Projeto Final da Disciplina de Requisitos de Software

Ferramenta de Gerência de Requisitos

Requisitos de Software

Rafael Fazzolino Pinto Barbosa - 11/0136942
Thiago Ramires Kairala - 12/0042916
Eduardo Brasil Martins - 11/0115104
Bruno Contessotto Bragança - 09/0107853

Brasília, DF - 2014

Histórico de Alterações

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

O desenvolvimento de *software* passa por inúmeras fases até que seja concluído e entregue ao cliente, uma delas, e provavelmente a mais importante, é a Engenharia de Requisitos, onde devemos entender o problema do usuário, compreender suas necessidades e apresentá-lo a uma solução. Nesta fase, negociações serão feitas, tanto sobre funcionalidades do sistema quanto custos, tempo para conclusão e restrições de qualquer tipo.

O resultado desta fase é uma documentação robusta, principalmente ao utilizar metodologias tradicionais de desenvolvimento. Nesta documentação encontram-se as funcionalidades do *software*, suas características e restrições, podendo abranger todo o *software* ou apenas uma primeira etapa de desenvolvimento, como é feito em metodologias ágeis.

A tarefa de construir e manter a documentação necessária em um projeto de *software* possui diversos problemas relacionados a diversas áreas diferentes, como por exemplo a gerência, organização, classificação e rastreabilidade dos requisitos. Surge assim a necessidade da utilização de ferramentas que possam amenizar as dificuldades encontradas.

1 .1 Propósito

Ao ler este documento, todos os *Stakeholders* deverão compreender todo o contexto de negócio, os objetivos e escopo do projeto, assim como, entender o problema que deverá ser resolvido, quais necessidades do cliente deverão ser analisadas e quais serão as funcionalidades do sistema.

1 .2 Escopo

Este documento abrange o contexto do desenvolvimento de *software* voltado para a Engenharia de Requisitos, desde a elicitação à gerência de requisitos, e tem como objetivo levar o entendimento do projeto a qualquer leitor, desde leigos até especialistas na área. Encontra-se neste documento, o problema de negócio do cliente, suas reais necessidades, características e funcionalidades do sistema que foram possíveis mapear.

Dessa forma, a partir deste documento, pode-se obter conhecimento total sobre o projeto de desenvolvimento da R.A.D.I.T., desde a metodologia utilizada até a forma de implementação do sistema.

1 .3 Definições, acrônimos e abreviações

Durante o processo de elicitação e gerenciamento de requisitos é necessário que todos os envolvidos possam se comunicar sem que existam falhas de entendimento, para isso, foi desenvolvido um sumário contendo nomes que serão utilizados no processo, assim como suas definições.

- *Stakeholders*

  Todas as partes envolvidas no contexto do sistema, desde o cliente e seus funcionarios até a equipe de desenvolvimento do sistema. Todos os interessados na solução de *software* são considerados *Stakeholders* do sistema [Sommerville et al. 2003].

- *Requisitos*

  Engloba tudo que o *software* deve possuir para solucionar o problema em questão, desde funcionalidades do sistema até características que o *software* deve possuir.

- *Requisitos Funcionais*

  São chamados de requisitos funcionais todos aqueles que apresentam as funcionalidades do sistema [Sommerville et al. 20

- *Requisitos não Funcionais*

  São chamados requisitos não funcionais todos aqueles que apresentam as características do sistema, incluindo compatibilidade, o tempo de resposta ou qualquer outra exigência que não inclua funcionalidades [Sommerville et al. 2003].

- *Engenharia de Requisitos*

  Engenharia de Requisitos é um conceito que engloba todo um contexto de desenvolvimento de *software* que envolve elicitação de requisitos, negociação, verificação e validação, e documentação e gerência de requisitos para o desenvolvimento de um sistema computacional. O uso da palavra *Engenharia* garante que técnicas sistematicas serão utilizadas para que os requisitos sejam completos, corretos e consistentes [Espindola, Majdenbaum e Audy 2004].

- *Fishbone*

  Consiste em uma técnica utilizada para o reconhecimento do macro problema do cliente. A utilização desta técnica garante uma facilidade maior para entender onde a solução deve atuar.

- *Framework do problema*

  Consiste em uma técnica para organizar e auxiliar o entendimento do problema, apresentar os stakeholders afetados pelo problema, o impacto que o problema gera para o cliente e uma possivel soluçao bem sucedida. A utilização do framework garante maior facilidade no entendimento do contexto do cliente.

- *Framework de Necessidades*

  Consiste em uma técnica para oganizar uma tabela identificando Necessidade, Problema, Solução atual e Solução Proposta. A utilização do framework de necessidade garante um melhor entendimento da necessidade do cliente.

- *WorkShop*

  Workshop é uma técnica no qual os partipantes discutem um problema em comum onde são aplicadas técnicas que ajudam em uma melhor identificação das necessidades do cliente e ajudam a melhorar o rendimento das reuniões.

- *Brainstorming*

  Brainstorming é uma técnica que consiste em uma dinâmica de grupo para recolher ideias a respeito de um determinado assunto e para a resolução de problemas.

- *Casos de Uso*

  Caso de uso define uma sequência de ações que produz um resultado de valor observável. Os casos de uso fornecem estrutura para expressar requisitos funcionais no contexto dos processos de negócio e de sistema.

