atualizadas por vários membros da equipe, mas o *Product Owner* que prioriza as histórias que serão implementadas primeiro (SAFE, 2015).

- *Backlog* de *Sprint*: Representa uma coleção com as histórias e tarefas que o time se compromete em fazer durante a próxima *sprint*, com base nas prioridades definidas pelo *Product Owner*. O *backlog* de *sprint* é atualizado constantemente pelo *Scrum Master*, para refletir o trabalho realizado pelo time de desenvolvimento.

- *Kanban*: Método que ajuda a equipe a enxergar o fluxo de trabalho de forma visual, contendo as histórias priorizadas na *sprint* e seu andamento. Nele está presente o *Backlog* de Time e os passos que representam o fluxo de trabalho, que está dividido em Análise, Revisão, Construção, Integração, Teste e se a história foi aceita (SAFE, 2015).

- Documento de Retrospectiva: Representa uma coleção contendo três listas com atividades, hábitos e itens que foram realizados na *sprint* que se passou. A lista de começar a fazer, contém os itens que o time acha relevante adicionar ao processo e a praticar para as próximas *sprints* a serem realizadas. A lista de parar de fazer, são listados as atividades os comportamentos que o time julga como nocivo ou que prejudicou de alguma forma o bom andamento do projeto. Por último é documentado a lista de continuar a fazer, que contém as atividades que o time julga como benéficas para o projeto e que deve ser mantida para próximas iterações, porém uma vez que tais atividades se tornam hábitos elas são retiradas dessa lista.

4.1.7 Descrição de atividades

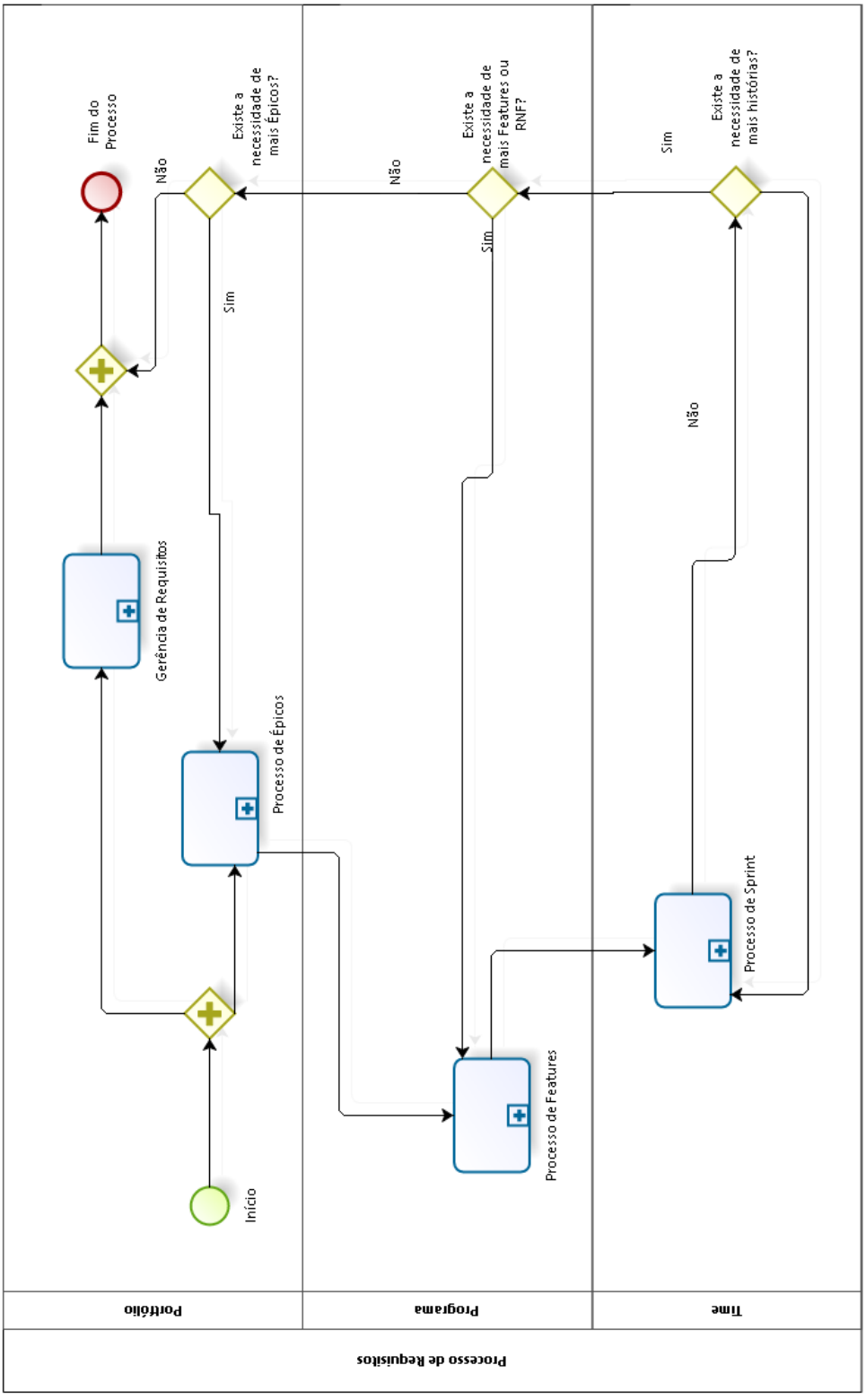

Figura 5 – Modelagem de Processo

4.1.7.1 Portfólio

Descreve-se aqui as atividades referentes ao nível de Portfólio proposto no SAFe.

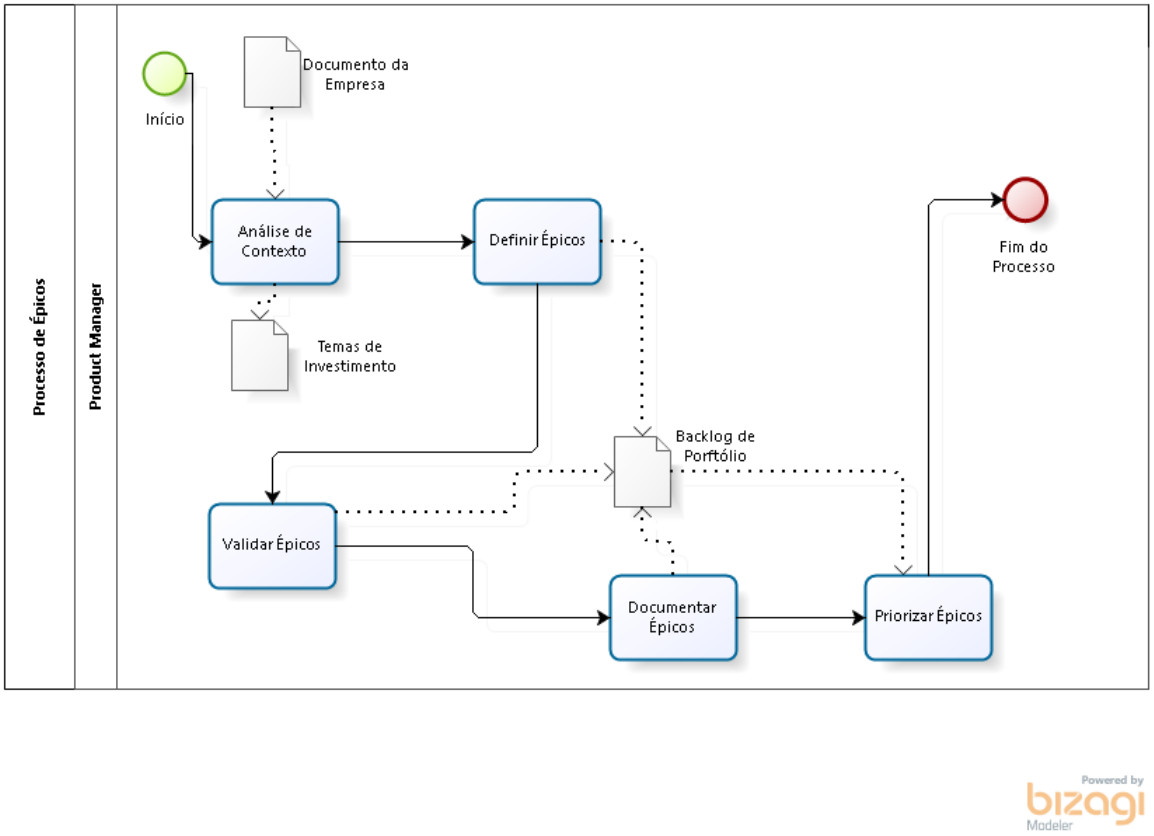

Figura 6 – Subprocesso de Épicos

Tabela 1 – Analisar Contexto

Descrição	Analisar o contexto e as abordagens da Azura Software.
Artefatos de entrada	Documentação da empresa.
Artefatos de saída	Temas de Investimentos.
Envolvidos	*Product Manager*

Tabela 2 – Definir Épicos

Descrição	Reunião com o cliente, com o uso da técnica de elicitação entrevista, para fazer o levantamento dos épicos no contexto do negócio.
Artefatos de entrada	Temas estratégicos.
Artefatos de saída	Backlog de Portfólio.
Envolvidos	*Product Manager*

Tabela 3 – Validar Épicos

Descrição	Validar, junto ao cliente, os épicos elicitados na etapa de definição dos épicos.
Artefatos de entrada	Backlog de Portfólio (Épicos elicitados).
Artefatos de saída	Backlog de Portfólio (Épicos validados).
Envolvidos	*Product Manager*

Tabela 4 – Documentar Épicos

Descrição	Documentar os épicos que foram validados e estão consistentes para fazer parte do Backlog de Portfólio.
Artefatos de entrada	Backlog de Portfólio (Épicos validados).
Artefatos de saída	Backlog de Portfólio (Épicos documentados).
Envolvidos	*Product Manager*

Tabela 5 – Priorizar Épicos

Descrição	Reunir e classificar épicos considerando seu nível de importância de acordo comas necessidades e interesses do cliente, definindo também as relações entre os mesmos.
Artefatos de entrada	Backlog de Portfólio (Épicos documentados).
Artefatos de saída	Backlog de Portfólio (Épicos priorizados).
Envolvidos	*Product Manager*

4.1.7.2 Programa

Descreve-se aqui as atividades relacionadas ao nível de Programa.

