UNIVERSIDADE FEDERAL DO PAMPA

NASSER OTHMAN RAHMAN

O IMPACTO DO USO DE ENGENHARIA DE REQUISITOS EM PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

ALEGRETE-RS

NASSER OTHMAN RAHMAN

O IMPACTO DO USO DE ENGENHARIA DE REQUISITOS EM PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Software da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Orientador: SAM DA SILVA DEVINCENZI

ALEGRETE-RS

2014

Ficha catalográfica elaborada automaticamente com os dados fornecidos
pelo(a) autor(a) através do Módulo de Biblioteca do
Sistema GURI (Gestão Unificada de Recursos Institucionais) .

R147i Rahman, Nasser Othman
O IMPACTO DO USO DE ENGENHARIA DE REQUISITOS EM PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE / Nasser Othman Rahman.
89 p.

Trabalho de Conclusão de Curso(Graduação)-- Universidade Federal do Pampa, ENGENHARIA DE SOFTWARE, 2014.
"Orientação: Sam da Silva Devincenzi".

1. Engenharia de Software. 2. Engenharia de Requisitos. 3. Especificação de Software. 4. Validação. 5. Retrabalho. I. Título.

NASSER OTHMAN RAHMAN

O IMPACTO DO USO DE ENGENHARIA DE REQUISITOS EM PROCESSOS DE DESENVOLVIMENTO DE SOFTWARE

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Software da Universidade Federal do Pampa, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

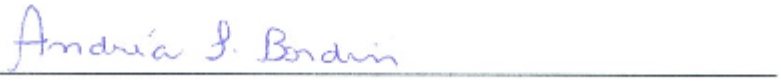
Trabalho de Conclusão de Curso defendido e aprovado em: 19 de Março de 2014.

Banca examinadora:

Prof. Me. Sam da Silva Devincenzi

Orientador

(UNIPAMPA)



Prof. Me. Andréa Sabedra Bordin

(UNIPAMPA)

Prof. Dr. Daniel Welfer

(UNIPAMPA)

RESUMO

A motivação para a realização deste trabalho foi a necessidade de definir um processo de especificação de requisitos para projetos desenvolvidos pelo Núcleo de Tecnologia de Informação e Comunicação (NTIC) na Universidade Federal do Pampa (UNIPAMPA), em virtude dos grandes entraves encontrados pela equipe de desenvolvimento de software dessa instituição, cuja falta de padronização acarreta em grande índice de retrabalho, além de aumento dos custos e prazos para entrega de seus produtos. O objetivo deste trabalho é definir um modelo de processo de desenvolvimento de software, que garanta uma especificação de requisitos eficiente, diminuindo custos e prazos, além de garantir uma melhor qualidade em produtos de software. Para realizar este trabalho, foi realizado um levantamento bibliográfico das principais técnicas e boas práticas de Engenharia de Requisitos, referenciadas nas literaturas escolhidas. Para realizar o experimento, foram analisados os artefatos gerados pelo NTIC, e a partir desses dados, foi definido um modelo de processo de desenvolvimento, com enfoque em especificação de requisitos, para ser executado em seus projetos de desenvolvimento de software. Os principais resultados encontrados foram o considerável decremento de horas dispensadas, pela equipe de desenvolvimento, em tarefas de retrabalho, além do menor índice defeitos encontrados na entrega do produto, e, consequentemente, o aumento da qualidade desses produtos.

Palavras-Chave: Engenharia de Software, Engenharia de Requisitos, Especificação de Software, Validação, Retrabalho.

ABSTRACT

The motivation for this work was the need to define a process for specifying requirements for projects developed by the Núcleo de Tecnologia de Informação e Comunicação (NTIC) at the Federal University of Pampa (UNIPAMPA) , because of the great obstacles encountered by the team software development of this institution, whose lack of standardization leads to high rates of rework, as well as increased costs and deadlines for delivery of your products. The objective of this work is to define a model of the software development process, which ensures efficient specification requirements, reducing costs and timelines, and ensure a better quality software products. To perform this work, a bibliographic survey of the main techniques and best practices in requirements engineering, referenced in the chosen literature was conducted. To perform the experiment, the artifacts generated by NTIC were analyzed, and from these data, a model of the development process was defined, focusing on requirements specification, to run on their software development projects . The main expected results are the decrement of hours exempted by the development team, task rework and increasing the quality of the delivered products.

Keywords: Software Engineering, Requirements Engineering, Software Specification, Validation, Rework.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

BPMN – Business Process Model and Notation
CAU – Coordenação de Apoio ao Usuário
CMM – Capability Maturity Model
CMMI – Capability Maturity Model Integration
CODEV – Coordenação de Desenvolvimento
DPN – Diagrama de Processos de Negócio
DRE – Desenvolvimento de Requisitos
E/S – Entrada / Saída
ER – Engenharia de Requisitos
FEUP - Faculdade de Engenharia da Universidade do Porto
GRE – Gerencia de Requisitos
GQM – Goal Question Metric
GURI – Gestão Unificada de Recursos Institucionais
HTML – HyperText Markup Language
IEEE – Institute of Electrical and Electronics Engineers
IU – Interfaces de Usuário
MPS-BR – Melhoria de Processos do Software Brasileiro
NTIC – Núcleo de Tecnologia da Informação e Comunicação
PRAEC – Pró-Reitoria de Assuntos Estudantis e Comunitários
SIE – Sistema de Informações Educacionais
SRS – Software Requierements Specification (Especificação de requisitos de software)
UML – Unified Modeling Language
UNIPAMPA – Universidade Federal do Pampa
XP – eXtreme Programming

1 INTRODUÇÃO

A evolução da sociedade cresce juntamente com o surgimento de novas tecnologias, novos equipamentos, novos materiais, etc. Hoje em dia, praticamente tudo que usamos é controlado por software, desde uma simples geladeira, até um grande sistema para controle do mercado financeiro mundial.

Segundo Sommerville (2011), existem vários tipos de sistemas de software, desde os simples sistemas embutidos até os sistemas de informação complexos, de alcance mundial.

Sendo assim, desenvolver um software passa a ser uma arte, visto que não basta apenas saber programar, mas também se faz necessário entender os problemas a serem solucionados a partir do software produzido.

Conforme dito anteriormente, a demanda por desenvolvimento de software cresceu muito nos últimos anos, consequentemente a necessidade de mão de obra qualificada cresce desde então.

Contudo, havia uma grande dificuldade no início em seguir padrões, visto que o desenvolvimento de software era algo relativamente novo, e não haviam padrões a serem seguidos. O processo de desenvolvimento era totalmente empírico, logo, começaram a surgir grandes problemas para a produção de software tais como: cronogramas atrasados, projetos abandonados, dificuldade em combinar módulos, programas que não retornavam os resultados esperados, dificuldades em utilização dos programas, custos elevados, manutenções extremamente caras, entre outros.

A situação foi se tornando cada vez mais caótica que, em 1968, foi criada uma conferência específica para tratar do assunto: *Software Engineering: Concepts and Techniques Proceedings of the NATO Conferences*, onde provavelmente foi a primeira vez que se utilizou o termo Engenharia de Software, visando tornar o processo de desenvolvimento algo sistemático e controlado.

> Engenharia de software é uma disciplina de engenharia cujo foco está em todos os aspectos da produção de software, desde os estágios iniciais da especificação do sistema até sua manutenção, quando o sistema já está sendo usado. (SOMMERVILLE, 2011, p. 5)

Um dos pilares para o desenvolvimento de software é a definição do problema, ou seja, como o software geralmente tem a finalidade de automatizar algum processo, se faz necessário entender esse processo, o que deve ser realizado, quais as restrições, etc., o que na Engenharia de Software é chamado de requisito. Paula Filho (2009) fala que requisitos são características que definem os critérios de aceitação de um produto. Assim sendo, requisitos é um fator crítico para o sucesso de um projeto de software.

Em contrapartida, nota-se que muitas vezes não é dada a importância devida para a especificação desses requisitos formalmente. Fazendo com que ao longo do projeto surjam muitas dúvidas sobre as reais necessidades dos clientes, gerando erros no desenvolvimento, perda de qualidade, muito retrabalho e aumento considerável no tempo e nos custos dos projetos.

Assim sendo, esse trabalho tem por finalidade analisar o quanto a gestão de requisitos é importante para garantir a qualidade nos processos de desenvolvimento de software. Para realizar este estudo, será realizada uma análise desse processo em projetos de desenvolvimento de software no âmbito da Universidade Federal do Pampa (UNIPAMPA).

1.1 Problemática

Devido a grande demanda por automatizar processos de trabalho no âmbito da UNIPAMPA, se fez necessário criar um setor exclusivo para desempenhar as funções referentes à tecnologia de informação e comunicação, assim sendo, foi criado o Núcleo de Tecnologia da Informação e Comunicação (NTIC) dentro das instalações da universidade.