- *Sprint*

  Representa o espaço de tempo no qual deverão ser realizadas atividades previamente estabelecidas para a resolução de um problema. [Beck 2000].

- *Release*

  São entregas de código funcional, as quais são feitas por etapa, entregando pequenas partes do *software* de tempos em tempos. [Beck 2000].

- *Product Owner (PO)*

  É o responsável pela atividade de repassar o conhecimento de todo o contexto de negócio para a equipe de desenvolvimento. Muitas vezes, o PO pode ser o próprio cliente ou qualquer funcionário que tenha conhecimento do problema e faz o intermédio entre a equipe de desenvolvimento e o cliente. [Beck 2000]

- *Product Backlog*

  Representa a produção do trabalho executado durante o desenvolvimento.[Sanches, Luiz et al. 2010].

- *Sprint Backlog*

  Representa o trabalho a ser desenvolvido durante uma *sprint* com o objetivo de criar um produto apresentável para a equipe. O *backlog* da *sprint* deve ser produzido de forma incremental.

2 Processo de Engenharia de Requisitos

Inicialmente, foi necessário entender o problema do qual iriamos tratar, traçar características e definir uma visão com o cliente para impedir problemas futuros, como, por exemplo, problemas de comunicação causados por ambiguidade ou coisas parecidas, estas informações estão esclarecidas no Documento de Visão, presente na sessão 3 deste documento.

Após a definição do Documento de Visão iniciou-se a parte de elicitações de requisitos, a modelagem desta está representada na imagem 1

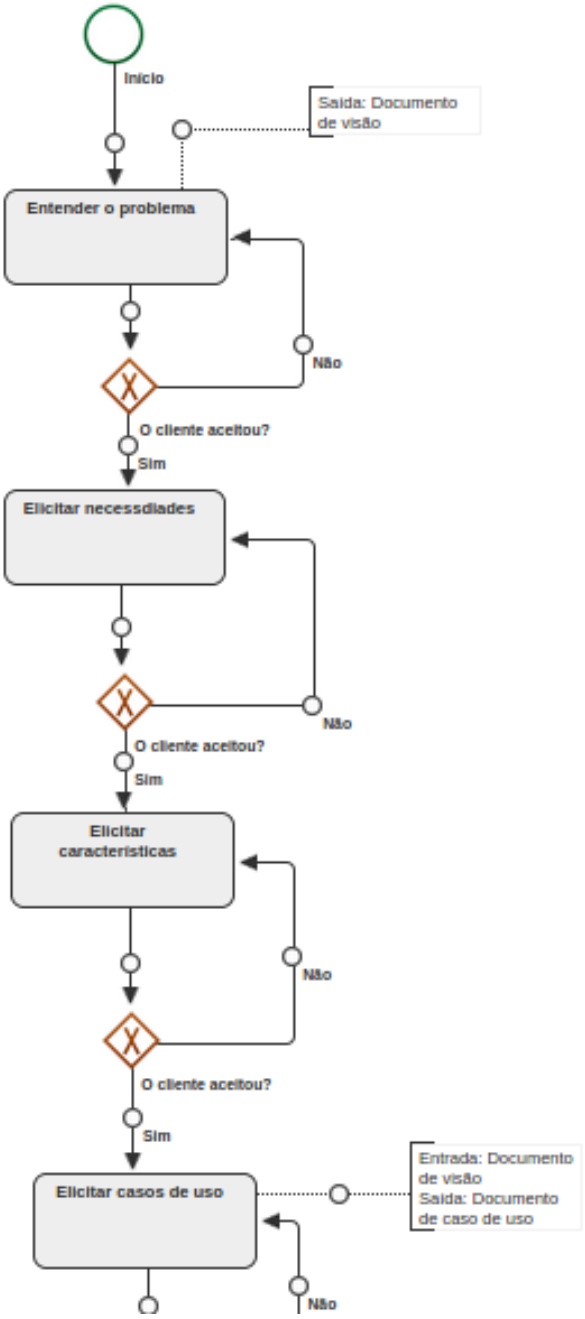

Figura 1. Modelagem Parte 1

Após os casos de uso do projeto definidos, vem a parte de definição de prioridades, criação dos *road maps*, assim como detalhamento dos casos de uso e implementação das funcionalidades com maior prioridade do projeto, a modelagem do mesmo esta presente na imagem 2.

O detalhamento dos casos de uso está presente na sessão 5 deste documento, assim como os roads maps se encontram na sessão 4 .