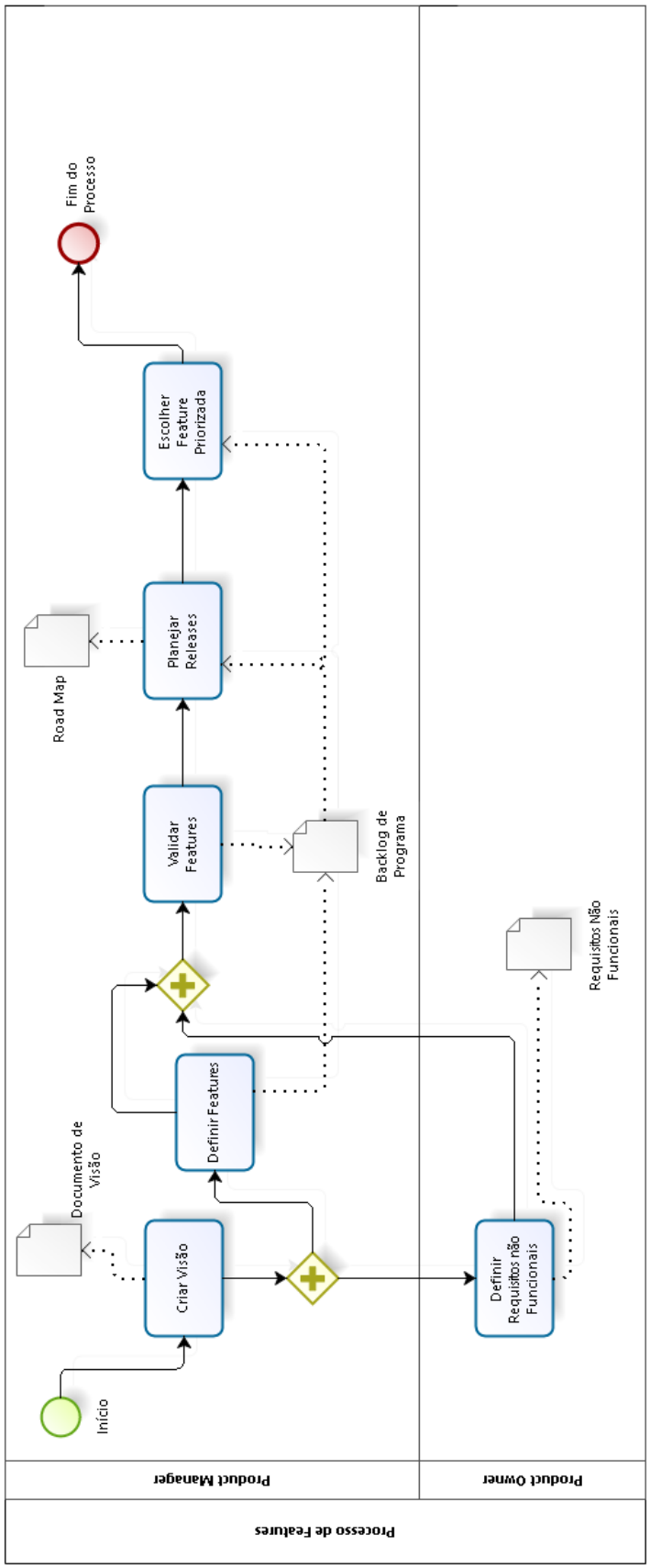

Figura 7 – Subprocesso de Features

Tabela 6 – Criar Visão

Descrição	Elaboração do documento de visão do projeto para obter uma visão futura da solução a ser implementada.
Artefatos de entrada	Backlog de Portfólio, Temas Estratégicos e Documento da empresa Azura.
Artefatos de saída	Documento de Visão
Envolvidos	*Product Manager*

Tabela 7 – Definir Requisitos não Funcionais

Descrição	Descrever os atributos do sistema como segurança, confiança, manutenibilidade, escalabilidade e usabilidade.
Artefatos de entrada	Documento de Visão
Artefatos de saída	Especificação Suplementar
Envolvidos	*Product Owner*

Tabela 8 – Definir Features

Descrição	Levantamento e definição das features de negócios necessárias para cada entrega (release).
Artefatos de entrada	Backlog de Portfólio
Artefatos de saída	Backlog de Programa (Features definidas)
Envolvidos	*Product Manager*

Tabela 9 – Validar Features

Descrição	Verificação e validação das features definidas, objetivando minimizar futuros problemas e riscos.
Artefatos de entrada	Backlog de Programa (Features definidas)
Artefatos de saída	Backlog de Programa (Features validadas)
Envolvidos	*Product Manager*

Tabela 10 – Planejar Releases

Descrição	Planejamento das entregas a partir dos épicos e features definidas.
Artefatos de entrada	Backlog de Programa (Features validadas)
Artefatos de saída	Road Map
Envolvidos	*Product Manager*

Tabela 11 – Escolher Feature Priorizada

Descrição	Selecionar as features de maior importância, considerando os relacionamentos entre as mesmas.
Artefatos de entrada	Backlog de Programa (Features validadas)
Artefatos de saída	Backlog de Programa (Features priorizadas)
Envolvidos	*Product Manager*

4.1.7.3 Time

Define-se aqui as atividades necessárias para realizar o processo à nível de Time, considerando a estrutura do SAFe.

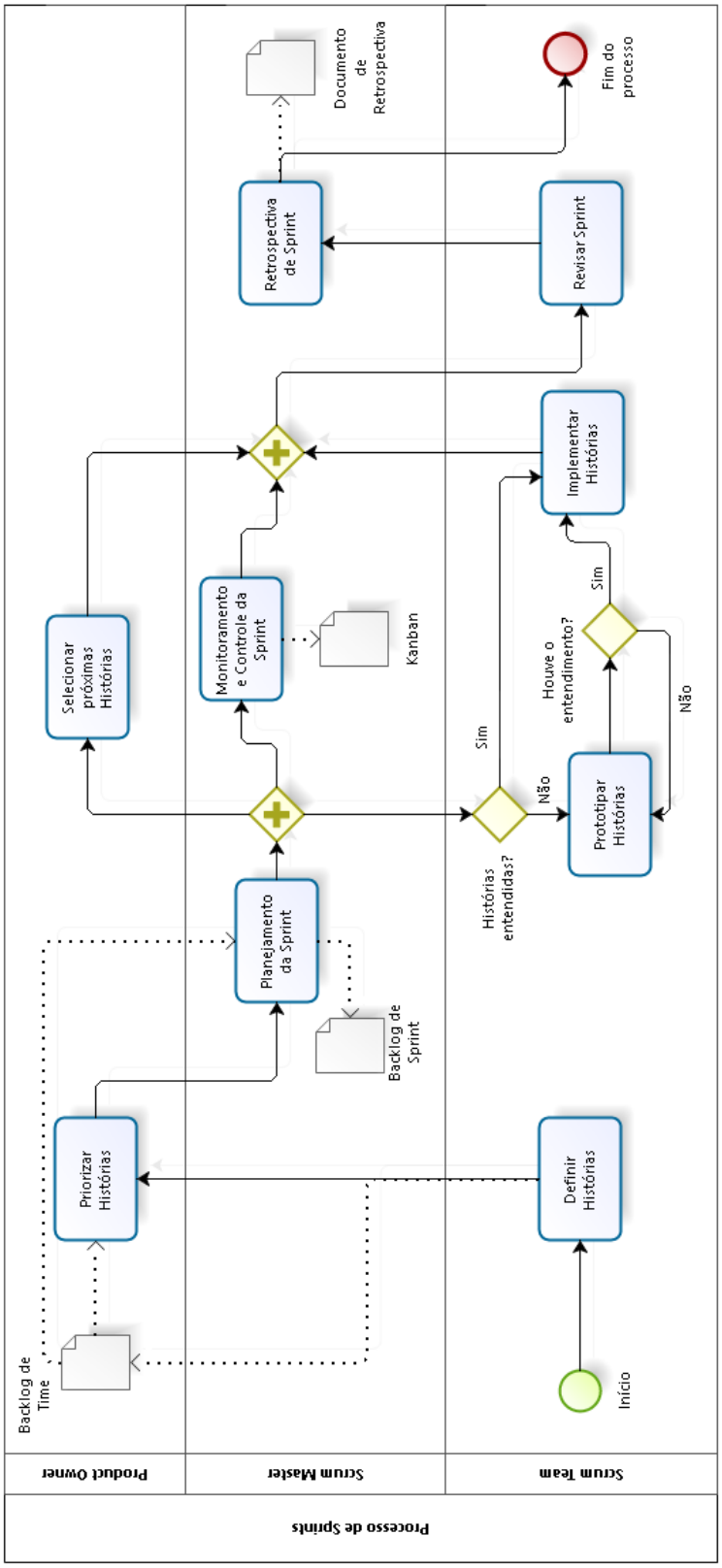

Figura 8 – Subprocesso de Sprints

Tabela 12 – Definir Histórias

Descrição	Realizar elicitação das histórias de usuário feitas pelo time e pelo product owner, definindo os critérios de aceitação das mesmas e as histórias habilitadoras.
Artefatos de entrada	Documento de Visão
Artefatos de saída	Backlog de Time (Histórias Elicitadas)
Envolvidos	*Scrum Team*