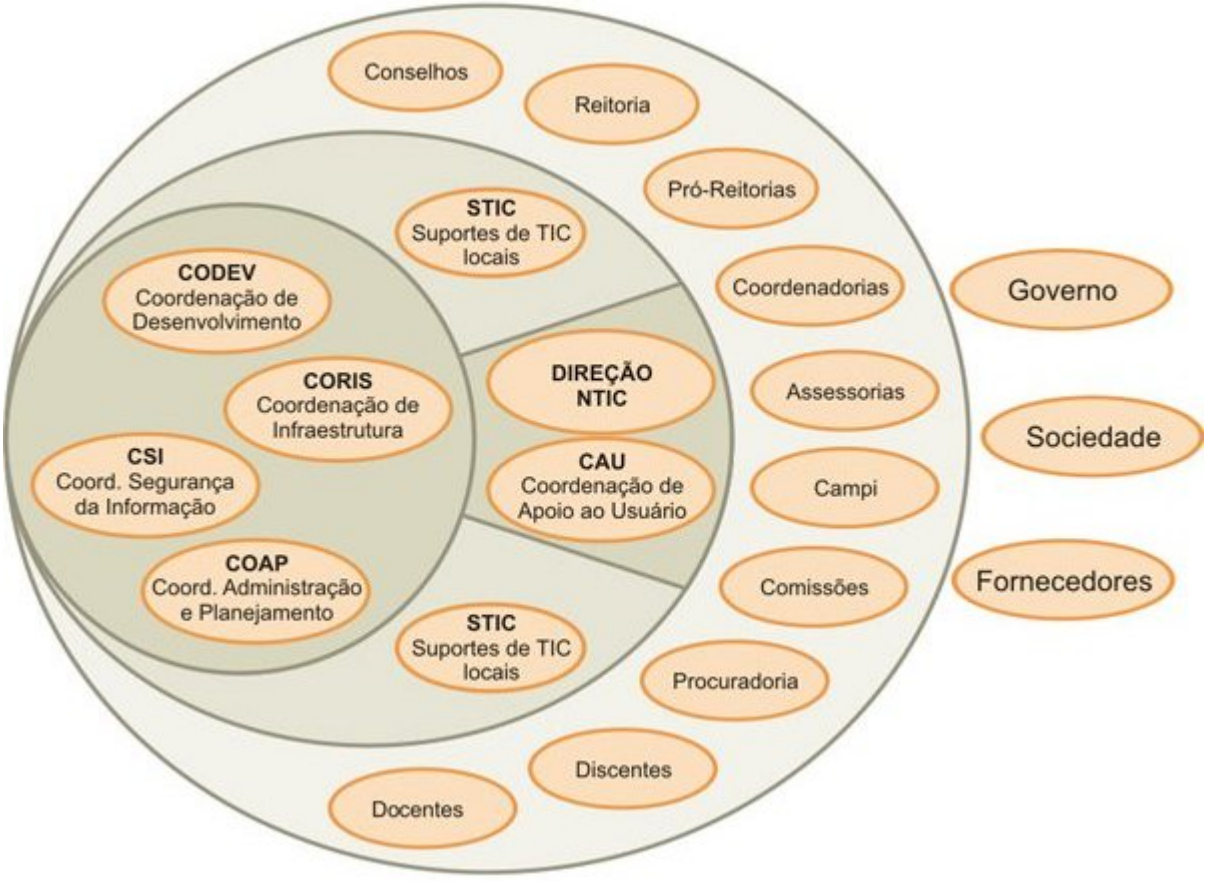
> O Núcleo de Tecnologia da Informação e Comunicação UNIPAMPA – NTIC, é órgão suplementar da Reitoria da Universidade Federal do Pampa, com estrutura prevista na Portaria UNIPAMPA nº 745 de 13 de abril de 2010 e tem por objetivo criar e manter condições para o funcionamento sistêmico das atividades ligadas à tecnologia da informação e comunicação na Universidade, a fim de dar suporte ao desenvolvimento do ensino, pesquisa, extensão, gestão e serviços à comunidade, de acordo com as diretrizes da Universidade. (UNIPAMPA)

Em virtude da denominação acima citada, uma das atividades do NTIC é o desenvolvimento de software para apoio aos objetivos elencados.

Conforme ilustrado na Figura 1, a Coordenação de Desenvolvimento (CODEV) é o setor responsável pelo desenvolvimento de software. Esse setor

atende diretamente as demandas advindas da Coordenação de Apoio ao Usuário (CAU), que é o setor que responsável identificar as necessidades dos usuários de toda a Universidade.

Figura 1 – Organograma NTIC



Fonte: (NTIC - Núcleo de Tecnologia da Informação e Comunicação)

No âmbito da Engenharia de Software, cabe à CAU realizar o atendimento direto ao cliente, e consequentemente, ser responsável pela especificação do software. As tarefas seguintes (projeto e implementação de software, validação de software e evolução de software) são atribuições da CODEV. Ou seja, a CAU define o que deve ser desenvolvido e a CODEV desenvolve.

Como a maioria da demanda por software é originada pela Reitoria, situada na cidade de Bagé-RS, a CAU foi transferida para essa cidade, e a equipe da CODEV permaneceu na cidade de Alegrete-RS. Essa distância entre ambas acabou acentuando a necessidade de uma comunicação bem alinhada para que o processo seja bem executado. Em virtude dessa diferença de local, a CODEV acaba não tendo contato direto com o cliente, sendo esse contato realizado pela CAU, assim,

para que a CODEV consiga ter um melhor entendimento das demandas a serem atendidas, se faz necessário uma boa especificação do problema.

Porém, a falta de padronização no processo de especificação, acaba gerando documentos incompletos, imprecisos e confusos, o que, agregados, influenciam em um entendimento precário pela equipe de desenvolvimento, acarretando em um software de má qualidade como produto final e gerando considerável e indesejável atividade de retrabalho.

1.2 Justificativa

Em virtude de especificações incompletas, a CODEV acaba gastando muito tempo tentando entender a definição dos projetos a serem desenvolvidos, consequentemente liberando produtos com prazos estourados, incompletos e com defeitos, além de haver um grande índice de retrabalho, tanto por parte da CAU, quanto da CODEV, tendo que corrigir os erros gerados pela má especificação.

Para garantir uma maior eficiência, se faz necessária uma especificação completa, para que a CODEV consiga interpretar da melhor maneira os requisitos e desenvolver os seus processos eficientemente.

Essa documentação garantirá um melhor entendimento entre as partes (cliente, CAU e CODEV), além de permitir uma gerência mais completa de todo o processo de desenvolvimento e justificando o objetivo deste trabalho.

1.3 Objetivos

Em desenvolvimento de software, é praticamente uma utopia imaginar que o produto será 100% aceito em sua primeira entrega, assim, a constante refatoração em função da evolução do produto é algo que não se pode fugir, principalmente quando se busca uma melhor qualidade no produto entregue, porém, o retrabalho é algo que realmente se deve manter distância. Diferentemente da refatoração, que tem objetivo de melhorar a qualidade do produto, o retrabalho é ter que refazer algo que está bem feito, porém não era o que o cliente necessitava. Além do tempo perdido, do custo gerado, existe ainda o fator emocional, onde os envolvidos acabam perdendo motivação e tornando o processo cada vez menos eficiente.

Este trabalho tem como objetivo principal diminuir o retrabalho causado por uma má especificação de requisitos. Em complemento a isso, os seguintes objetivos específicos serão buscados:

- Definir um modelo de processo de desenvolvimento de software, com foco em engenharia de requisitos que, ao ser implantado, garanta que a especificação dos requisitos de software seja mais eficiente;
- Diminuir tempo de entrega;
- Diminuir o custo final do produto;
- Aumentar a qualidade do produto entregue.

1.4 Organização do documento

O conteúdo deste documento está distribuído da seguinte forma:

Capítulo 2: apresenta a fundamentação teórica do trabalho, contendo as definições de Engenharia de Software, requisitos e especificação de requisitos, além da definição de métricas;

Capítulo 3: são apresentados alguns trabalhos que são relacionados com o tema em questão, justificando a necessidade de seu estudo, além de ser base para a modelagem do processo proposto;

Capítulo 4: é apresentada a metodologia de execução deste trabalho, onde são definidas e apresentadas a sequências da metodologia utilizada para a realização do mesmo;

Capítulo 5: onde é apresentada a proposta de solução de um processo de desenvolvimento de software, para ser executado pela CAU e pela CODEV;

Capítulo 6: são apresentados os resultados obtidos através de experimento realizado no NTIC – Unipampa;

Capítulo 7: são apresentadas as considerações finais, identificadas a partir do estudo realizado;

Capítulo 8: é apresentado o cronograma de execução deste trabalho;

Capítulo 9: são apresentadas as referências bibliográficas utilizadas para fundamentar este trabalho;

APÊNDICE A: apresenta os resultados obtidos através dos questionários utilizados para a avaliação do NTIC-Unipampa;

APÊNDICE B: apresenta a avaliação dos artefatos utilizados em um dos projetos executados pelo NTIC-Unipampa;

APÊNDICE C: apresenta a cópia do documento enviado ao NTIC-Unipampa, contendo o modelo de processo de software, desenvolvido neste trabalho, para a sua consequente execução.