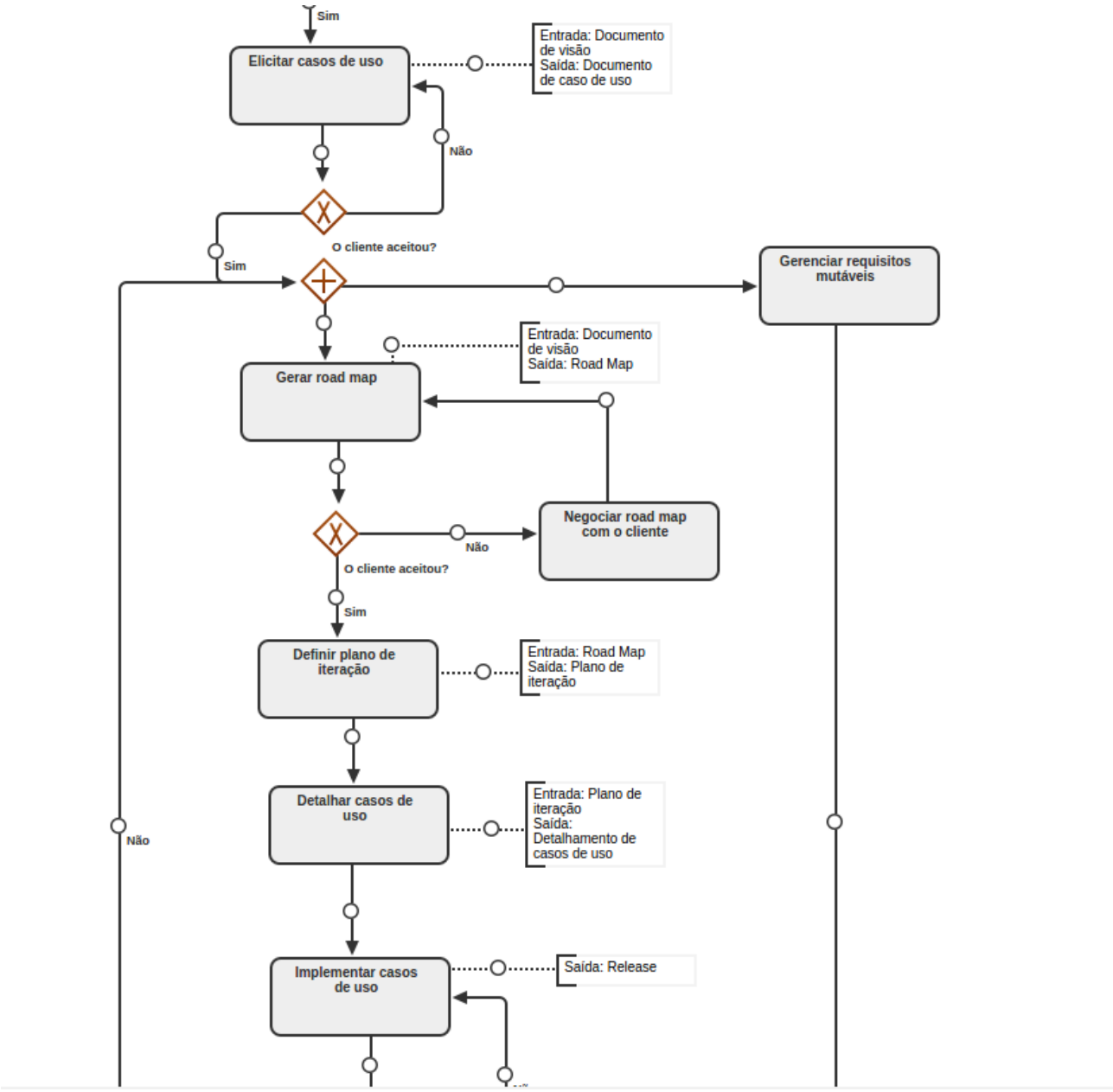

Figura 2. Modelagem Parte 2

Após a implementação dos casos de uso da iteração, caso o cliente aceite, segue-se para a próxima iteração ou final do projeto, dependendo se existe ou não outros casos de uso a serem implantados, caso haja, o processo volta para as atividades de gerar o road map e gerenciar requisitos mutáveis presentes na figura 2, assim como mostra a figura 3.