Tabela 13 – Priorizar Histórias

Descrição	Selecionar as histórias que serão implementadas na corrente sprint de acordo com a necessidade do PO.
Artefatos de entrada	Backlog de Time (Histórias Elicitadas)
Artefatos de saída	Backlog de Time (Histórias Priorizadas)
Envolvidos	*Product Owner*

Tabela 14 – Planejar Sprint

Descrição	Realizar o planejamento da sprint a ser realizada juntamente com o PO e Scrum Master, definindo as necessidades de cada sprint, o prazo de entrega e as histórias a serem implementadas.
Artefatos de entrada	Backlog de Time (Histórias Elicitadas)
Artefatos de saída	Backlog de Time (Histórias Priorizadas da Sprint)
Envolvidos	*Scrum Team*, *Scrum Master*, *Product Owner*

Tabela 15 – Selecionar Próximas Histórias

Descrição	Product Owner seleciona as histórias que devem ser implementadas na próxima Sprint, considerando as dependências e relacionamento entre as mesmas e suas prioridades.
Artefatos de entrada	Backlog de Time (Histórias Elicitadas)
Artefatos de saída	Backlog de Time (Histórias Priorizadas da Próxima Sprint)
Envolvidos	*Product Owner*

Tabela 16 – Monitoramento e Controle de Sprint

Descrição	Realizar o acompanhamento da Sprint juntamente com o time, com encontros diários de 15 minutos, para realizar a lista de atividades realizadas e atividades que serão realizadas até a próxima Daily Meeting (reuniões diárias).
Artefatos de entrada	Backlog de Time (Histórias Priorizadas)
Artefatos de saída	
Envolvidos	*Scrum Master* e *Scrum Team*

Tabela 17 – Prototipar Histórias

Descrição	Realiza-se a prototipagem das histórias priorizadas para a verificação do entendimento da mesma, considerando os critérios de aceitação e validação.
Artefatos de entrada	Backlog de Time (Histórias Priorizadas)
Artefatos de saída	Backlog de Time (Histórias Prototipadas)
Envolvidos	*Scrum Team*

Tabela 18 – Implementar Histórias

Descrição	Codificação das histórias prototipadas validadas como incremento do produto de *software*.
Artefatos de entrada	Backlog de Time (Histórias Prototipadas)
Artefatos de saída	Backlog de Time (Histórias Implementadas)
Envolvidos	*Scrum Team*

Tabela 19 – Revisar Sprint

Descrição	Reunião de revisão da Sprint, para verificar se os critérios foram atendidos, apresentando as histórias implementadas como incremento para o produto, aceitas de acordo com as especificações do Product Owner.
Artefatos de entrada	Backlog de Time (Histórias Implementadas)
Artefatos de saída	Backlog de Time (Histórias Aceitas)
Envolvidos	*Scrum Team* e *Scrum Master*

Tabela 20 – Realizar Retrospectiva da Sprint

Descrição	Inspecionar como foi a ultima sprint, considerando o relacionamento entre os envolvidos, processos, ferramentas, identificando os principais pontos que necessitam de melhorias, pontos que foram bons e propor melhorias.
Artefatos de entrada	Backlog de Time (Histórias Implementadas)
Artefatos de saída	Documento de Retrospectiva da Sprint
Envolvidos	*Scrum Team* e *Scrum Master*

4.1.7.4 Gerência de Requisitos

Descreve-se aqui as atividades de gerência de requisitos aplicadas no projeto, que consiste em utilizar as atividades do modelo de maturidade selecionado, que será falado no tópico 4.2.

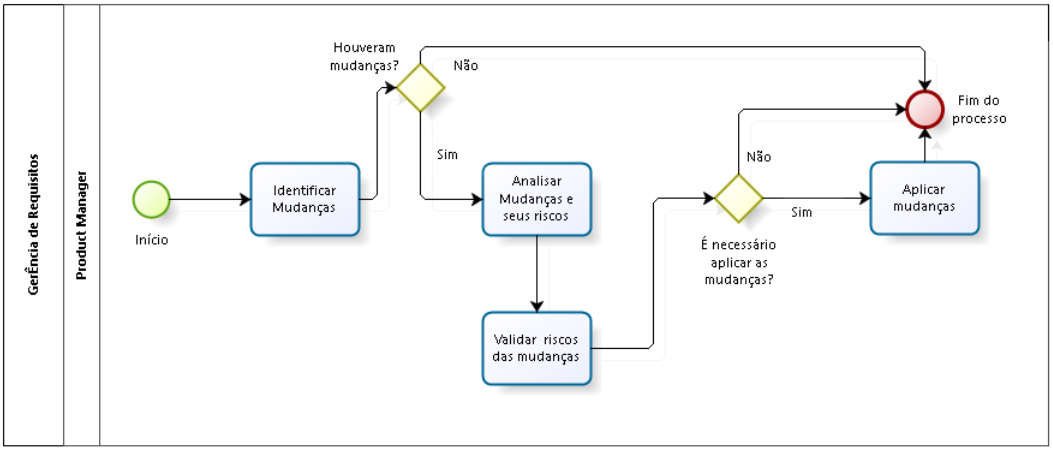

Figura 9 – Subprocesso de Gerência de Requisitos

Tabela 21 – Identificar Mudanças

Descrição	Estudo dos backlogs dos níveis do SAFe para a identificação de mudanças ocorridas pela má formulação de artefatos.
Artefatos de entrada	Backlogs (não gerenciados)
Artefatos de saída	Backlogs (gerenciados)
Envolvidos	*Product Manager*

Tabela 22 – Analisar Mudanças e seus riscos

Descrição	Avaliação das mudanças ocorridas, identificando a causa e as consequências da mesma no processo.
Artefatos de entrada	Backlogs (gerenciados)
Artefatos de saída	Backlogs (analisados)
Envolvidos	*Product Manager*

Tabela 23 – Validar riscos das mudanças

Descrição	Análise e validação das mudanças juntamente com o *Product Owner*, identificando se há a necessidade de aplicar as mesmas.
Artefatos de entrada	Backlogs (analisados)
Artefatos de saída	Backlogs (validados)
Envolvidos	*Product Manager*

Tabela 24 – Aplicar mudanças

Descrição	Aplicação das mudanças validadas e identificadas como sendo viáveis e necessárias para a melhoria do projeto.
Artefatos de entrada	Backlogs (validados)
Artefatos de saída	Backlogs (modificados)
Envolvidos	*Product Manager*

4.2 Modelo de Maturidade de Processos

Um Modelo de Maturidade de Processos fornece uma abordagem disciplinada para identificação dos processos críticos e definição de ações de melhoria alinhadas com os objetivos estratégicos do negócio e consistentes com o estágio de maturidade de seus processos (SIQUEIRA, 2011).

Dada o contexto da empresa Azura Software, foi definido o modelo MPS.BR, tendo em vista que o custo para sua implementação é menor que o CMMI e a empresa, por ter seu ambiente de trabalho e clientes no Brasil, não procura reconhecimento e internacional. Portanto, para avaliar o grau de maturidade do Processo de Engenharia de Requisitos desenvolvido, o Modelo de Maturidade de Processos escolhido o foi o MPS.BR.

4.2.1 Sobre o Modelo

Com o objetivo de impulsionar a melhoria da capacidade de desenvolvimento de *software* e serviços nas empresas brasileiras, o MPS.BR proporciona ganhos de competitividade para as indústrias e evolução do processo e da qualidade do *software*. (SOFTEX, 2011)

Os níveis de maturidade, que combina o processo de uma empresa e a sua capacidade, são definidas pelo Modelo de Referência MPS para *software* (MR-MPS-SW), permitindo avaliar, atribuir graus de efetividade na execução dos processos e estabelecer patamares na sua evolução. A capacidade do processo é definida como a habilidade do processo para alcançar os objetivos de negócio atuais ou futuros, sendo diretamente relacionada aos níveis de maturidade (SOFTEX, 2011).