2 CONCEITOS GERAIS

Neste capítulo serão apresentados os conceitos utilizados para a realização do estudo em questão. Foi realizado um estudo nas principais bibliografias encontradas sobre as áreas de Engenharia de Software e Engenharia de Requisitos, que serviram de base para a realização do trabalho.

Primeiramente é apresentado o conceito e a origem da Engenharia de Software e posteriormente é focado o estudo referente à Engenharia de Requisitos. No fim desse capítulo, é apresentada uma metodologia de métrica.

2.1 Engenharia de Software

O termo “Engenharia de Software” surgiu no final da década de 1960, durante uma época conhecida como a Crise de Software.

Como o desenvolvimento de software era algo relativamente novo, porém com crescimento acentuado, não existiam na época padrões a serem seguidos. Os software construídos eram praticamente produtos artesanais, com baixa qualidade, altos custos, e difícil manutenção. Com a falta desses padrões, cada programador desenvolvia da sua maneira, empiricamente.

Com a necessidade de software cada vez maiores, mais complexos, e em prazos cada vez menores, ficou cada vez mais evidenciada a necessidade da criação de técnicas e padrões para o gerenciamento desses projetos.

Paula Filho (2009), diz que a tecnologia só resolve problemas quando é usada por pessoas qualificadas, dentro de processos adequados.

Sendo assim, a Engenharia de Software surgiu para ser um guia para o desenvolvimento de sistemas de software, definindo processos, técnicas e métodos para a gestão e o desenvolvimento de software cada vez melhores, com menores prazos, e principalmente, menores custos.

Sommerville (2011) divide os processos da engenharia de software em quatro fases:

1. Especificação de software: a funcionalidade do software e as restrições a seu funcionamento devem ser definidas;

2. Projeto e implementação de software: O software deve ser produzido para atender às especificações;

3. Validação de software: O software deve ser validado para garantir que atenda às demandas do cliente;

4. Evolução de software: o software deve evoluir para atender às necessidades de mudança do cliente.

Cada um dos processos citados possuem vários subprocessos necessários para o desenvolvimento de software.

Para não fugir do escopo deste trabalho, iremos detalhar apenas o primeiro processo, que é a Especificação de Software.

2.2 Requisitos

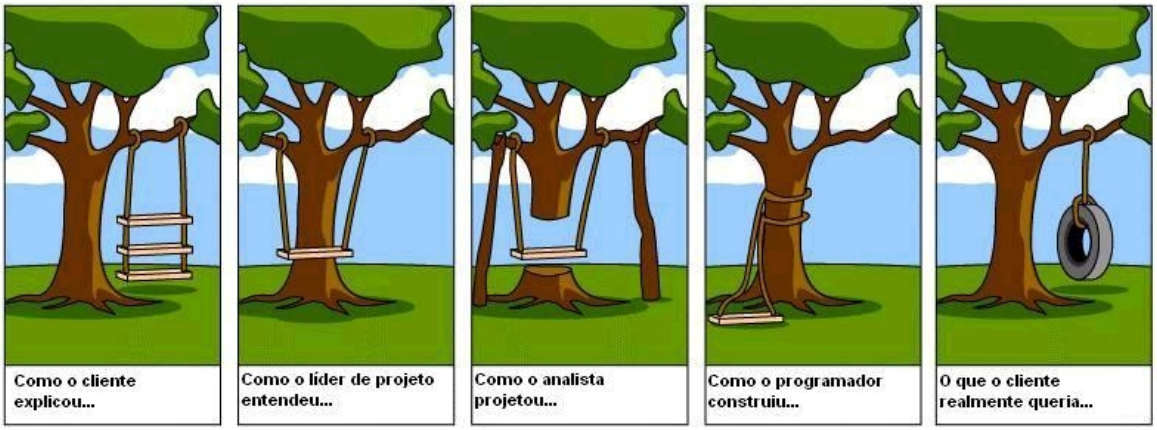
Certamente a maioria dos software produzidos no mundo são gerados a partir de necessidades identificadas por alguém, que podem ser gerentes, clientes, usuários finais, etc. Logo, a criação de um software se justifica pela necessidade de automatização de algum processo já existente.

Paula Filho (2009, p. 8) fala que “requisitos são características que definem os critérios de aceitação de um produto”. Essas necessidades são a alma do desenvolvimento. De que adianta gastar tempo e dinheiro para desenvolver algo que ninguém irá usar, ou que irá gerar resultados incorretos, ou pior, que não realize o que realmente foi exigido. Requisito é o termo utilizado, na área de desenvolvimento, para referenciar essas necessidades de usuários.

A grande dificuldade da gestão dos requisitos é ter a garantia de que todos os envolvidos no projeto tenham a mesma percepção de cada requisito. Visto que em um projeto participam várias pessoas, com diferentes visões e pontos de vista, e cada qual pode interpretar um requisito de forma diferente. Muitas vezes podem ocorrer omissões, por exemplo, por parte dos clientes, ao não deixar claro tudo o que necessita, imaginando que muitas coisas estão subentendidas. Por vezes analistas mal escutam os clientes, imaginando que já entenderam o problema, existe ainda a situação que nem o cliente sabe exatamente o que realmente quer.

Sendo assim, requisitos mal interpretados, podem gerar ambiguidade e fazer com que o projeto não gere os resultados esperados, tanto no quesito produto, quanto no quesito custo. A Figura 2 ilustra como muitas vezes ocorre o entendimento de cada participante do projeto no entendimento dos requisitos.

Figura 2 – Caricatura de problemas no entendimento de requisitos



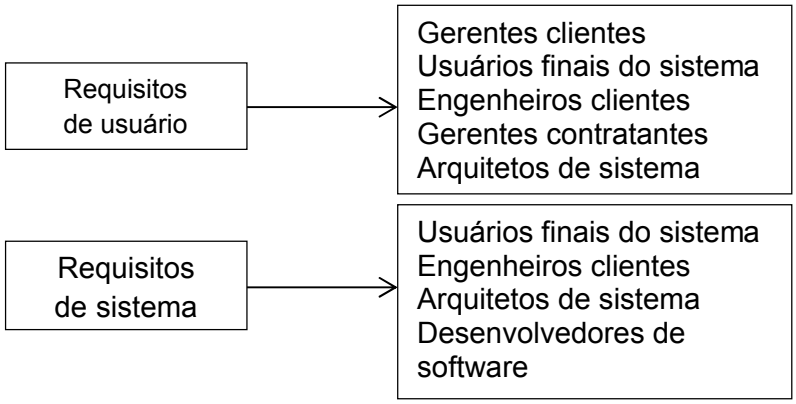
Fonte: Desconhecida

2.2.1 Níveis de descrição de requisitos

O documento de requisito pode se tornar um texto longo, com vários termos técnicos e, por muitas vezes, de leitura cansativa. Esses termos técnicos acabam inibindo a leitura completa do documento de requisitos pelos usuários e clientes, visto que esses termos não são úteis para o entendimento do problema por parte deles, em contrapartida, a falta de detalhamento pode ser um grave entrave para a equipe de desenvolvimento, não deixando explícito o que eles realmente devem desenvolver. Para resolver esse impasse, Sommerville (2011) encoraja a separar os requisitos em dois níveis: requisitos de usuário e requisitos de sistema. Para que cada um dos envolvidos tenha em mãos um documento que seja de fácil leitura e entendimento.

A Figura 3 mostra a quem cada nível de descrição está direcionado.

Figura 3 – Descrição dos requisitos



Fonte: (SOMMERVILLE, 2011, p. 59)

2.2.1.1 Requisitos de usuário

Devem expressar quais os serviços que o sistema a ser desenvolvido deve conter, em linguagem natural e com uso de diagramas, visando ficar de fácil compreensão, e de forma mais abstrata, as necessidades dos clientes/usuários.

2.2.1.2 Requisitos de sistema

Seria uma descrição mais detalhada, contendo as funções e as restrições operacionais do sistema. Esses requisitos tem foco de uso pela equipe de desenvolvimento, podendo inclusive ser utilizada como parte de um contrato entre o comprador e o desenvolvedor.

2.2.2 Classificação de requisitos

Os requisitos são basicamente divididos em dois tipos: Requisitos Funcionais e Requisitos Não Funcionais.

2.2.2.1 Requisitos Funcionais:

“Estes requisitos descrevem as funcionalidades que se espera que o software forneça quando estiver pronto, como entradas, saídas, exceções, etc.“ (KOSCIANSKI; SOARES, 2007, p. 179).