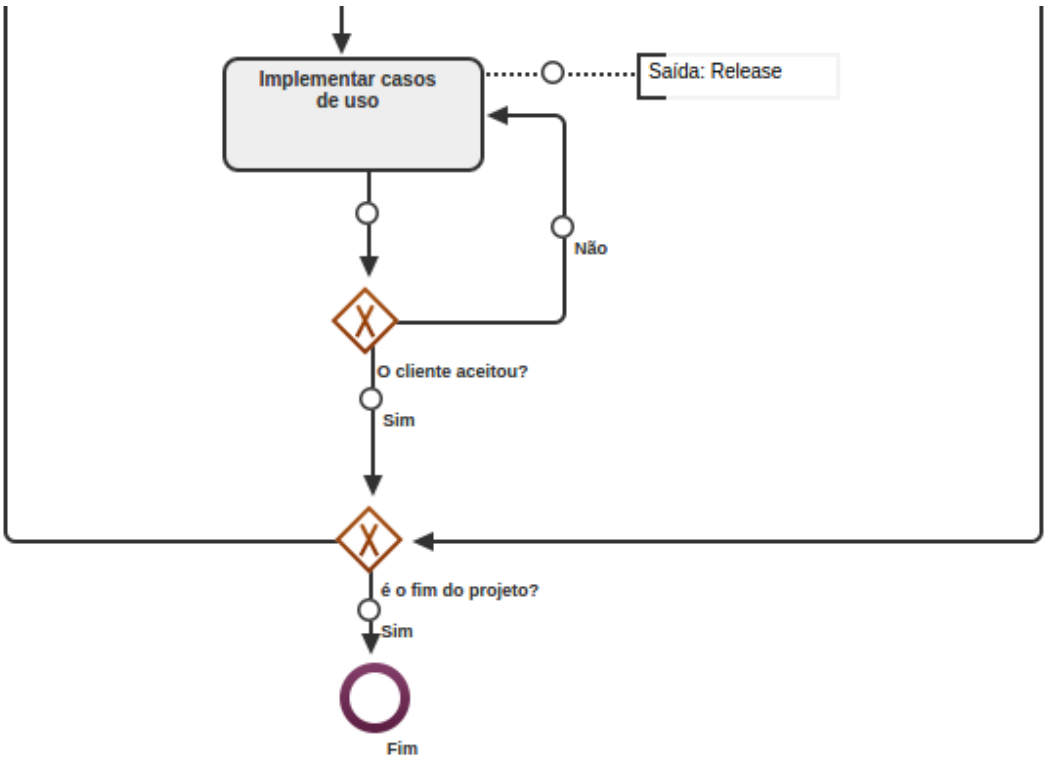

Figura 3. Modelagem Parte 3

3 Documento de visão

O documento de visão tem como objetivo definir uma visão geral do projeto, apresentar os problemas, os requisitos funcionais, não funcionanis, atores, entre outras informações que serão definidos com o cliente a fim de garantir que a equipe de desenvolvimento e o cliente estejam na maior sincronia possível [IBM 2014].

3 .1 Posicionando

3 .1.1 Oportunidade de Negócios

Atualmente, as ferramentas no mercado possuem limitações, como de qualidade, falta de flexibilidade na gerência, ou até mesmo o fechamento do código, que pode ser considerado uma limitação devida a redução de mão de obra para manutenção e evolução.

3 .1.2 Instrução do Problema

A Engenharia de Requisitos possui diversas *rotas* possíveis para se seguir, como, por exemplo a rota ágil, tradicional ou até mesmo uma mistura das duas.

Infelizmente, cada ferramenta de gerência de requisitos é voltada para uma dessas possibilidades, tornando díficil a tarefa voltada para outras, gerando assim nos engenheiros de requisitos a necessidade de aprender a utilizar diversas ferramentas para poder organizar projetos com *rotas* diferentes.

A utilização de apenas uma ferramenta que abrangesse as duas metodologias e ainda uma mistura das duas resolveria todo problema de gerência de requisitos em projetos que não se adequam a uma metodologia específica perfeitamente.

3 .1.3 Instrução de Posição do Produto

Para os engenheiros de Engenharia de Requisitos, a R.A.D.I.T. representará um avanço nas atividades de gerenciamento, pois os mesmos apenas precisarão aprender as funcionalidades de uma ferramenta, simplificando

a mudança entre projetos que tomam rotas distintas.

3 .2 Matriz de rastreabilidade de requisitos

A matriz de rastreabilidade resume o modo como serão organizados os requisitos do sistema, e a escolhida para o projeto esta ilustrada na figura 4.

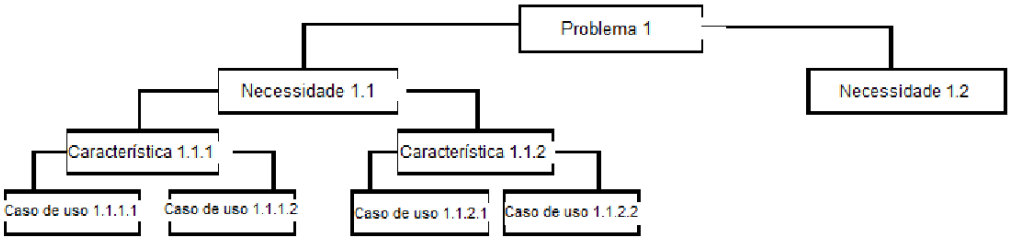

Figura 4. Matriz de rastreabilidade

3 .3 Descrições da Parte Interessada e do Usuário

Nesta sessão serão identificados e detalhados os interessados e usuários da R.A.D.I.T..

3 .3.1 Resumo da Parte Interessada e do Usuário

Para melhor entendimento das características e responsabilidades dos interessados, utilizou-se uma tabela que apresenta todos os interessados no sistema, suas descrições, responsabilidades e os critérios de sucesso de suas funções na equipe, ilustrada na tabela 1. Com esta tabela, pode-se obter o entendimento necessário sobre os interessados e o quão importante eles são para o sucesso do sistema.

Interessado	Descrição	Responsabilidade	Critérios de Sucesso
Analista de Requisitos	Membro da equipe de desenvolvimento com facilidade em comunicação, psicologia, sociologia, filosofia e mais áreas que possam facilitar a relação com o cliente. Seu conhecimento na área pode ser, dependendo da organização, baixo.	Pessoa responsável por realizar a elicitação dos requisitos junto ao usuário. Deve elicitar os requisitos de forma adequada à garantir sucesso no desenvolvimento do *software*.	Requisitos corretamente elicitados e prontos para serem documentados.
Gerente de Requisitos	Conhecedor de todo o processo de desenvolvimento e com contato frequente com o cliente. Seu conhecimento deve ser alto.	Pessoa responsável por administrar os requisitos durante todo processo de desenvolvimento de *software*, garantindo o mínimo esforço em casos de mudança de requisitos.	Requisitos bem administrados para, no caso de mudanças nos requisitos, existir o menor impacto possível na equipe de desenvolvimento.
Programador	Pessoa com capacidade em linguagens e lógica de programação	Implementar o sistema utilizando as técnologias definidas	Implementação do sistema de acordo com os requisitos levantados e cadastrados na ferramenta

Tabela 1. Parte Interessada

3 .3.2 Principais Problemas e Necessidades da Parte Interessada

O problema a ser resolvido pela R.A.D.I.T. deve estar bastante claro entre todos os *Stakeholders*, para que o desenvolvimento passe pela menor quantidade possível de dificuldades quanto ao entendimento de onde focar esforços para desenvolver a solução.