O MR-MPSSW define sete níveis de maturidade: A (Em Otimização), B (Gerenciado Quantitativamente), C (Definido), D (Largamente Definido), E (Parcialmente Definido), F (Gerenciado) e G (Parcialmente Gerenciado), iniciando no nível G e progride até o nível A. Apesar da presença desses sete níveis, o processo de requisitos é estabelecido nos níveis G e D, tendo uma relevância efetiva para o alcance das melhorias no processo (SOFTEX, 2011).

4.2.2 Nível G – Parcialmente Gerenciado

Dentro do nível de maturidade G, um dos objetivos é gerenciar todos os requisitos do produto visando identificar inconsistências e incoerência entre eles, os planos do projeto e os produtos de trabalho do projeto. Visando esse objetivo, existe o Processo de Gerência de Requisitos (GRE) (SOFTEX, 2011).

Os resultados esperados do GRE são:

- **GRE 1 - O entendimento dos requisitos é obtido junto aos fornecedores de requisitos.** Neste resultado esperado, a atividade relacionada no processo é a "Definir épicos". As atividades relacionadas ao nível de Programa são "Definir requisitos não funcionais"e "Definir Features". Já no nível de Time, a atividade relacionada é "Definir histórias". O conjunto dessas atividades garantem que todos os stakeholders estejam cientes do escopo e dos objetivos do projeto.

- **GRE 2 - Os requisitos são avaliados com base em critérios objetivos e um comprometimento da equipe técnica com estes requisitos é obtido.** Neste resultado esperado, a atividade relacionada no nível de Portfólio é "Validar épicos". No nível de Programa, a atividade relacionada é "Validar features". No nível de Time, as atividades relacionada são "Planejar Sprint"e "Selecionar próximas histórias". Devido sua proximidade com a equipe técnica e participação constante no projeto, as atividades de avaliação e validação serão realizadas em conjunto com o cliente.

- **GRE 3 - A rastreabilidade bidirecional entre os requisitos e os produtos de trabalho é estabelecida e mantida.** Os requisitos elicitados conterão a informação de requisito de origem onde se tem a identificação bidirecional, para o produto gerado e para requsito origem, tanto da origem quanto dos requisitos derivados, o que define a rastreabilidade vertical.

- **GRE 4 - Revisões em planos e produtos de trabalho do projeto são realizadas visando identificar e corrigir inconsistências em relação aos requisitos.** Neste resultado esperado, no nível de Portfólio as atividades relacionadas são "Validar épicos"e "Revisar épico". No nível de Programa, a atividade relacionada é

"Revisar feature"e no nível de Time as atividades relacionadas são "Revisar sprint"e "Realizar retrospectiva da sprint". A partir dessas atividades é possível rever e corrigir possíveis imprecisões e equívocos de toda parte do projeto já implementada, ou até realizar mudanças de requisitos se for preciso.

- **GRE 5 - Mudanças nos requisitos são gerenciadas ao longo do projeto.** Mudanças de escopo podem acontecer em qualquer atividade incluída no processo, realizando alterações e reformulando épicos sempre que necessário, para melhor atender os requisitos do cliente. Apesar disso, há atividades especificas para identificar a necessidade de possíveis mudanças no escopo.

4.2.3 Nível D - Largamente Definido

O nível de maturidade D tem como propósito definir os requisitos gerais do cliente, englobando também em seu escopo os requisitos do produto e de seus componentes. Para isso, é utilizado o processo Desenvolvimento de Requisitos (DRE) (SOFTEX, 2011).

Os resultados esperados do DRE são:

- **DRE 1 -As necessidades, expectativas e restrições do cliente, tanto do produto quanto de suas interfaces, são identificadas.**
- **DRE 2 - Um conjunto definido de requisitos do cliente é especificado e priorizado a partir das necessidades, expectativas e restrições identificadas.** Neste objetivo, a atividade relacionada no nível de Portfólio é "Priorizar épico". No nível de Programa, a atividade relacionada é "Escolher feature priorizada"e no nível de Time, a atividade relacionada é "Priorizar histórias".
- **DRE 3 - Um conjunto de requisitos funcionais e não-funcionais, do produto e dos componentes do produto que descrevem a solução do problema a ser resolvido, é definido e mantido a partir dos requisitos do cliente.** O Backlog de Portfólio (épicos), Backlog de Programa (features) e o Backlog de Time (histórias) são gerados através de diversas atividades para atingir esse objetivo.
- **DRE 4 - Os requisitos funcionais e não-funcionais de cada componente do produto são refinados, elaborados e alocados.** As atividades relacionadas a esse resultado esperado são "Validar feature"e "Escolher feature priorizada"e no fim, são alocados no Backlog de Programa.
- **DRE 5 - Interfaces internas e externas do produto e de cada componente do produto são definidas.** Identifica as características funcionais e não funcionais do sistema, sem a necessidade de definição de atividades para a construção de interfaces do *software*.

- **DRE 6 - Conceitos operacionais e cenários são desenvolvidos.** Os artefatos definidos para o processo de engenharia de *software* da empresa Azura, não define criação de cenários específicos para descrição de requisitos.

- **DRE 7 - Os requisitos são analisados, usando critérios definidos, para balancear as necessidades dos interessados com as restrições existentes.** Neste resultado esperado, a atividade relacionada no nível de Portfólio é "Revisar épico"e no nível de Programa é relacionado a atividade "Revisar feature", onde é analisado a viabilidade de cada elemento antes de ir para o Visão.

- **DRE 8 - Os requisitos são validados.** Neste resultado esperado, a atividades relacionada no nível de Portfólio é "Validar épicos". No nível de Programa, a atividade relacionada é "Validar features"e no nível de Time, "Validar histórias". A partir disso, pode-se afirmar que a aceitação dos requisitos por parte do cliente será garantida nos três níveis do processo.

5 Elicitação dos Requisitos

A elicitação dos requisitos é a primeira etapa para o desenvolvimento de um software e é onde as técnicas de elicitação são aplicadas. A elicitação de requisitos é uma parte essencial da engenharia de requisitos, já que a partir dessa atividade são levantados e definidos os requisitos, tanto funcionais e não funcionais de um projeto, que por sua vez caracterizam o sistema a ser desenvolvido. Sendo assim, uma má elicitação dos requisitos pode causar trabalhos desnecessários, uma vez que se pode desenvolver funcionalidades que não vão trazer valor para o cliente, que com isso pode causar o desagrado com o software. Pular essa etapa e partir para um ponto onde os requisitos ainda não estejam completamente elicitados, levaria a um erro, pois o sistema a ser desenvolvido não estaria em conformidade com as necessidades reais do cliente, o que causaria gastos indevidos de recursos. (BELGAMO; MARTINS, 2000).

De forma a se alcançar o entendimento mais completo e claro do sistema em questão se utiliza as técnicas mais corretas. Para determinar tal coisa é utilizado como base para a decisão, além da metodologia utilizada, o perfil do cliente, o perfil da equipe e a interação da equipe com o cliente. Visando tal proposta foram selecionadas as seguintes técnicas:

- Entrevista;
- Brainstorm;
- Prototipagem.

5.1 Entrevista

Existem dois tipos de entrevista, aquelas que são planejadas, fechadas, onde o entrevistador procura as perguntas para um conjunto pré-definido de questões e as entrevistas abertas, que não possuem agenda, sendo assim o entrevistador esta livre para definir o curso da entrevista enquanto ela acontece, perguntando de modo aberto o que o cliente deseja (KOTONYA; SOMMERVILLE, 1998). Está técnica, de forma livre, se adéqua muito bem ao contexto, visto que o cliente é próximo dos integrantes da equipe e o vocabulário utilizado é similar. Desde modo, essa técnica será utilizada para elicitação dos requisitos de nível de abstração mais baixo, visto que há uma boa interação com o cliente.

5.2 Brainstorm

A técnica de Brainstorm se caracteriza quando um grupo de pessoas se junta, em um cenário e assunto simulados, para discutir ideias de forma livre. Os participantes devem estar confortáveis o bastante para discutir o tema sem o sentimento de intimidação. Sendo assim, nenhuma ideia é descartada, todas as ideias são boas ideias (LAUESEN, 2002). Com base nessa definição, escolheu-se essa técnica por ser simples e rápida, com uma boa taxa de resposta. Para aplicação dessa técnica será utilizado o problema em questão com o cenário de uma possível solução para ele. Um facilitador coordenará a todo o momento a aplicação dessa técnica, para que não se fuja do tema estabelecido.

5.3 Prototipagem

A fim de elicitar as preferências do cliente quanto a solução final, será utilizada a técnica de prototipagem. Essa técnica servirá de base para organizar os requisitos referentes tanto a utilização do sistema, quanto a conceitos de *design* e requisitos não funcionais. Além disso, essa técnica também pode ser utilizada para validação de alguns requisitos já levantados com a utilização de outras técnicas. Para a aplicação dessa técnica utilizaremos um protótipo de baixa fidelidade, onde se espera conseguir uma resposta rápida do cliente.