Já Sommerville (2011) diz que requisitos funcionais descrevem o que ele deve fazer.

Em suma, requisitos funcionais são as funcionalidades do sistema, é a descrição das ações possíveis que o software fornecerá ao usuário.

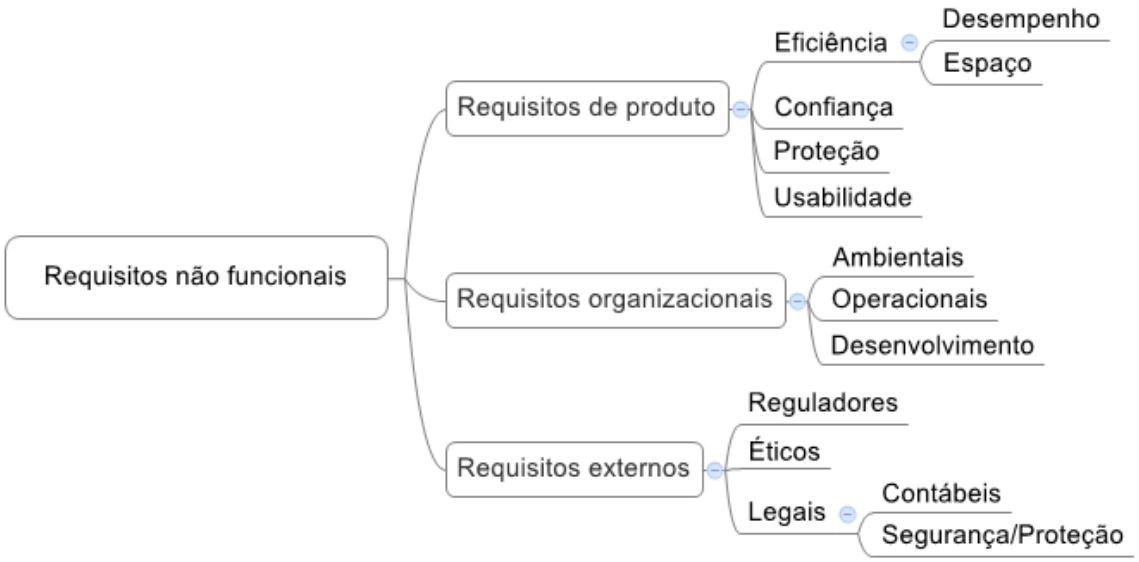
2.2.2.2 Requisitos Não Funcionais:

“Os requisitos não funcionais descrevem restrições ao software de forma geral. Não são, portanto, relativos diretamente às funções desempenhadas pelo produto.” (KOSCIANSKI; SOARES, 2007, p. 179)

Corroborando com essa afirmativa, Sommerville (2011) descreve como requisitos não funcionais os requisitos que não estão diretamente relacionados com os serviços específicos oferecidos pelo sistema a seus usuários.

Ou seja, requisitos não funcionais estão ligados ao software como um todo, geralmente estão ligados à confiabilidade, segurança, desempenho, disponibilidade, legais, etc. Sommerville (2011) ainda classifica os requisitos não funcionais em três subcategorias: Requisitos de produto, organizacionais e externos, a Figura 4 ilustra os possíveis níveis de requisitos não funcionais.

Figura 4 – Requisitos não funcionais



Fonte: Adaptado de (SOMMERVILLE, 2011, p. 61)

2.3 Engenharia de Requisitos

Controlar e gerenciar os requisitos são um dos pilares do desenvolvimento de software, visando à garantia de qualidade do mesmo.

> A engenharia de requisitos é uma ação de engenharia de software importante que se inicia durante a atividade de comunicação e continua na de modelagem. Ela deve ser adaptada às necessidades do processo, do projeto, do produto e das pessoas que estão realizando o trabalho. (PRESSMAN, 2011, p. 127)

> A maior dificuldade em modelarmos um sistema não está nos diagramas que temos ou desenhar, no código que devemos criar ou nas bases de dados que devemos projetar. Na realidade está nos requisitos que devemos gerenciar. (MELO, 2010, p. 53)

Corroborando com as afirmações acima, Pfleeger (2004) cita em seu livro uma pesquisa realizada pela *Standish Group*, realizada em 1994, que analisou os fatores críticos para o insucesso de projetos de software. Foram entrevistadas 350 empresas e 8.000 projetos de software, cujo resultado está apresentado na Tabela 1.

Tabela 1 – Fatores Críticos de Insucesso

	Fatores Críticos	**%**
1.	Requisitos Incompletos	13,10%
2.	Falta de Envolvimento do Usuário	12,40%
3.	Falta de Recursos	10,60%
4.	Expectativas Irreais	9,90%
5.	Falta de Apoio Executivo	9,30%
6.	Mudança de Requisitos e Especificações	8,70%
7.	Falta de Planejamento	8,10%
8.	Sistema não mais necessário	7,50%

Fonte: (PFLEEGER, 2004, p. 112)

Conforme podemos observar na Tabela 1, os itens (1) Requisitos Incompletos, (2) Falta de envolvimento do Usuário e (6) Mudanças de Requisitos e Especificações estão diretamente relacionados com Engenharia de Requisitos (ER), impactando em 34,2% dos projetos fracassados.

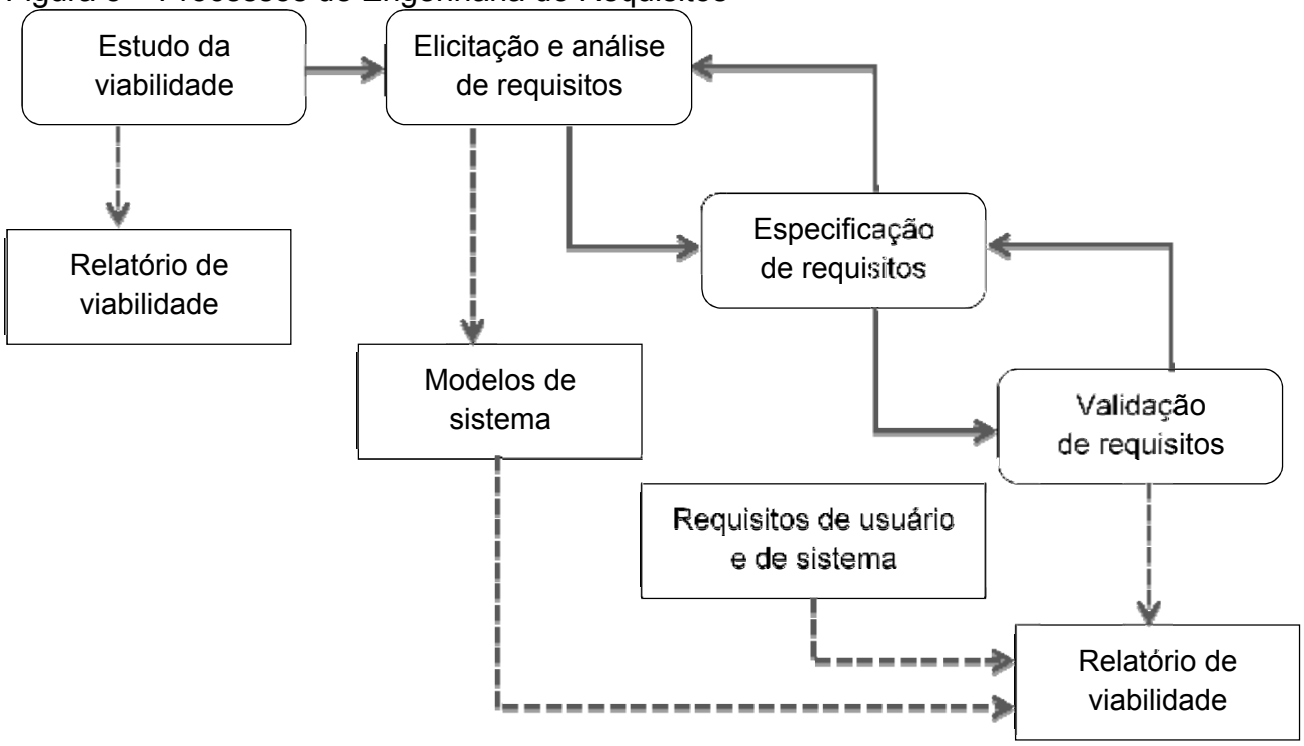
No capítulo 3.2 deste trabalho apresenta uma pesquisa realizada por (GIANESINI, 2008), da situação da área de ER, onde a autora acaba ratificando os números citados pela Tabela 1.

O processo de ER, conforme citado por Sommerville (2011), é dividido em quatro atividades: estudo de viabilidade, elicitação e análise de requisitos, especificação de requisitos e validação de requisitos. A Figura 5 demonstra o processo de ER.