Para o mapeamento do problema principal e suas causas, foi utilizada a técnica do *Diagrama de Ishikawa*, que se contra na figura 5.

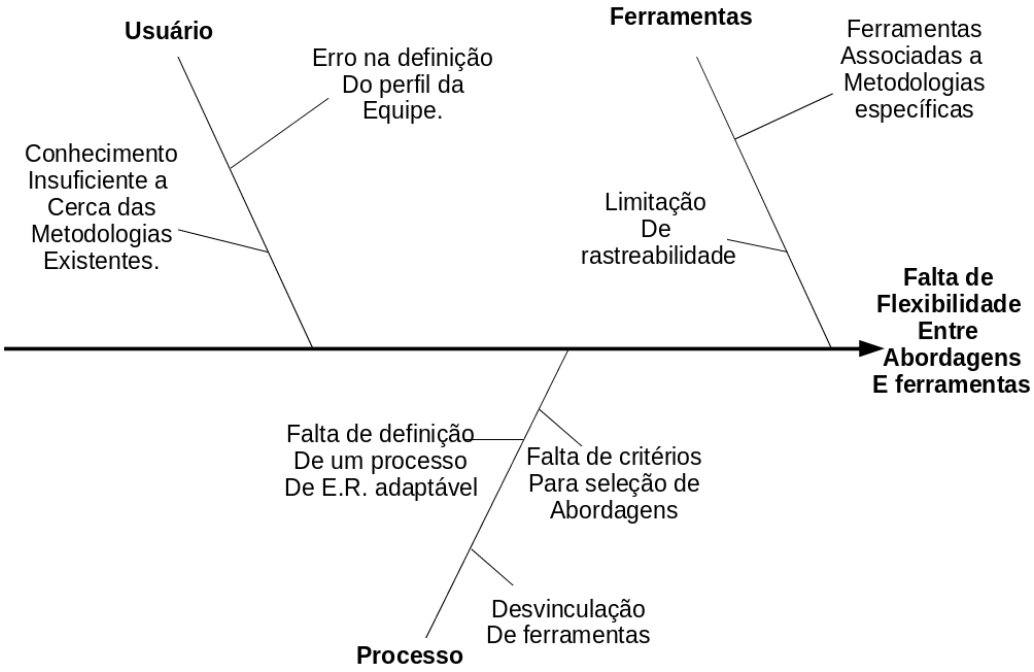

Figura 5. Diagrama de Ishikawa

Para melhor entendimento do problema, utilizamos a técnica de *Framework de problema*, que consiste em criar uma tabela apresentando o problema, os afetados, o impacto e qual seria uma solução bem sucedida, o Framework esta retratado na tabela 2.

O Problema:	Falta de flexibilidade entre Abordagens e Ferramentas.
Afeta:	Todos os desenvolvedores de *software* que necessitam de uma flexibilidade maior na gerência de requisitos.
Cujo impacto é:	Processo de requisitos mal gerenciados, aumentando a possibilidade de erros durante o desenvolvimento.
Uma solução bem sucedida seria:	Utilização de uma ferramenta que faça gerência de requisitos de forma flexivel, podendo utilizá-la em qualquer metodologia.

Tabela 2. Framework de Problema

Após o entendimento do problema, vê-se necessária a documentação das necessidades do cliente. Utilizou-se de uma técnica chamada *framework de necessidades* na qual são apresentados todos os problemas, as necessidades, a solução atual e a solução proposta. Dessa forma, pode-se obter um entendimento mais organizado dos problemas e necessidades do cliente, de acordo com o retradado na tabela 3.

Necessidade	Problema	Solução Atual	Solução Proposta
Utilização de ferramentas que se adequem as metodologias.	Ferramentas associadas a metodologias específicas.	Equipe utiliza mais de uma ferramenta para abranger as abordagens utilizadas.	Criação de uma ferramenta que seja flexível para qualquer metodologia, abrangendo todas as agordagens e até a mesclagem das mesmas.
Apoio a utilização de uma rastreabilidade organizada e eficiente em qualquer abordagem.	Limitação de ratreabilidade.	A equipe precisa criar sua rastreabilidade sem o apoio de uma ferramenta flexível.	Criação de uma ferramenta que gere a rastreabilidade dos requisitos de forma organizada e eficiente para qualquer abordagem.
Obter critérios fixos que redirecionem o projeto para a abordagem mais adequada.	Falta de critérios para seleção de abordagens.	Equipe precisa estudar as características do projeto e decidir qual a abordagem mais adequada.	Criação de uma ferramenta que recolha as características do projeto e apresente a abordagem mais adequada.
Obter um processo de *E.R.* adaptável a qualquer abordagem.	Falta de definição de um processo de *E.R.* adaptável.	Utilização de um processo inflexível e voltado apenas para uma abordagem.	Criação de uma ferramenta que gerencie processos flexíveis.
Gerar documentação de qualidade e fácil entendimento	Dificuldade em gerar documentação para pessoas de fora da equipe	Utilização de ferramentas a parte para gerar Diagramas de Caso de Uso e *Diagramas de Ishikawa*.	Integração da documentação na ferramenta de requisitos.