6 Gerência de Requisitos

Gerenciar um projeto de software é essencial para o sucesso do mesmo, visto que a metodologia abordada neste trabalho - Método Ágil - mostra que a adaptação às mudanças é essencial. Essa adaptação é melhorada a partir de um bom gerenciamento do projeto, que consiste em identificar, controlar e rastrear as mudanças. Considerando a ramificação do processo como um todo para a área de Engenharia de Requisitos, a gerência desses requisitos é fundamental para a construção de um projeto saudável e adaptável às mudanças que vão ocorrendo.

Em geral, gerênciar requisitos pode ser entendido melhor como gerenciar mudanças nos requisitos do sistema, e, para minimizar as dificuldades em adaptar o projeto à novos requisitos, os mesmos devem ser controlados e monitorados.

6.1 Atributos de um requisito

Atributos de requisito são características de um requisito que se dá a partir de um dado padrão. Para identificar os requisitos em todos os níveis serão utilizados os atributos descritos abaixo:

- Identificador;
- Prioridade;
- Estado;
- Estabilidade;
- Risco.

6.1.1 Identificador

Este atributo define qual é o nível de abstração do requisito e qual seu identificador único, o que permite identifica-lo e destingui-lo dos demais requisitos.

Ex: EP 1, FT 5, US 153.

6.1.2 Prioridade

Este atributo é utilizado para determinar a importância de um requisito, levando em conta o funcionamento ideal do sistema, e determinar sua prioridade na execução. Para esse atributo existe as possíveis opções:

- **Essencial**: Indica um requisito o qual o sistema não funciona sem sua implementação, portanto é um requisito critico para o projeto. Esses requisitos são de implementação imprescindível.

- **Importante**: Essa categoria é voltada para os requisitos que tem uma importância alta, mas não são essenciais para o funcionamento do sistema, porém sem sua implementação o sistema não responderá de forma satisfatória.

- **Desejável**: Requisitos desejáveis são os requisitos que o sistema pode funcionar de forma satisfatória sem eles, onde eles não se classificam como funcionalidades básicas do sistema. Tais requisitos podem ser postergados para versões futuras sem nenhuma perda para o funcionamento do programa.

6.1.3 Estado

Este atributo define o andamento da implementação do requisito no sistema. Para este atributo existem os possíveis campos, que seguem em fluxo:

- **Aberto**: Indica que um requisito foi criado, porém pode estar em faze de validação ou apenas em um estado de inércia dentro do projeto, esperando para ser planejado.

- **Planejado**: Este valor é dado para requisitos que foram criados e já estão separados para serem implementados em futuras iterações, com uma data planejada.

- **Em progresso**: Indica que a implementação do requisito está em andamento, porém ainda não foi concluído, independente da porcentagem da conclusão.

- **Em teste**: Indica que um requisito foi completo e está em fase de testes, onde é verificado se a implementação do requisito atende todas as necessidades e características descritas a priori nos critérios de aceitação de tal requisito.

- **Completo**: Este valor é dado apenas para os requisitos que foram implementados e sua implementação foi aceita nos testes.

6.1.4 Estabilidade

A estabilidade define qual é a volatilidade de um requisito, ou seja qual é a chance dele ser modificado no futuro, com o andamento do projeto. Para esse atributo foram definidos três valores:

- **Alta**: O valor de estabilidade alta é dado para os requisitos que se tem a certeza, ou baixa probabilidade, que não sofreram mudanças ao longo do projeto.

- **Média**: Indica que um requisito tem uma probabilidade média de sofrer mudanças no decorrer do projeto.

- **Baixa**: Os requisitos com estabilidade baixa são aqueles que tem uma probabilidade muito alta de sofrer mudanças futuras com o andamento do projeto.

6.1.5 Risco

Este atributo está relacionado a condição de um requisito gerar algum risco que seja capaz de prejudicar, de alguma forma, o andamento do projeto. Para este atributo foram definidos três valores:

- **Alto**: Esse valor indica que o requisito tem riscos de impacto significativos no projeto.

- **Médio**: Indica que o requisito tem riscos elevados, porém com níveis de impacto mais amenos.

- **Baixo**: Indica a ausência de riscos no projeto.

Na Tabela 25 é demonstrado a aplicação dos atributos de requisitos à historias de usuário.

Tabela 25 – Exemplificação do uso dos atributos de requisito

Identificador	**Prioridade**	**Estado**	**Estabilidade**	**Risco**
US 1	Alta	Aberto	Alta	Baixo
US 2	Média	Planejado	Média	Baixo
US 3	Alta	Em Progresso	Média	Alto

6.2 Rastreabilidade

Com as mudanças no projeto é necessário ter uma visão de origem e derivados, para assim ter o conhecimento de quais outros requisitos serão afetados por essa mudança. Para o correto gerenciamento dos requisitos deve-se atribuir uma via de mão dupla para as dependências dos requisitos, assim obtendo a visão de produtos gerados e requisitos precursores.

A estratégia definida para a rastreabilidade começa no nível de portfólio e percorre todos os demais níveis, incluindo todos os requisitos.

6.2.1 Rastreabilidade vertical

De forma a se obter uma concordância concisa entre os requisitos e a execução ótima do processo de engenharia de requisitos para o contexto deste projeto, foi abordada a seguinte rastreabilidade vertical, ilustrada visualmente na Figura 10. Onde há o relacionamento do requisito mais abstrato para o requisito mais definido.

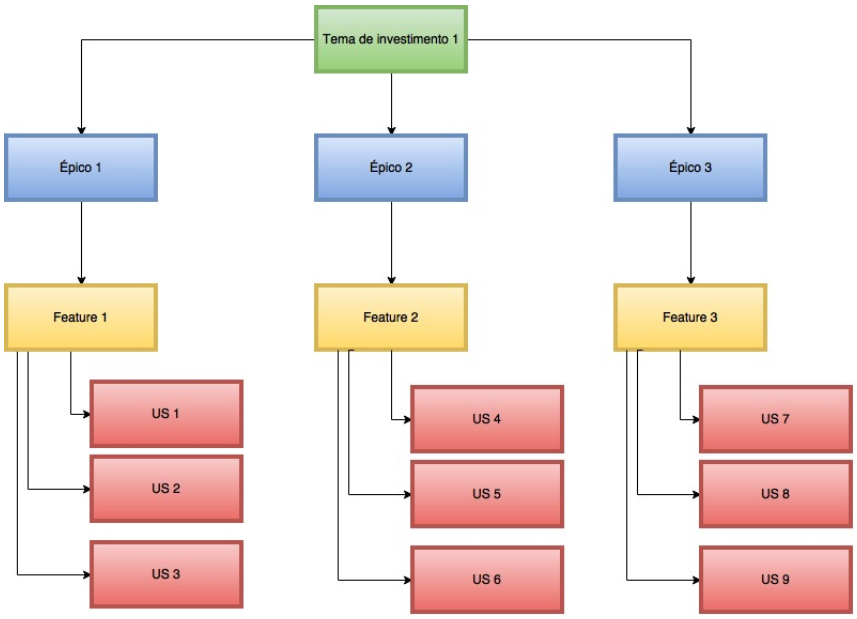

Figura 10 – Exemplificação da estratégia de rastreabilidade vertical

6.2.2 Rastreabilidade horizontal

Para ajudar a visualizar dependências e a garantir a completude do sistema será usado a rastreabilidade horizontal, que será mantida entre requisitos do mesmo nível de abstração. Desta forma pode-se ter uma visualização simples, porém completa do sistema e suas dependências internas. A Figura 11 exemplifica de forma visual como será feita a rastreabilidade horizontal, onde a seta pontilhada é a dependência entre o requisito origem para o alvo.

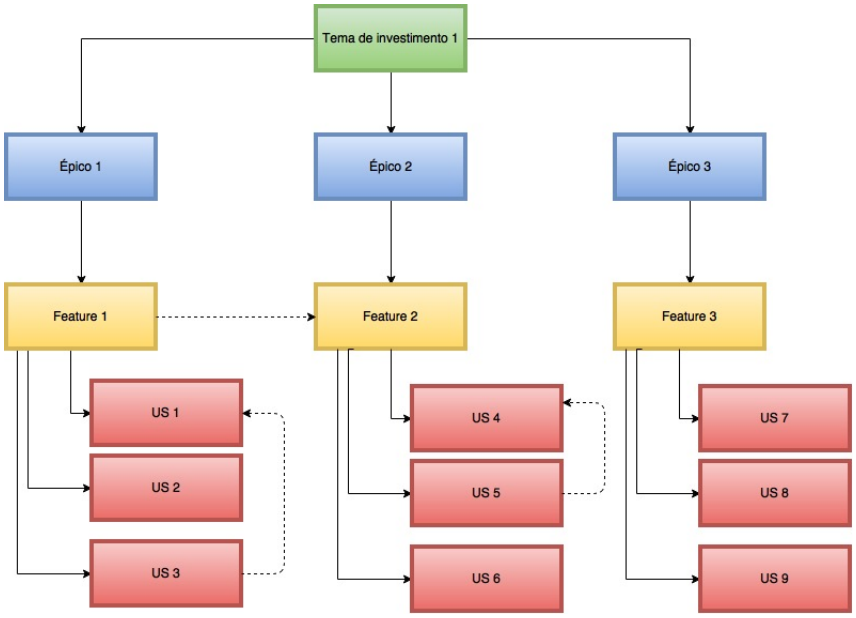

Figura 11 – Exemplificação da estratégia de rastreabilidade vertical e horizontal

7 Ferramenta de gerência de requisitos

"Aos requisitos estão associados os principais problemas do desenvolvimento de software. Requisitos que não refletem as reais necessidades dos usuários, incompletos e/ou inconsistentes, mudanças em requisitos já acordados e a dificuldade para conseguir um entendimento comum entre usuários e desenvolvedores são as principais dificuldades relatadas, provocando re-trabalho, atrasos no cronograma, custos ultrapassados e a insatisfação dos clientes e usuários de software" (BLASCHEK, 2002).