2.3.1 Estudo de Viabilidade

Sommerville (2011, p. 24) diz que o estudo de viabilidade é a “estimativa acerca da possibilidade de se satisfazerem as necessidades do usuário identificado, usando-se tecnologias atuais de software e hardware”. Ou seja, antes de começar o projeto propriamente dito, se faz necessário o estudo da viabilidade do desenvolvimento do mesmo, visando analisar, entre outros aspectos, se o custo está dentro do orçamento, se a solução é possível com as tecnologias atuais, se o prazo de entrega é compatível com o previsto.

Figura 5 – Processos de Engenharia de Requisitos



Fonte: (SOMMERVILLE, 2011, p. 24)

2.3.2 Elicitação e Análise de Requisitos

É nesta etapa do processo de Engenharia de Requisitos que irão surgir os requisitos, Sommerville (2011, p. 25) descreve elicitação e análise de requisitos como "processo de derivação dos requisitos do sistema por meio da observação dos sistemas existentes".

A elicitação e análise de requisitos podem envolver várias pessoas, com diferentes visões do problema e que possuem influência direta ou indireta no software a ser desenvolvido. Cabe ao analista identificar, junto aos *stakeholders*[1] do sistema, os requisitos do problema. Para isso, existem algumas técnicas que auxiliarão os analistas a identificá-los. Segundo (KOSCIANSKI; SOARES, 2007, p. 180), "não existe um processo ideal de levantamento de requisitos que seja adaptável a todas as empresas", sendo assim, cabe ao analista definir qual a técnica de melhor se adapta a situação de cada projeto a ser desenvolvido. Serão descritas abaixo algumas técnicas citadas pela literatura que poderão auxiliar nesse processo.

[1] *Stakeholder* – termo utilizado para referenciar as partes interessadas em um projeto de software

2.3.2.1 Entrevistas

Essa é uma das técnicas mais utilizadas em ER. Consiste em reuniões formais, ou informais, entre os analistas e os *stakeholders*, para definir os requisitos.

As entrevistas podem ser realizadas de duas formas:

- Entrevistas fechadas: os *stakeholders* respondem perguntas montadas *a priori* pelos analistas.

- Entrevistas abertas: as perguntas vão surgindo durante a entrevista, a partir das respostas colhidas.

Geralmente as entrevistas são uma mescla das duas, fechadas e abertas, onde se criam algumas perguntas antes de começar a entrevista, e durante a mesma, surgem novas perguntas a serem respondidas, de uma forma menos estruturada. O importante é tentar abstrair o máximo possível os requisitos através dessa entrevista

Sommerville (2011) reitera que a abordagem mista é bem interessante, cita inclusive que entrevistas totalmente abertas raramente funcionam bem.

É importante também ressaltar que as questões realizadas devem levar em conta qual é o papel do *stakeholder* na organização, visto que em algumas situações o *stakeholder* entrevistado não possui o domínio total do problema.

2.3.2.2 Etnografia

Segundo (KOSCIANSKI; SOARES, 2007, p. 182), etnografia é uma “técnica de observação que pode ser utilizada para compreender os requisitos sociais e organizacionais“.

Sendo mais abrangente, Sommerville (2011) diz que é a técnica de observação que pode ser usada para compreender os processos operacionais e ajudar a extrair os requisitos de apoio para esses processos.

Ou seja, etnografia é uma técnica que consiste em imergir o analista no ambiente de trabalho para a qual está desenvolvendo a solução. Esse, por sua vez, observará de forma imparcial os processos utilizadas pelos *stakeholders* e conseguirá visualizar exatamente como eles trabalham, conseguindo identificar requisitos implícitos e detalhes que são difíceis de expressar.

2.3.2.3 Questionários

Segundo (KOSCIANSKI; SOARES, 2007, p. 183), os questionários são utilizados quando não é possível utilizar a etnografia ou entrevistar os *stakeholders* em razão da distância geográfica ou da falta de disponibilidade destes.

Consiste em montar um questionário de forma sistemática e enviado aos *stakeholders*, para que, a partir das respostas, o analista consiga absorver as informações do problema.

É útil como ferramenta adicional na análise, visto que muitos requisitos podem não aparecer no questionário. Pode ser utilizada juntamente com as entrevistas, quando se envia um questionário antes da entrevista, para que o analista consiga ter uma visão melhor do problema e que consuma menor tempo na entrevista. Podendo inclusive ser ao contrário, montando um questionário após a análise de uma entrevista para refinar os requisitos que possam não ter ficado claros na entrevista.

2.3.2.4 Prototipação

A prototipação consiste em criar modelos de telas, painéis, etc., de baixa resolução, com o intuito apenas de visualizar de melhor forma as funcionalidades de um sistema.

> O protótipo de requisitos é um protótipo visual de baixa fidelidade, que tem por objetivo explorar aspectos críticos dos requisitos de um produto, simulando de forma rápida um pequeno subconjunto da sua funcionalidade. (PAULA FILHO, 2009, p. 171).

A Figura 6 ilustra um exemplo de protótipo de tela, demonstrando além das funcionalidades da tela principal, as funcionalidades derivadas.

Na prototipação, não se deve perder muito tempo, reiterando o que Paula Filho (2009) diz, não há a necessidade de desenvolvimento de telas muito trabalhadas, com alta resolução, podendo inclusive ser feitos com papel e caneta.

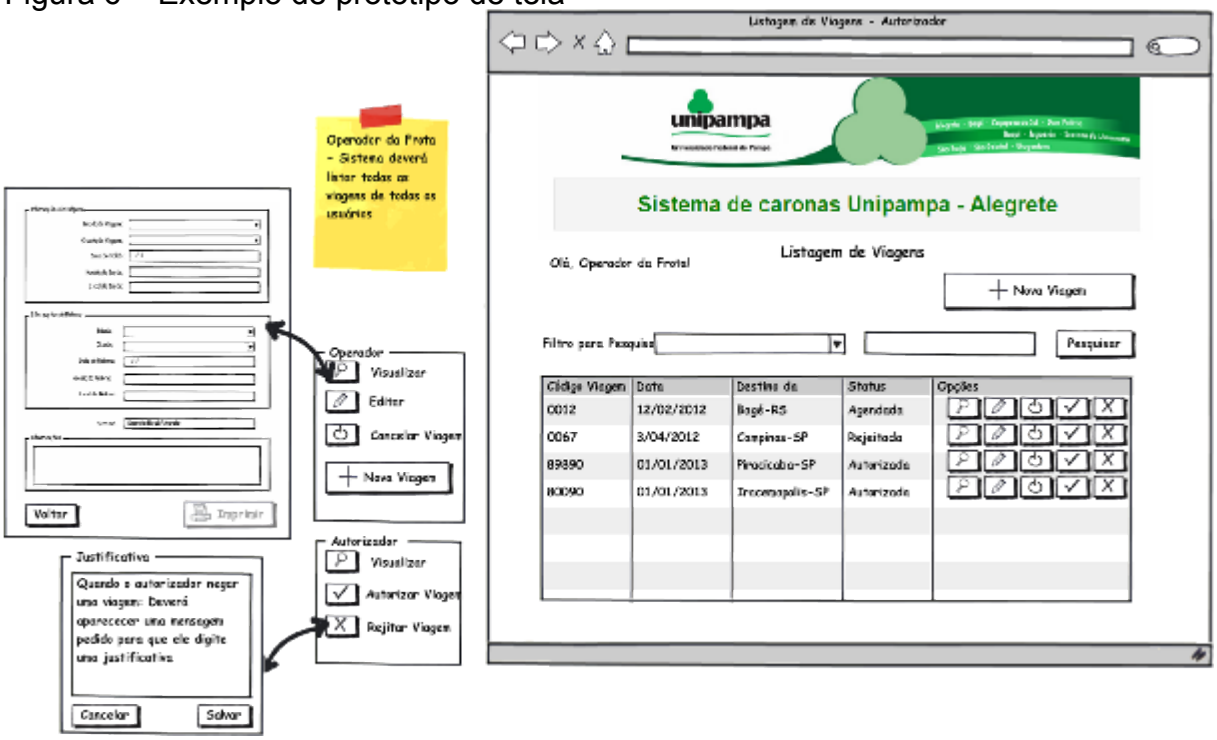
Protótipos também podem ser criados por planilhas, editores de texto e etc.

2.3.2.5 Diagramas de contexto

Muitas vezes uma imagem simples pode expressar melhor um problema do que um texto grande e complexo. Os diagramas de contexto conseguem mostrar de

forma rápida as principais funcionalidades do sistema e as interações entre o sistema e os *stakeholders*.

Figura 6 – Exemplo de protótipo de tela



Fonte: Elaborado pelo autor

Existe muita contradição nas literaturas estudadas quanto ao conceito de diagramas de contexto e casos de uso.

Sommerville (2011) cita que os casos de uso identificam as interações individuais entre o sistema e seus usuários ou outros sistemas.

Já (COCKBURN, 2005, p. 21) afirma que casos de uso “são fundamentalmente uma forma textual, embora possam ser escritos usando fluxogramas, diagramas de sequência, redes de Petri ou linguagens de programação”.