Tabela 3. Framework de Necessidades

3 .4 Visão Geral do Produto

Nesta seção, pode-se ter um entendimento geral de como será o produto final, quais serão suas características, como serão suas funcionalidades e etc.

3 .4.1 Perspectiva do Produto

O produto se encontrará em um contexto onde existem inúmeras ferramentas com o mesmo propósito, porém, as ferramentas existentes são inflexíveis quando se trata da abordagem que será seguida durante o desenvolvimento de *software*. Esta falha será corrigida na R.A.D.I.T., que irá propor uma metodologia para cada projeto em particular de acordo com suas características.

A ferramenta pode ser autocontida, não necessitando do apoio de nenhum outro sistema, porém a utilização de ferramentas de modelagem de processos é bastante indicada para que a máxima organização do projeto seja alcançada.

3 .4.2 Resumo das Capacidades

O grande diferencial da R.A.D.I.T. será a flexibilização na abordagem que será seguida durante o gerenciamento de projetos de *software*. O sistema deverá indicar a melhor abordagem a ser seguida pela equipe de desenvolvimento, garantindo a otimização do processo de desenvolvimento.

A ferramenta será capaz de disponibilizar a opção de modificar a abordagem indicada pela ferramenta, para que a equipe de desenvolvimento possa escolher a abordagem na qual os mesmos se sentem mais a vontade.

3 .5 Recursos do Produto

Os recursos do produto são as funcionalidades do sistema com uma pequena descrição, o que futuramente será transformado em casos de uso, estes recursos estão organizados de acordo com a rastreabilidade proposta na figura 4, e serão detalhados no documento de casos de uso, presente na sessão 5 deste documento. Os seguintes atributos serão organizados da forma que se apresenta na 6:

Problema	Necessidade	Características	Casos de Uso
P1	N1.1	C1.1.1	UC1.1.1.1
			UC1.1.1.2
			UC1.1.1.3
			UC1.1.1.4
		C1.1.2	UC1.1.2.1
			UC1.1.2.2
			UC1.1.2.3
			UC1.1.2.4
	N1.2	C1.2.1	UC1.2.1.1
			UC1.2.1.2
		C1.2.2	UC1.2.2.1
			UC1.2.2.2
	N1.3	C1.3.1	UC1.3.1.1
	N1.4	C1.4.1	UC1.4.1.1
			UC1.4.1.2
	N1.5	C1.5.1	UC1.5.1.1
			UC1.5.1.2
		C1.5.2	UC1.5.2.1
			UC1.5.2.2

Figura 6. Tabela de rastreabilidade

Seguem os recursos do sistema:

3 .5.1 Problema 1 - Falta de flexibilidade entre abordagens e ferramentas

O problema 1, gera algumas necessidades, que serão colocadas a seguir.

3 .5.1.1 Necessidade N1.1 - Utilização de ferramentas que se adequem as metodologias

3 .5.1.1.1 Característica C1.1.1 - Manter metodologias tradicionais

- Caso de uso UC1.1.1.1 - Manter Problema;
- Caso de uso UC1.1.1.2 - Manter Necessidades;
- Caso de uso UC1.1.1.3 - Manter Características;
- Caso de uso UC1.1.1.4 - Manter Casos de Uso.

3 .5.1.1.2 Característica C1.1.2 - Manter metodologias ágeis

- Caso de uso UC1.1.2.1 - Manter Temas de Investimento;
- Caso de uso UC1.1.2.2 - Manter Épicos;
- Caso de uso UC1.1.2.3 - Manter Features;
- Caso de uso UC1.1.2.4 - Manter Histórias de Usuário.

3 .5.1.2 Necessidade N1.2 - Apoio a utilização de uma rastreabilidade organizada e eficiente em qualquer abordagem

3 .5.1.2.1 Característica C1.2.1 - Manter informações sobre requisitos

- Caso de uso UC1.2.1.1 - Manter Atributos;
- Caso de uso UC1.2.1.2 - Manter *Roadmaps*.

3 .5.1.2.2 Característica C1.2.2 - Manter relação entre Requisitos

- Caso de uso UC1.2.2.1 - Manter rastreabilidade horizontal entre os requisitos;
- Caso de uso UC1.2.2.2 - Manter rastreabilidade vertical entre os requisitos.

3 .5.1.3 Necessidade N1.3 - Obter critérios fixos que direcionem o projeto para abordagem mais adequada

3 .5.1.3.1 Característica C1.3.1 - Auxiliar na escolha da metodologia

- Caso de Uso UC1.3.1.1 - Definir Metodologia.