Muitos dos erros presente dentro dos projetos poderiam ser minimizados se as organizações utilizassem processos de Engenharia de Requisitos bem definido, controlado, medido e aprimorado (BLASCHEK, 2002). Com isso em mente, algumas empresas criaram ferramentas para apoiar e auxiliar os envolvidos nos processos de Engenharia de Requisitos, que possuem a capacidade de monitorar, gerenciar, manter e armazenar as mudanças dos requisitos (ANANIAS, 2009).

Neste tópico será presentado três ferramentas de Gerência de Requisitos e, dentre elas, será selecionada uma para ser utilizada no escopo do corrente trabalho.

7.1 Analise de ferramentas

7.1.1 Target Process

O *Target Process* é uma ferramenta bem mais visual e amigável aos desenvolvedores do que outras ferramentas no mercado. Ele oferece transparência e visibilidade através de toda a organização, e vários aspectos importantes e mais adaptativos ao processo, como:

- Projeto elaborado especificamente para ágil
- Customização
- Gestão de mudanças

É através dessas funcionalidades e de seus elementos visuais que a ferramenta cobre com êxito itens como rastreabilidade, seus componentes e o épico, item necessário no projeto em questão

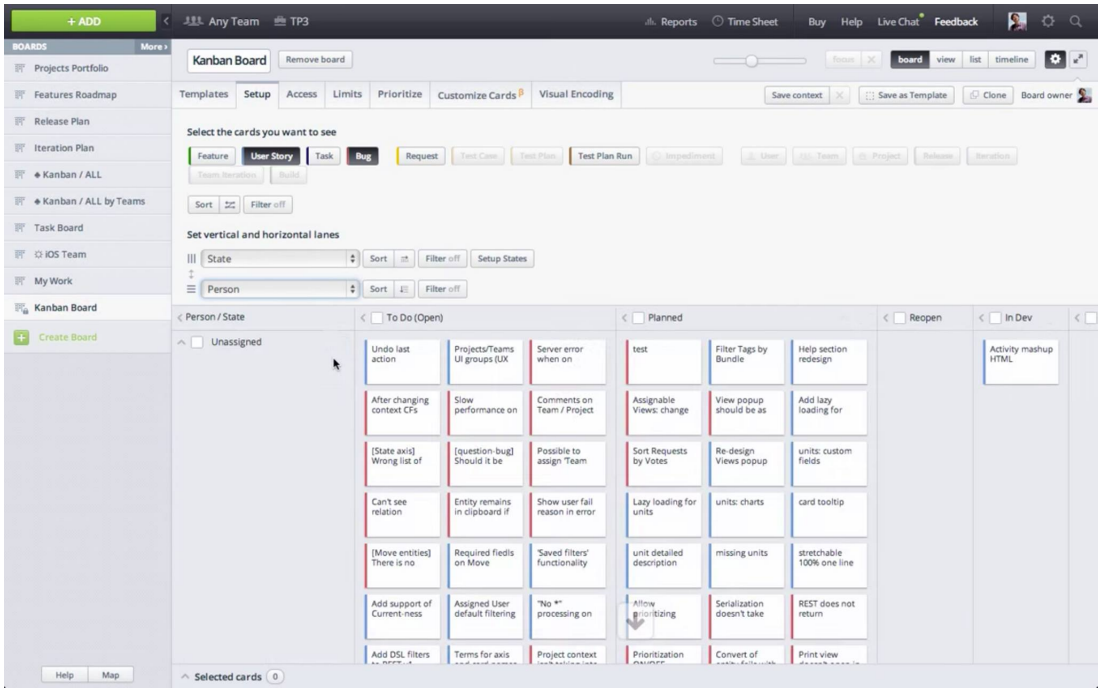

Figura 12 – Exemplo da ferramenta *Target Process*

7.1.2 OSRMT

O *Open Source Requirements Management Tool* (OSRMT) é uma ferramenta livre desenvolvida em Java. Ela foi projetada especificamente para apoiar o processo de gerência de requisitos. Essa ferramenta tem como característica principal permitir uma completa rastreabilidade do ciclo de vida de desenvolvimento de software em relação aos requisitos do projeto em questão, além de ser paralelamente uma gerência em relação aos componentes de produtos de trabalho. É de notório destaque várias funcionalidades disponíveis na ferramenta, como por exemplo:

- A origem e motivo da necessidade de cada requisito. E como foi citado, ela é paralela a todos os componentes. A ferramenta possibilita que essa origem e motivos de cada requisito estejam ligados ao registro de autor, registro de casos de uso, status e categorias.

- A rastreabilidade é um dos grande focos da ferramenta. Através de gráficos que identificam todas as dependências entre requisitos

- Geração de relatórios padronizados em formato *Portable Document Format* (PDF) e *Hyper Text Markup Language* (HTML);

- Definição e organização de artefatos e entrada de dados

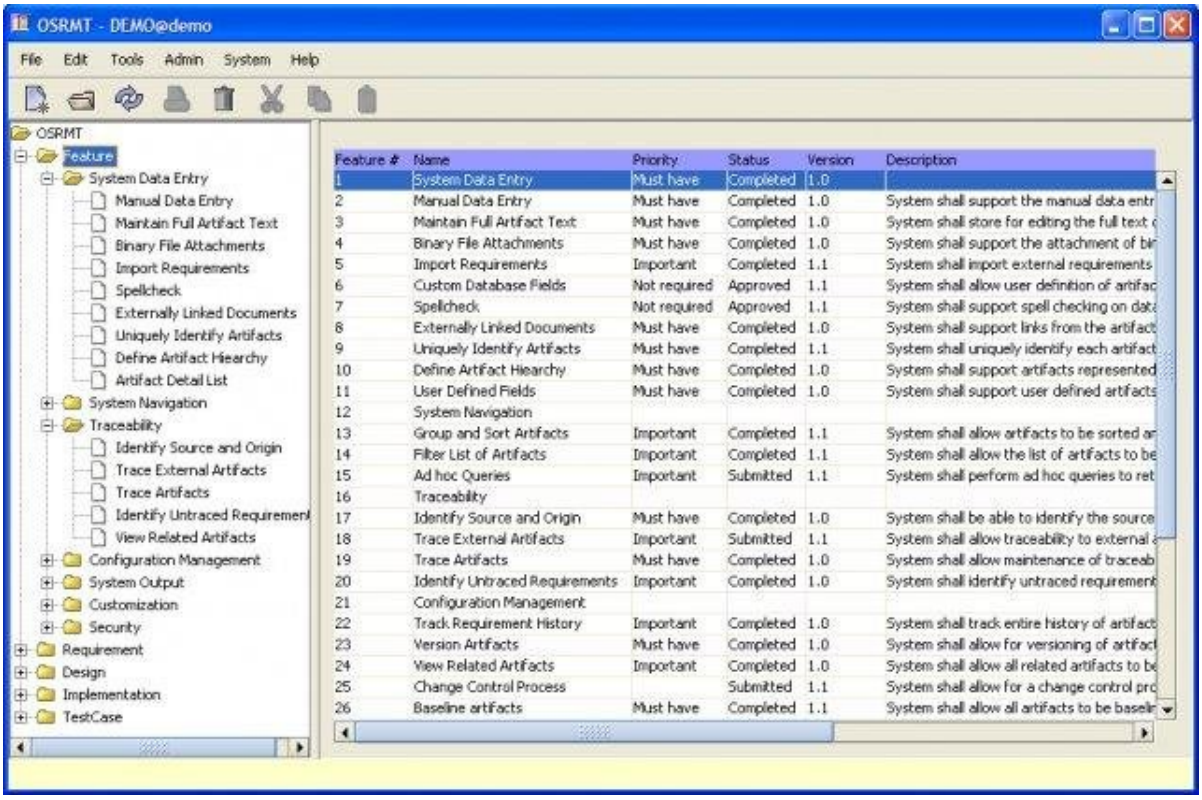

Figura 13 – Exemplo da Ferramenta OSRMT

7.1.3 SIGERAR

O SIGERAR é ferramenta é operada via interface Web e foi desenvolvida em linguagem *Java* com *Java Server Pages* (JSP) e com o Sistema Gerenciador de Banco de Dados (SGBD).

Uma das principais características da ferramenta é o tratamento da rastreabilidade dos requisitos no processo de alteração, onde a mesma permite análise e atribuição de valores de risco, importância, impacto, prioridade e custo a todos requisitos envolvidos (origem e dependentes), de forma a produzir informações ao Gerente do Projeto

Outro diferencial, é que o SIGERAR permite que cada organização possa customizá-los, de acordo com sua própria cultura ou características, de forma a obter maior adequação.