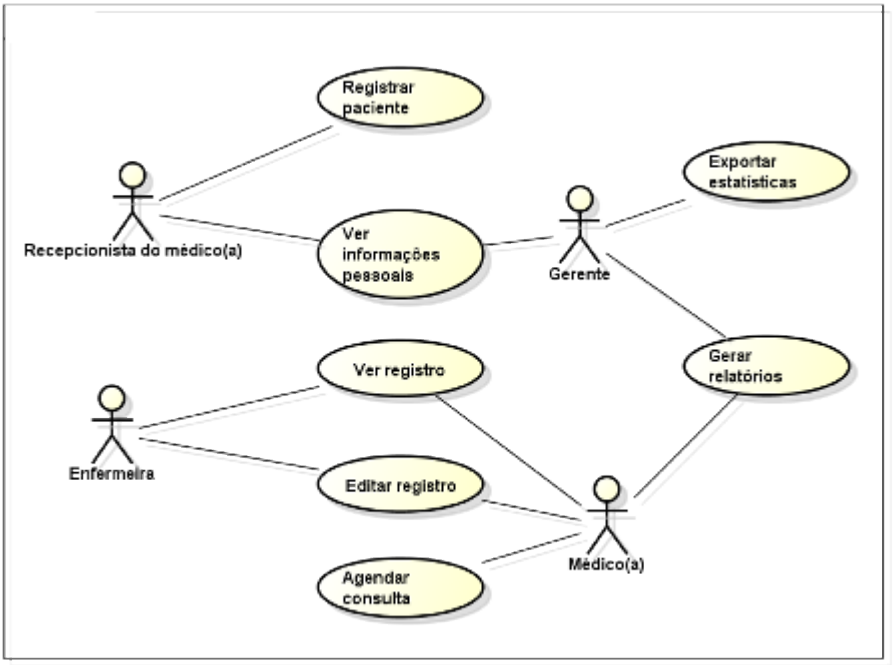
Além dessa contradição, existe outro ponto a discutir: usa-se caso de uso no contexto de elicitação e análise ou em especificação de software?

Neste trabalho, utilizaremos o termo diagramas de contexto para definir os diagramas utilizados na elicitação e análise de software, e o termo caso de uso para a especificação de software.

Os diagramas de contexto são uma ferramenta que auxilia no processo de elicitação, visto que se consegue abstrair muitas funcionalidades, com uma imagem

simples. A Figura 7 mostra um exemplo de diagrama de contexto. Porém esses diagramas são úteis para visualizar quais as funções que o sistema deve contemplar e quais os seus devidos *stakeholders* (atores), porém não como cada uma delas funciona.

Figura 7 – Exemplo diagrama de contexto



Fonte: (SOMMERVILLE, 2011, p. 75)

2.3.2.6 Cenários

Dependendo do problema, *stakeholders* podem preferir descrever quais são os processos executados de forma textual. (KOSCIANSKI; SOARES, 2007, p. 183) diz que “descrever tais situações permite que o cliente se lembre de detalhes que podem não ser revelados em questionários”.

Esses cenários podem ser escritos na forma de casos de uso ou de estórias de usuário da *eXtreme Programming*[2] (XP). A partir dos textos redigidos pelos *stakeholders*, os analistas podem compreender de melhor maneira as funcionalidades do sistema.

Sommerville (2011) descreve, na Tabela 2 um cenário referente ao caso de uso Agendar Consulta, que está representado na Figura 7.

[2] eXtreme Programming(XP) – modelo ágil de desenvolvimento de software

Tabela 2 – Cenário de uso

Agendar consulta permite que dois ou mais médicos de consultórios diferentes possam ler o mesmo registro o mesmo tempo. Um médico deve escolher, em um menu de lista de médicos on-line, as pessoas envolvidas. O prontuário do paciente é então exibido em suas telas, mas apenas o primeiro médico pode editar o registro. Além disso, uma janela de mensagens de texto é criada para ajudar a coordenar as ações. Supõe-se que uma conferência telefônica para comunicação por voz será estabelecida separadamente.

Fonte: (SOMMERVILLE, 2011, p. 75)

2.3.3 Especificação de Requisitos

Sommerville (2011) diz que o documento de requisitos, também conhecido como Especificação de Requisitos de Software (SRS – do inglês *Software Requirements Specification*), é uma declaração oficial do que os desenvolvedores do sistema devem implementar.

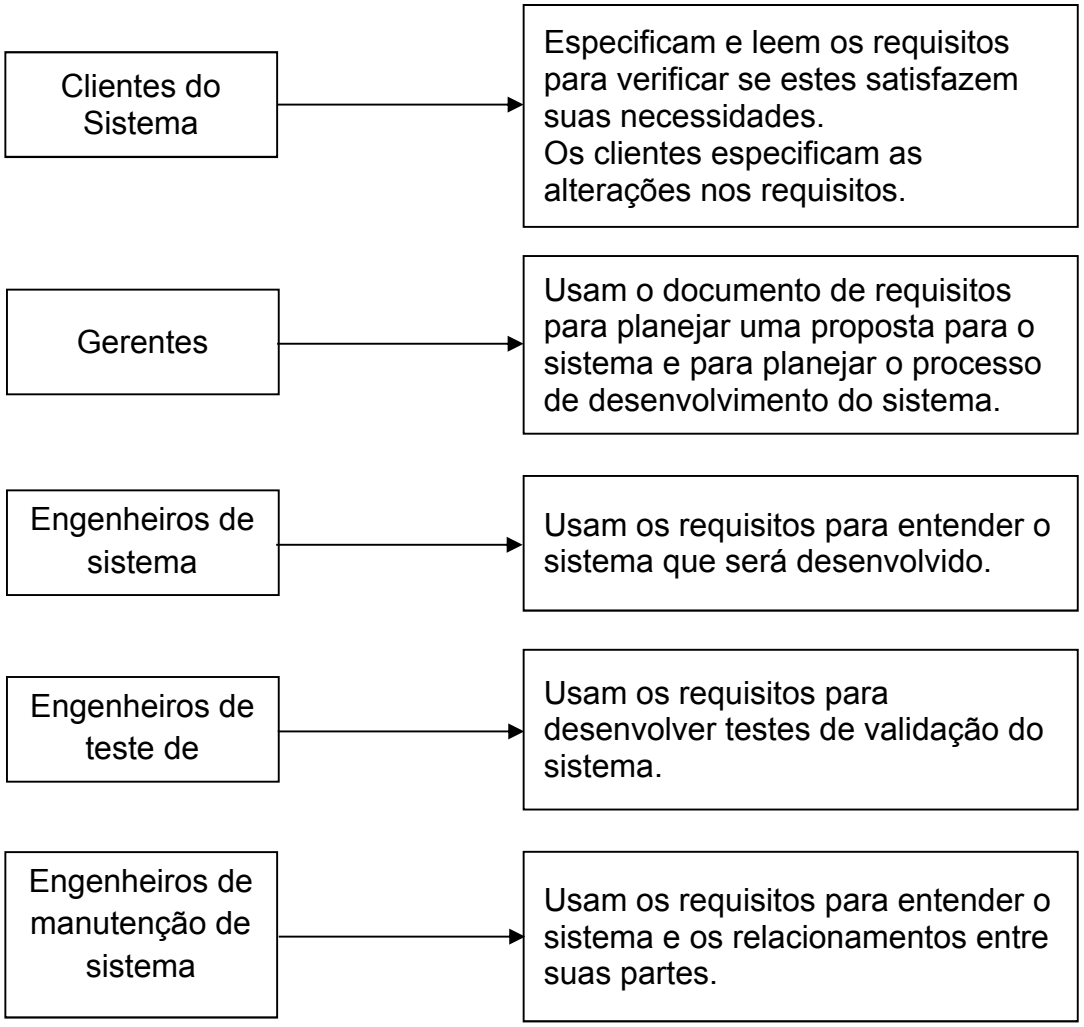
Pfleeger (2004), cita que a documentação dos requisitos é independente dos métodos escolhidos para defini-los.

Em suma, podemos dizer que o documento de requisitos é o contrato entre os envolvidos, onde é registrado o que o projeto consiste, quais os objetivos, qual é o escopo do desenvolvimento. Esse documento deve ser escrito de forma que todos os interessados no projeto tenham capacidade de ler e interpretá-lo.

Sommerville (2011) cita inclusive que devido à diversidade de usuários interessados, o documento tem diferentes visões, a Figura 8, demonstra a visão de cada usuário do mesmo.

A estrutura do documento pode variar, dependendo do projeto, da metodologia de desenvolvimento adotada e do tipo de software, porém, Sommerville (2011), especifica um modelo de estrutura baseado em uma norma da IEEE para documentos de requisitos (IEEE, 1998), conforme a Tabela 3.