3 .5.1.4 Necessidade N1.4 - Obter um processo de Engenharia de Requisitos adaptável a qualquer abordagem

3 .5.1.4.1 Característica C1.4.1 - Criar processos Híbridos

- Caso de uso UC1.4.1.1 - Definir "hibridez" do projeto;
- Caso de uso UC1.4.1.2 - Manter processos híbridos.

3 .5.1.5 Necessidade N1.5 - Gerar documentação de qualidade e fácil entendimento

3 .5.1.5.1 Característica C1.5.1 - Gerar e manter diagramas

- Caso de uso UC1.5.1.1 - Gerar *Diagrama de Ishikawa*;
- Caso de uso UC1.5.1.2 - Gerar Diagramas de Casos de Uso.

3 .5.1.5.2 Característica C1.5.2 - Controlar projeto por toda sua duração

1. Caso de uso UC1.5.2.1 - Gerar plano de iteração;
2. Caso de uso UC1.5.2.2 - Controlar histórico de versão.

3 .6 Restrições

3 .6.1 Restrição Técnica

A ferramenta poderá ser executada pelos navegadores Google Chrome versão 37.0.2062.120 ou superior Firefox versão 33.0 ou superior, não sendo possível sua utilização no Internet Explorer ou Safari.

3 .7 Faixas de Qualidade

3 .8 Atributos do Recurso

Atributos de recursos são basicamente descrições dos requisitos em alguma área em específico. Durante o projeto foram utilizados os atributos de arquitetura, prioridade e status.

3 .8.1 Atributos de Arquitetura

Atributos de arquitetura são atributos que definem a complexidade arquitetural de se implementar algum requisito, por exemplo, se haverá necessidade de alterar arquitetura do software, e estão retratados na tabela 4

Atributo	**Descrição**
Grande	Para implementar o requisito, a arquitetura sofrerá uma grande alteração
Média	A arquitetura terá uma alteração consideravel na implementação do requisito
Baixa	A arquitetura terá uma pequena alteração para sustentar o requisito
Nenhuma	O requisito não terá impacto nenhum na arquitetura do projeto

Tabela 4. Atributo de Arquitetura

3 .8.2 Atributo de Prioridade

Atributos de prioridade são atributos que definem o quão importante um requisito é para o cliente, definindo se ele deve ser implementado o mais rápido possível ou se pode ter sua implementação adiada, estão retratados na tabela 5.

Atributo	**Descrição**
Alta prioridade	Os requisitos marcados com este atributo são requisitos que possuem um grande interesse do cliente
Média prioridade	Os requisitos marcados por este atributo são requisitos que o cliente possui um grande interesse, porém não existe necessidade de implementá-lo rapidamente.
Baixa prioridade	Os requisitos marcados por este atributo são requisitos que o cliente deseja, porém não são essenciais para o funcionamento da solução.

Tabela 5. Atributo de prioridade

3 .8.3 Atributos de Status

Atributos de status são atributos que indicam em que fase um requisito está, de acordo com a tabela a seguir, e estão retrados na tabela 6

4 RoadMap

Roadmaps são uma priorização dos recursos do sistema, para definir por qual requisito a implementação terá início.

Atributo	Descrição
Aceito	Requisito devidamente implementado e aceito pelo cliente
Implementado	Requisito implementado porém esperando aceitação
Detalhado	Requisito detalhado, esperando por implementação
Elicitação aceita	Requisito elicitado e aceito pelo cliente porém sem detalhamento
Elicitado	Elicitado porém esperando aceitação do cliente

Tabela 6. Atributo de status

Para gerar o *roadmap* foi gerada uma pontuação nos atributos dos recursos presentes nas tabelas 4 e 5, mostrada na tabela 7.

Atributo	Classificação	Pontuação
Prioridade	Alta prioridade	5
	Média prioridade	3
	Baixa prioridade	1
Arquitetura	Grande	7
	Média	5
	Baixa	3
	Nenhuma	1

Tabela 7. Pontuação dos Atributos

Utilizando a tabela 7, fomos capazes de fazer uma relação numérica para pontuar cada um dos recursos, apresentados na sessão 3 .5 deste documento, e fazer a escolha de qual deve ser implementado primeiro, esta relação está apresentada na tabela 8.

Recurso	Atributo de prioridade	Atributo de arquitetura	Pontuação final
Definir Metodologia	Alta prioridade	Média	10
Manter Problema	Alta prioridade	Grande	12
Manter Necessidades	Alta prioridade	Grande	12
Manter Características	Alta prioridade	Grande	12
Manter Casos de Uso	Alta prioridade	Grande	12
Manter Temas de Investimento	Alta prioridade	Grande	12
Manter Épicos	Alta prioridade	Grande	12
Manter Features	Alta prioridade	Grande	12
Manter Histórias de Usuário	Alta prioridade	Grande	12
Manter Atributos	Média prioridade	Baixa	6
Manter Rastreabilidade de Requisitos	Alta prioridade	Média	10
Gerar *Diagrama de Ishikawa*	Média prioridade	Média	8
Manter Atores do Projeto	Baixa prioridade	Baixa	4
Gerar Diagramas de Casos de Uso	Média prioridade	Média	8
Manter *Roadmaps*	Média prioridade	Média	8
Gerar plano de iteração	média prioridade	Nenhuma	4
Definir “hibridez” do projeto	Alta prioridade	Grande	12
Controlar histórico de versão	Baixa prioridade	Nenhuma	2

Tabela 8. Pontuação dos recursos

Utilizando os dados apresentados na tabela 8, podemos então gerar um *rank* da ordem em que as funcionalidades devem ser implementadas, e a ordem deve ser de acordo com a lista a baixo

1. Primeira prioridade
    - Manter problema;

- Manter Necessidade;
- Manter Características;
- Manter Casos de uso;
- Manter Temas de investimento;
- Manter Épicos;
- Manter Features;
- Manter Histórias de usuário;
- definir "hibridez" do projeto.