Figura 14 – Exemplo em que a ferramenta destaca os envolvidos

7.2 Comparativo

7.2.1 Os atributos

Diversos atributos foram analisados durante a escolha de uma ferramenta de gerenciamento de requisitos. Dentre eles, foram escolhidos os três maiores focos para análise detalhada:

- Usabilidade
- Rastreabilidade
- Gestão de mudanças

7.2.2 Tabela comparativa

Foi pontuado cada um dos atributos correspondes as ferramentas analisadas em uma nota geral de 1 a 5, onde 5 é uma nota muita boa, 3 uma nota razoável, e 1 uma nota muito ruim.

Tabela 26 – Tabela de comparação das ferramentas.

Critérios de Avaliação				
Ferramenta	Usabilidade	Rastreabilidade	Gestão de Mudanças	Total Avalidado
Target Process	5	5	4	14
OSRMT	3	4	4	11
SIGERAR	2	3	4	9

7.3 Escolha da ferramenta

Após análise dos atributos citados em cada uma das ferramentas, optou-se pelo *Target Process*.

Essa escolha foi afetada principalmente pelo fator de consistência, onde todos os atributos analisados receberam destaque, e o projeto em questão necessita da ferramenta consistente e completa com seus objetivos

8 Planejamento do projeto

8.1 Cronograma

O cronograma é um instrumento de planejamento que informa o tempo que será gasto no desenvolvimento de um projeto, explicitando as atividades, metas e prazos a serem cumpridos. Nele é possível enxergar de forma gráfica ou através de tabelas o andamento de determinada atividade e quem é o responsável por ela, aumentando a praticidade e disposição de informações referentes ao projeto.

Um cronograma bem planejado facilita o entendimento das atividades e gera proatividade na equipe, diminuindo o trabalho de gerentes e responsáveis por monitorar atividades dentro de uma empresa.

Os atributos presentes no cronograma do projeto são:

- **Nome:** Nome da atividade a ser desempenhada.
- **Duração:** Quantidade de dias que irá durar determinada atividade.
- **Início:** Data início da atividade.
- **Fim:** Data fim da atividade.
- **Completo:** Percentual referente ao desenvolvimento da atividade
- **Recursos:** Responsável pela atividade.

8.2 Fase 1

O cronograma presente na Figura 15 ilustra de forma fiel as atividades que serão efetuadas visando a construção deste relatório e do processo de Engenharia de Requisitos.

	Nome	Duração	% Completo	Início	Fim	Recursos
1	**Trabalho 1**	24d?	100%	**25/08/2016**	**27/09/2016**	**Equipe**
2	**Contextualização da empresa**	3d?	100%	**25/08/2016**	**29/08/2016**	
3	Definir reuniões com cliente	2d?	100%	25/08/2016	26/08/2016	Equipe
4	Entender contexto e problema	1d?	100%	26/08/2016	26/08/2016	Equipe
5	Analisar perfil do cliente e problemática	2d?	100%	26/08/2016	29/08/2016	Equipe
6	Definir Cronograma	1d?	100%	26/08/2016	26/08/2016	Equipe
7	Definir técnicas de elicitação	2d?	100%	26/08/2016	29/08/2016	Equipe
8	Definir metodologia	2d?	100%	29/08/2016	30/08/2016	Equipe
9	Definir papéis da equipe	3d?	100%	30/08/2016	01/09/2016	Equipe
10	**Semana 1**	6d?	100%	**01/09/2016**	**08/09/2016**	
11	Documentar template do relatório	1d?	100%	01/09/2016	01/09/2016	Danilo Barros
12	Documentar contexto e problema do cliente	1.5d?	100%	02/09/2016	05/09/2016	Vitor Bertulucci Borges,Thiago Moreira
13	Definir ferramenta de gerenciamento	1d?	100%	02/09/2016	02/09/2016	Equipe
14	Definir processo de ER	1d?	100%	02/09/2016	02/09/2016	Equipe
15	Modelar processo	2d?	100%	02/09/2016	05/09/2016	Vitor Bertulucci Borges,Danilo Barros
16	Documentar justificativa de escolha da metodologia	2.5d?	100%	05/09/2016	07/09/2016	Vitor Bertulucci Borges,Thiago Moreira
17	Documentar metodologia usada	0.5d?	100%	07/09/2016	07/09/2016	Vitor Bertulucci Borges,Thiago Moreira
18	**Reunião com cliente**	1d?	100%	**05/09/2016**	**05/09/2016**	**Equipe**
19	Validar processo junto ao cliente	1d?	100%	05/09/2016	05/09/2016	Equipe
20	Refinar processo validado	2d?	100%	05/09/2016	06/09/2016	Danilo Barros
21	Reunião do time	1d?	100%	06/09/2016	06/09/2016	Equipe
22	Documentar processo validado	2d?	100%	06/09/2016	07/09/2016	Equipe
23	Ponto de controle 1	1d?	100%	08/09/2016	08/09/2016	Vitor Bertulucci Borges,Thiago Moreira
24	**Semana 2**	8d?	100%	**06/09/2016**	**15/09/2016**	
25	Pesquisar ferramentas de gerenciamento de requisitos	2d?	100%	07/09/2016	08/09/2016	Equipe
26	Justificar escolha de rastreabilidade e atributos	6d?	100%	06/09/2016	13/09/2016	Vitor Bertulucci Borges
27	Documentar rastreabilidade e atributos dos requisitos	3d?	100%	09/09/2016	13/09/2016	Thiago Moreira,Danilo Barros
28	Documentar escolha de ferramenta	3d?	100%	09/09/2016	13/09/2016	Henrique Lopes
29	Estudar CMMI	1d?	100%	09/09/2016	09/09/2016	Thiago Moreira,Danilo Barros
30	Estudar MPS-Br	0.5d?	100%	09/09/2016	09/09/2016	Vitor Bertulucci Borges,Henrique Lopes
31	Definir Modelo de maturidade	1d?	100%	12/09/2016	12/09/2016	Equipe
32	**Reunião com cliente**	1d?	100%	**12/09/2016**	**12/09/2016**	**Equipe**
33	Validar atividades	1d?	100%	12/09/2016	12/09/2016	Equipe
34	Reunião do time	0.5d?	100%	15/09/2016	15/09/2016	
35	Documentar relatório	1d?	100%	15/09/2016	15/09/2016	Equipe
36	**Semana 3**	7d?	100%	**19/09/2016**	**27/09/2016**	
37	Documentar relatório final	6d?	100%	19/09/2016	26/09/2016	Equipe
38	Elaborar cronograma Trabalho 2	1d?	100%	26/09/2016	26/09/2016	Equipe
39	Revisar relatório final	2d?	100%	26/09/2016	27/09/2016	Equipe

Figura 15 – Cronograma referente a Fase 1 do projeto

8.3 Fase 2

O cronograma presente na Figura 16 define as atividades previstas que serão realizadas na segunda fase do projeto. Devido o projeto ter uma Abordagem Ágil, pode haver divergências entre as datas por ser um processo muito dinâmico e variável, impossibilitando um detalhamento mais afundo.