Figura 8 – Usuários de um documento de requisitos



Fonte: (SOMMERVILLE, 2011, p. 64)

Tabela 3 – Estrutura de um documento de requisitos

Capítulo	Descrição
a) Prefácio	Deve definir os possíveis leitores do documento e descrever seu histórico de versões, incluindo uma justificativa para a criação de uma nova versão e um resumo das mudanças feitas em cada versão.
b) Introdução	Deve descrever a necessidade para o sistema. Deve descrever brevemente as funções do sistema e explicar como ele vai funcionar com outros sistemas. Também deve descrever como o sistema atende aos objetivos globais de negócio ou estratégicos da organização que encomendou o software.
c) Glossário	Deve definir os termos técnicos usados no documento. Você não

	deve fazer suposições sobre a experiência ou conhecimento do leitor.
d) Definição dos requisitos de usuário	Deve descrever os serviços fornecidos ao usuário. Os requisitos não funcionais de sistema também devem ser descritos nessa seção. Essa descrição pode usar a linguagem natural, diagramas ou outras notações compreensíveis para os clientes. Normas de produto e processos que devem ser seguidos devem ser especificadas.
e) Arquitetura do sistema	Deve apresentar uma visão geral em alto nível da arquitetura do sistema previsto, mostrando a distribuição de funções entre os módulos do sistema. Componentes de arquitetura que são reusados devem ser destacados.
f) Especificação de requisitos do sistema	Deve descrever em detalhes os requisitos funcionais e não funcionais. Se necessário, também podem ser adicionados mais detalhes aos requisitos não funcionais. Interfaces com outros sistemas podem ser definidas.
g) Modelos do sistema	Podem incluir modelos gráficos do sistema que mostram os relacionamentos entre os componentes do sistema, o sistema e seu ambiente. Exemplos de possíveis modelos são modelos de objetos, modelos de fluxo de dados ou modelos semânticos de dados.
h) Evolução do sistema	Deve descrever os pressupostos fundamentais em que o sistema se baseia, bem como quaisquer mudanças previstas, em decorrência da evolução de hardware, de mudanças nas necessidades do usuário etc. Essa seção é útil para projetistas de sistema, pois pode ajudá-los a evitar decisões capazes de restringir possíveis mudanças futuras no sistema.
Apêndices	Deve fornecer informações detalhadas e específicas relacionadas à aplicação em desenvolvimento, além de descrições de hardware e banco de dados, por exemplo. Os requisitos de hardware definem as configurações mínimas ideais para o sistema. Requisitos de banco de dados definem a organização lógica dos dados usados pelo sistema e os

	relacionamentos entre esses dados.
Índices	Vários índices podem ser incluídos no documento. Pode haver, além de um índice alfabético normal, um índice de diagramas, de funções, entre outros pertinentes.

Fonte: (SOMMERVILLE, 2011, p. 65)

Já Pressman (2011, p. 129) fala que a “formalidade e o formato de uma especificação variam com o tamanho e a complexidade do software a ser construído”. Cita inclusive um modelo de SRS completo que pode seguir o exemplo da Tabela 4.

Assim sendo, o mais importante ao definir um modelo de SRS, é identificar quais os artefatos que realmente agregam valor ao documento e descrevê-los de forma consistente e compreensível a todos os envolvidos.

Tabela 4 – Modelo SRS

Sumário	
Histórico de revisão	
1.	**Introdução**
1.1	Propósito
1.2	Convenções do documento
1.3	Público-alvo e sugestões de leitura
1.4	Escopo do projeto
1.5	Referências
2.	**Descrição geral**
2.1	Perspectiva do produto
2.2	Características do produto
2.3	Classes de usuários e características
2.4	Ambiente operacional
2.5	Restrições de projeto e implementação
2.6	Documentação para usuários
2.7	Hipóteses e dependências
3.	**Características do sistema**
3.1	Características do sistema 1
3.2	Características do sistema 2 (e assim por diante)
4.	**Requisitos de interfaces externas**

4.1 Interfaces do usuário
4.2 Interfaces de hardware
4.3 Interfaces de software
4.4 Interfaces de comunicação
5. Outros requisitos não funcionais
5.1 Necessidades de desempenho
5.2 Necessidades de proteção
5.3 Necessidades de segurança
5.4 Atributos de qualidade de software
6. Outros requisitos
Apêndice A: Glossário
Apêndice B: Modelos de análise
Apêndice C: Lista de problemas

Fonte: Adaptado de (PRESSMAN, 2011, p. 129)

O grande valor em um SRS são os requisitos propriamente ditos e, principalmente, como são redigidos. Lembre-se que para que agregue valor, o requisito deve ser descrito de forma que as pessoas que farão uso do documento consigam ter um claro e conciso entendimento do problema. Cabe ressaltar ainda que a sequência do projeto levará em conta exatamente o que for entendido do documento, e caso existam ambiguidades e inconsistências, certamente o trabalho subsequente terá um futuro pouco promissor.

Na sequência, serão discutidas algumas formas de registrar os requisitos.

2.3.3.1 Casos de uso

Conforme explicado nas seções 2.3.2.5 e 2.3.2.6, casos de uso é uma maneira bem eficaz de registrar os requisitos. Os diagramas de contexto muito auxiliam na abstração do problema, e os cenários mostram um pouco da visão por parte do usuário. Porém, em ambos os casos, não se consegue ter uma idéia completa, necessitando-se de uma escrita mais precisa e formal para garantir a consistência do problema.

É nesse ponto que entram os casos de uso, que segundo (COCKBURN, 2005, p. 29), casos de uso corretamente descritos, especificam exatamente o que o sistema deve fazer.

Tomando como o exemplo a Figura 7, o caso de uso Agendar Consulta possui inclusive um cenário descrito (Tabela 2), porém, para fins de projeto, esse cenário se torna incompleto. A Tabela 5 mostra uma maneira mais completa de descrever esse caso de uso.

Tabela 5 – Exemplo de caso de uso

CASO DE USO 01: AGENDAR CONSULTA
Ator primário: Médico
Escopo: Agendar consulta
Nível: Objetivo do usuário
***Stakeholders* e interesses:**
Médicos: analisar o prontuário de um paciente e discutir sobre o mesmo;
Pré-condição: paciente já cadastrado no sistema;
Cenário de Sucesso principal:
1. O médico abre o prontuário do paciente;
2. O sistema mostra o prontuário do paciente na tela;
3. O sistema mostra uma lista de médicos on-line;
4. O médico seleciona os demais envolvidos na lista de médicos;
5. O sistema abre o prontuário, do paciente selecionado, para a tela dos outros médicos envolvidos, como somente-leitura;
6. O sistema abre uma janela de mensagem de texto em cada tela dos envolvidos;
Extensões:
2.a O médico deseja editar o prontuário do paciente;
2.a.1. O médico selecionará a opção de Editar Prontuário;
2.a.2. O médico irá editar o prontuário;
2.a.3. O médico selecionará a opção para gravar as alterações;
3.a Não existem outros médicos on-line;
3.a.1. No local da lista de médicos, o sistema mostra a mensagem “Nenhum médico on-line”;

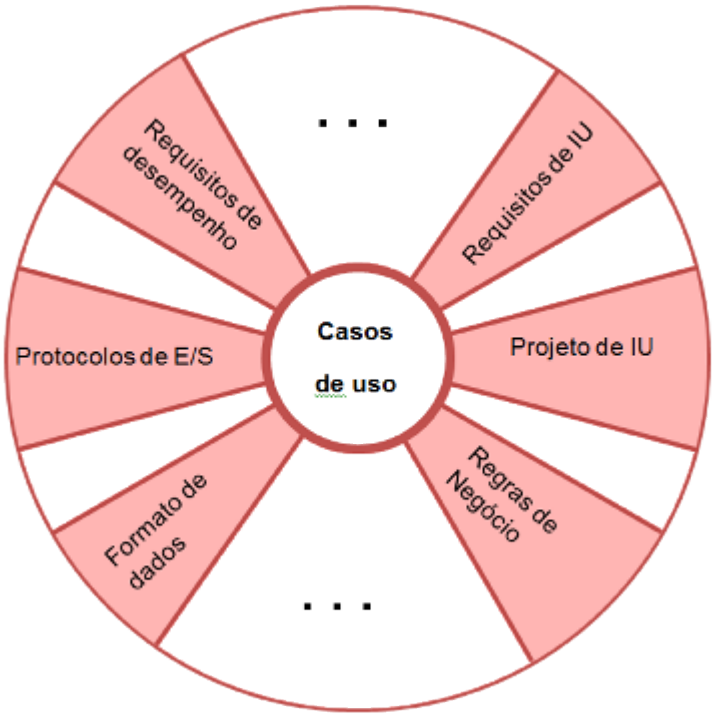
Fonte: Elaborada pelo autor

Descrevendo o caso de uso, pode-se identificar melhor a ordem de execução das tarefas e as possíveis exceções. Também se pode realizar a ligação entre casos de uso, basta olhar na Tabela 5, que existe um momento em que o

termo “Editar Prontuário” encontra-se sublinhado, isso indica que naquele momento será executado outro caso de uso, descrito separadamente.