2. Segunda prioridade
    - Definir metodologia;
    - Manter rastreabilidade de requisitos;
3. Terceira prioridade
    - Gerar *Diagrama de Ishikawa*;
    - Gerar Diagrama de Caso de Uso;
    - Manter *roadmaps*.
4. Quarta prioridade
    - Manter Atributos.
5. Quinta prioridade
    - Manter atores do projeto;
    - gerar planos de iteração.
6. Sexta prioridade
    - Controlar histórico de versão.

Por uma decisão da equipe, a implantação irá ser iniciada pela metodologia ágil, e infelizmente não haverá tempo para a implementação de todos os requisitos, então o proposto roadmap para as iterações que serão realizadas estão descritos na tabela 9

	Iteração 1	**Iteração 2**
Casos de uso	• Definir metodologia; • Manter Temas de investimento; • Manter Épicos.	• Manter Features; • Manter Histórias de usuário.

Tabela 9. *Roadmap*

Este *roadmap* será utilizado mais a frente na sessão de 5 para auxiliar quais casos de uso devem ou não ser detalhados durante o processo de desenvolvimento.

5 Documento de casos de uso

O documento de casos de uso tem como finalidade o detalhamento a fundo dos recursos do programa, aqui chamados de caso de uso, listados na sessão 3 .5 deste documento, colocando todas as suas características, restrições e caminhos possíveis.

5 .1 Identificação dos atores

Neste contexto, atores são representações genéricas de usuários do sistema, podende ser qualquer utilizador, sem se preocupar com nome do executor, apenas com sua função, esses atores estão listados na tabela 10.

Atores	**Descrição**
Engenheiro de Requisitos	Responsável, em metodologias tradicionais, pela elicitação dos requisitos e pela manutenção da rastreabilidade do sistema
Analista de Requisitos	Responsável, em metodologias tradicionais, gerência dos requisitos
Product Owner	Responsável, em metodologias ágeis, por escrever histórias de usuário
Equipe de portfólio	Responsável, em metodologias ágeis, por gerir a parte do portfólio, como temas de investimento e épicos
Equipe de programa	Responsável, em metodologias ágeis, por gerir as features do sistema e manter a entrega das releases em dia
Time	Responsável, em metodologias ágeis, por implementar as histórias de usuário
Cliente	Responsável por validar os requisitos

Tabela 10. Atores do sistema

5 .2 Diagrama de casos de uso

5 .3 Detalhamento dos casos de uso

5 .3.1 Caso de Uso: UC-01x Buscar cruzeiros

Referências Bibliográficas

[Beck 2000]BECK, K. Extreme programming explained: embrace change. [S.l.]: Addison-Wesley Professional, 2000.

[Espindola, Majdenbaum e Audy 2004]ESPINDOLA, R. S. de; MAJDENBAUM, A.; AUDY, J. L. N. Uma análise crítica dos desafios para engenharia de requisitos em manutenção de software. In: WER. [S.l.: s.n.], 2004. p. 226–238.

[IBM 2014]IBM. Rational Unified Process. out. 2014. Disponível em: <http://pic.dhe.ibm.com/infocenter/rpcmpose/v2r0/inc

[Sanches, Luiz et al. 2010]SANCHES, F.; LUIZ, M. et al. Aplicação das abordagens scrum e xp em um processo de software. Sistemas de Informação & Gestão de Tecnologia., n. 3, 2010.

[Sommerville et al. 2003]SOMMERVILLE, I. et al. Engenharia de software. [S.l.]: Addison Wesley, 2003.

Anexos

- **ENTREVISTAS**
  - **Entrevista realizada dia 16 de outubro de 2014**

    Cliente: George Marsicano

    Foi planejado pela equipe de desenvolvimento uma entrevista superficial com seis perguntas, porêm, após a primeira pergunta respondida não foi mais necessário as outras perguntas, salvo a pergunta de número dois.

    São elas:

    1. Por que criar uma nova ferramenta de requisitos? As existentes nao te agradram?

    R: Não me agradam?

    A necessidade de criação ou não de uma ferramenta não vem da necessidade de ferramente em si. Vocês em primeiro passo estão modelando um processo de requisitos. Depois será realizado uma avaliação para saber se existe ou não alguma ferramenta que implemente este processo definido, e, por ultimo, caso não exista nenhuma ferramente, poderá ser desenvolvido um plug-in para alguma existente que seja possível tal adaptação, ou o desenvolvimento de uma nova ferramente por completo.

    Tudo dependerá a modelagem inicial para decidir o andamento e escopo do projeto.

    2. Se fosse possível resumir a sua necessidade em uma palavra, qual seria?

    R: Abrangência.