		Nome	Duração	% Completo	Início	Fim	Recursos
40		**Trabalho 2**	34d?	0%	**03/10/2016**	**17/11/2016**	
41		**Portfólio**	12d?	0%	**03/10/2016**	**18/10/2016**	
42		Análise de Contexto	2d?	0%	03/10/2016	04/10/2016	Equipe
43		Criar Temas de Investimento	1d?	0%	04/10/2016	04/10/2016	Equipe
44		Criar Backlog de Portfólio	1d?	0%	04/10/2016	04/10/2016	Henrique Lopes
45		Definir Épicos	4d?	0%	05/10/2016	10/10/2016	Thiago Moreira
46		Validar Épicos	2d?	0%	11/10/2016	12/10/2016	Vitor Bertulucci Borges
47		Documentar Épicos	2d?	0%	13/10/2016	14/10/2016	Danilo Barros
48		Priorizar Épicos	1d?	0%	14/10/2016	14/10/2016	Equipe
49		Registrar Épicos na ferramenta	2d?	0%	17/10/2016	18/10/2016	Vitor Bertulucci Borges
50		**Programa**	9d?	0%	**19/10/2016**	**31/10/2016**	
51		Criar Visão	3d?	0%	19/10/2016	21/10/2016	Equipe
52		Definir Requisitos não Funcionais	1d?	0%	19/10/2016	19/10/2016	Equipe
53		Documentar Requisitos não Funcionais	1d?	0%	19/10/2016	19/10/2016	Henrique Lopes
54		Criar Backlog de Programa	1d?	0%	20/10/2016	20/10/2016	Danilo Barros
55		Definir Features	2d?	0%	20/10/2016	21/10/2016	Thiago Moreira
56		Validar Features	2d?	0%	24/10/2016	25/10/2016	Danilo Barros
57		Planejar Releases	2d?	0%	26/10/2016	27/10/2016	Equipe
58		Criar Roadmap	1d?	0%	27/10/2016	27/10/2016	Vitor Bertulucci Borges
59		Registrar Features na Ferramenta	1d?	0%	28/10/2016	28/10/2016	Vitor Bertulucci Borges
60		Escolher Feature Priorizada	1d?	0%	31/10/2016	31/10/2016	Danilo Barros
61		Documentar no relatório resultados preliminares	3d?	0%	21/10/2016	25/10/2016	Equipe
62		Ponto de controle 2	1d?	0%	25/10/2016	25/10/2016	Equipe
63		**Time**	11d?	0%	**31/10/2016**	**14/11/2016**	
64		Definir Histórias (elicitar)	2d?	0%	31/10/2016	01/11/2016	Equipe
65		Criar Backlog de Time	1d?	0%	31/10/2016	31/10/2016	Henrique Lopes
66		**Sprint 01**	6d?	0%	**31/10/2016**	**07/11/2016**	
67		Priorizar Histórias	1d?	0%	01/11/2016	01/11/2016	Equipe
68		Planejamento de Sprint	2d?	0%	01/11/2016	02/11/2016	Equipe
69		Criar Backlog de Sprint	1d?	0%	02/11/2016	02/11/2016	Henrique Lopes
70		Criar Kanban	1d?	0%	02/11/2016	02/11/2016	Danilo Barros
71		Priorizar próximas histórias	3d?	0%	03/11/2016	07/11/2016	Vitor Bertulucci Borges
72		Monitoramento e Controle da Sprint	6d?	0%	31/10/2016	07/11/2016	Danilo Barros
73		**Execução**	4d?	0%	**02/11/2016**	**07/11/2016**	
74		Prototipar histórias	1d?	0%	02/11/2016	02/11/2016	Equipe
75		Implementar histórias	4d?	0%	02/11/2016	07/11/2016	Equipe
76		Revisar Sprint	1d?	0%	07/11/2016	07/11/2016	Equipe
77		Retrospectiva da Sprint	1d?	0%	07/11/2016	07/11/2016	Equipe
78		Documentar retrospectiva	1d?	0%	07/11/2016	07/11/2016	Danilo Barros
79		**Gerenciamento de Requisitos**	6d?	0%	**31/10/2016**	**07/11/2016**	
80		Identificar mudanças no projeto	5d?	0%	31/10/2016	04/11/2016	Vitor Bertulucci Borges
81		Analisar riscos	1d?	0%	07/11/2016	07/11/2016	Vitor Bertulucci Borges
82		Validar mudanças	1d?	0%	07/11/2016	07/11/2016	Vitor Bertulucci Borges
83		Aplicar Mudanças	1d?	0%	07/11/2016	07/11/2016	Vitor Bertulucci Borges
84		**Sprint 02**	6d?	0%	**07/11/2016**	**14/11/2016**	
85		Definir histórias	1d?	0%	07/11/2016	07/11/2016	Equipe
86		Priorizar Histórias	1d?	0%	08/11/2016	08/11/2016	Equipe
87		Planejamento de Sprint	1d?	0%	08/11/2016	08/11/2016	Equipe
88		Criar Backlog de Sprint	1d?	0%	08/11/2016	08/11/2016	Danilo Barros
89		Criar Kanban	1d?	0%	08/11/2016	08/11/2016	Thiago Moreira
90		Priorizar próximas histórias	5d?	0%	08/11/2016	14/11/2016	Henrique Lopes
91		Monitoramento e Controle da Sprint	6d?	0%	07/11/2016	14/11/2016	Thiago Moreira
92		**Execução**	5d?	0%	**08/11/2016**	**14/11/2016**	
93		Prototipar histórias	1d?	0%	08/11/2016	08/11/2016	Equipe
94		Implementar histórias	4d?	0%	09/11/2016	14/11/2016	Equipe
95		Revisar Sprint	1d?	0%	14/11/2016	14/11/2016	Equipe
96		Retrospectiva da Sprint	1d?	0%	14/11/2016	14/11/2016	Equipe
97		Documentar retrospectiva	1d?	0%	14/11/2016	14/11/2016	Thiago Moreira
98		**Gerenciamento de Requisitos**	6d?	0%	**07/11/2016**	**14/11/2016**	
99		Identificar mudanças no projeto	6d?	0%	07/11/2016	14/11/2016	Henrique Lopes
100		Analisar riscos	1d?	0%	14/11/2016	14/11/2016	Henrique Lopes
101		Validar mudanças	1d?	0%	14/11/2016	14/11/2016	Henrique Lopes
102		Aplicar Mudanças	1d?	0%	14/11/2016	14/11/2016	Henrique Lopes
103		Construir documentação final	4d?	0%	11/11/2016	16/11/2016	Equipe
104		Construir apresentação	1d?	0%	15/11/2016	15/11/2016	Equipe
105		Apresentar	1d?	0%	17/11/2016	17/11/2016	Equipe

Figura 16 – Cronograma referente a Fase 2 do projeto

9 Considerações finais

Atualmente, muitos dos projetos de software são acompanhados de diversas falhas e erros, tanto de comunicação quanto de entendimento dos requisitos e uma boa parte desses erros derivam de uma má gerência dos mesmos. No presente trabalho, pôde observar a importância da Engenharia de Requisitos dentro de um processo de construção de software.

Um dos pontos mais importantes na construção de um software é conseguir realizar o gerenciamento e acompanhamento dos requisitos no decorrer de todo o processo, visto que as principais falhas que ocorrem dentro de projetos de software se dá pela má formulação e entendimento dos requisitos do mesmo.

Nesta Relatório 1 conseguiu-se absorver o essencial dos conceitos por trás de uma Engenharia de Requisitos e a sua importância na construção de um processo de Engenharia de Software de qualidade. No decorrer da disciplina, será realizado o Relatório 2, que consiste na aplicação dos conceitos aqui apresentados sobre o contexto da empresa Azura Software.

Referências

AMBLER, S. W. *Agile Modeling: Effective Practices for eXtreme Programming and the Unified Process*. [S.l.: s.n.], 2002. v. 1. Citado na página 22.

ANANIAS, R. R. T. *Estudo comparativo entre ferramentas de Gerência de Requisitos*. 2009. Disponível em: <http://www.cin.ufpe.br/~tg/2009-2/rrta.pdf>. Citado na página 51.

AZURA. Regimento interno. 2016. Citado na página 16.

BELGAMO, A.; MARTINS, L. E. G. Estudo comparativo sobre as técnicas de elicitação de requisitos do software. In: *XX Congresso Brasileiro da Sociedade Brasileira de Computação (SBC), Curitiba–Paraná*. [S.l.: s.n.], 2000. Citado na página 43.

BLASCHEK, J. R. *Gerência de Requisitos: O principal problema dos projetos de software*. 2002. Disponível em: <http://www.bfpug.com.br/islig-rio/Downloads/Ger%C3%AAncia%20de%20Requisitos-o%20Principal%20Problema%20dos%20Projetos%20de%20SW.pdf>. Citado na página 51.

BOEHM, B.; TURNER, R. *Balancing Agile and Discipline: A guide of the perplexed*. [S.l.: s.n.], 2009. Citado 3 vezes nas páginas 19, 20 e 21.

IBM. *IBM Knowledge Center Documento de Visão*. 2015. Disponível em: <http://www.ibm.com/support/knowledgecenter/pt-br/SSCP65_4.0.6/com.ibm.rational.rrm.help.doc/topics/r_vision_doc.html>. Citado na página 26.

IEEE. *Guide to the Software Engineering Body of Knowledge*. [S.l.: s.n.], 2014. v. 3. Citado na página 17.

KOTONYA, G.; SOMMERVILLE, I. *Requirements engineering: processes and techniques*. [S.l.]: Wiley Publishing, 1998. Citado na página 43.

KRUCHTEN, P. *Introdução ao RUP Rational Unified Process*. [S.l.: s.n.], 2000. v. 2. Citado 2 vezes nas páginas 21 e 22.

LAUESEN, S. *Software requirements: styles and techniques*. [S.l.]: Pearson Education, 2002. Citado na página 44.

LEFFINGWELL, D. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise*. [S.l.: s.n.], 2011. Citado 2 vezes nas páginas 25 e 26.

LEFFINGWELL, D. Welcome to saled agile framework v4.0! 2016. Citado na página 19.

LIBARDI, P.; BARBOSA, V. Métodos Ágeis. 2010. Citado na página 25.

PRESSMAN, R. *Software Engineering*. [S.l.]: Mc Grall Hill Education, 2016. Citado 3 vezes nas páginas 17, 19 e 20.

SAFE. *Scaled Agile Framework - SAFe for Lean Software and System Egineering*. 2015. Disponível em: <http://scaledagileframework.com/>. Citado 5 vezes nas páginas 23, 24, 25, 26 e 27.

SCHWABER, K. *Agile Project Management with Scrum*. [S.l.: s.n.], 2004. v. 1. Citado 2 vezes nas páginas 24 e 25.

SIQUEIRA, J. O modelo de maturidade de processos: como maximizar o retorno dos investimentos em melhoria da qualidade e produtividade. 2011. Citado na página 38.

SOFTEX. Guia geral mps de software. 2011. Citado 3 vezes nas páginas 38, 39 e 40.

SOMMERVILLE, I. *Software Engineering*. [S.l.]: Pearson Education, 2007. Citado 2 vezes nas páginas 17 e 21.