Apesar de tudo isso, a utilização de casos de uso não consegue especificar todos os requisitos de sistema, segundo (COCKBURN, 2005, p. 29), “casos de uso não especificam interfaces externas, formatos de dados, regras de negócio e fórmulas complexas, porém, todos esses requisitos estarão ligados aos casos de uso”, conforme Figura 9. Para esses requisitos, existem outras formas de especificação, que serão mostradas mais adiante.

Figura 9 – Modelo de requisitos “eixo-e-raios”



Fonte: Adaptado de (COCKBURN, 2005, p. 32)

2.3.4 Validação de Requisitos

Para garantir se os requisitos definem o que o cliente necessita, se faz necessário cumprir algumas premissas que se justificam para o bom entendimento do problema pelas pessoas envolvidas no projeto. Na etapa de validação, são verificadas algumas premissas para garantir a qualidade dos requisitos. Sommerville (2011, p. 77), cita os itens que diferentes tipos de verificações devem ser efetuados com o documento de requisitos. São eles:

a) **Validade:** verificar se as funcionalidades registradas realmente atenderão todas as necessidades;

b) **Consistência:** O requisito não pode ser ambíguo, e não pode entrar em conflito com outro requisito. Por exemplo: Um requisito restringe o acesso ao sistema apenas após a identificação do usuário através de cartão com chip. Outro requisito exige que o sistema possa ser acessado através de dispositivos móveis, tipo smartphone.

c) **Completude**: verificar se os requisitos definem todas as funções pretendidas pelo cliente

d) **Corretude:** O requisito realmente descreve a funcionalidade exigida, garantindo a possibilidade de implementar, projetar e testar;

e) **Verificabilidade:** Deve haver a possibilidade de testar se o requisito foi realmente implementado de forma correta. Por exemplo, desejamos que a liberação de dinheiro em um caixa eletrônico seja efetuada em até 5 segundos.

Além das verificações citadas por Sommerville, (KOSCIANSKI; SOARES, 2007, p. 184-185) acrescentam as abaixo:

f) **Precisão:** precisa garantir que o requisito tenha apenas uma única interpretação;

g) **Priorizável:** Para a facilitação da gestão dos requisitos, se faz necessário priorizá-los em virtude de uma possível limitação de tempo, os mais importantes deverão ter prioridade de implementação. Paula Filho (2009) cita que os requisitos podem ser classificados de três formas: essencial, cuja falta de atendimento ao requisito é inaceitável, desejável, requisito que agrega grande valor ao produto e, opcional, cujo impacto final no produto não é tão relevante e deverá ser atendido apenas caso a disponibilidade de tempo comporte.

h) **Modificável:** a especificação deve garantir a sua modificabilidade, garantindo a consistência e completude, de maneira fácil;

i) **Rastreabilidade:** Muitos requisitos se originam de outros requisitos e assim por diante, sendo assim, ao alterar o primeiro, é necessário verificar quais os outros requisitos também poderão ser afetados.

Completando as validações, (PFLEEGER, 2004, p. 119) acrescenta:

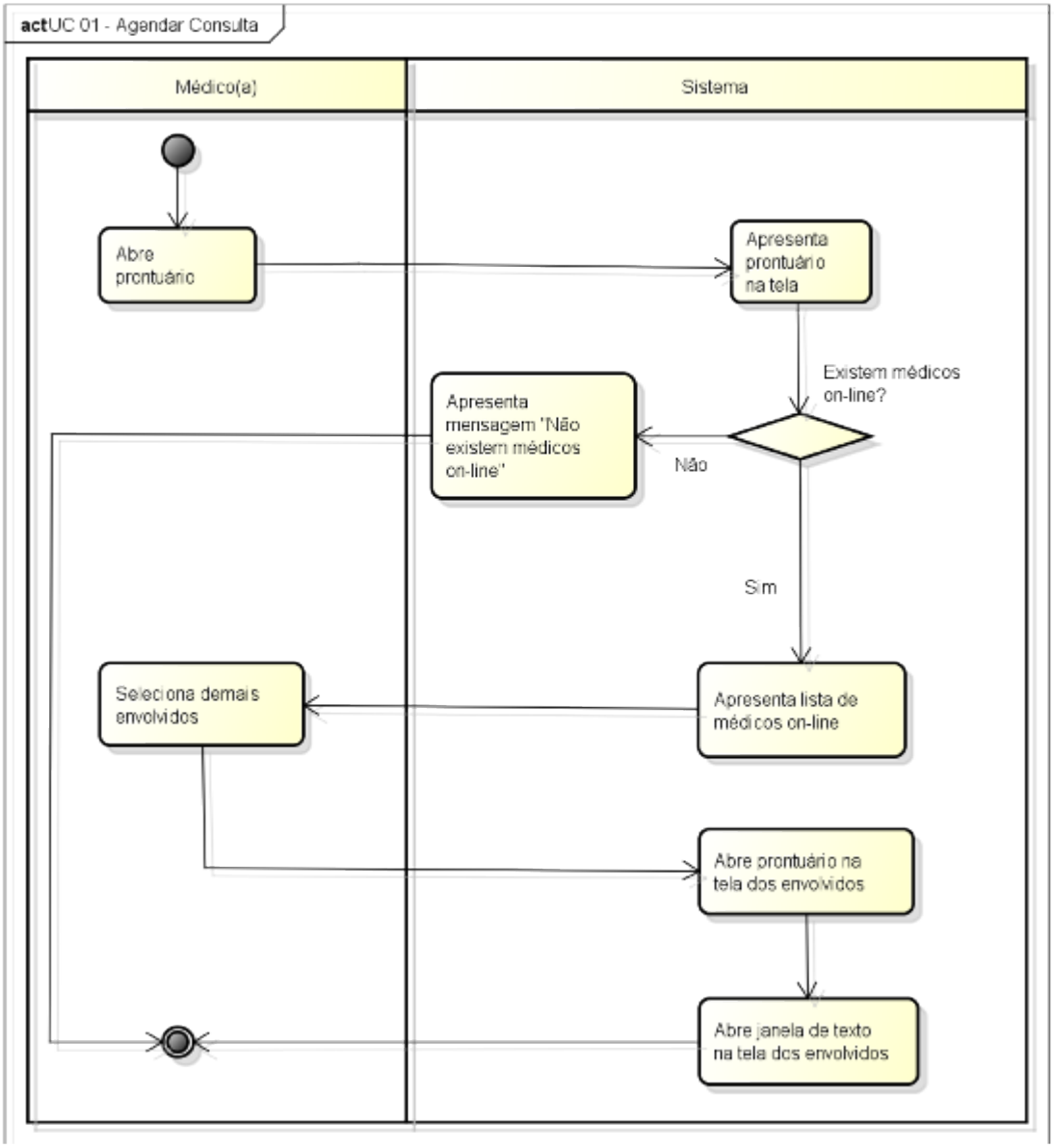
j) **Realista:** Deve ser implementável computacionalmente, a partir das limitações exigidas tanto de hardware, quanto do ambiente de desenvolvimento;

k) **Necessário:** Verificar a real necessidade do requisito, ou seja, o requisito deve realmente ser de utilidade para o usuário, direta ou indiretamente;

2.3.4.1 Fluxo de eventos

Por vezes, descrever casos de uso se torna algo demorado e de difícil manutenção, além de tornar a leitura cansativa. A descrição desses casos de uso em forma de fluxo de eventos permite uma visualização mais rápida, principalmente quando os casos de uso são muito complexos e apresentam muitos fluxos alternativos. Paula Filho (2009, p. 183) cita que esses diagramas “representam um ponto de equilíbrio entre notações mais formais e as descrições textuais”. Para representar esses fluxos, é utilizado o diagrama de atividades, proposto pela UML. A Figura 10 mostra o exemplo de caso de uso descrito na Tabela 5, no formato de diagrama de atividades.

Figura 10 – Diagrama de Atividade



Fonte: Elaborada pelo autor

2.3.4.2 Business Process Model and Notation - BPMN

Apesar de não ter foco em especificação de requisitos, o padrão BPMN é uma excelente ferramenta para a modelagem de processos de negócio. Semelhante ao diagrama de atividades (apresentado no capítulo 2.3.4.1), porém, com foco em processos.

> O BPMN é um padrão para a modelagem de processos. Criado inicialmente como uma evolução das experiências anteriores pelo BPMI (Business Process Management Initiative), foi incorporado pela OMG (Object Management Group), após a fusão entre essas entidades, ocorrida em 2005. Trata-se de uma técnica especialmente voltada para a definição e documentação de processos de negócio com padrões de notação bem definidos. (VALLE; OLIVEIRA, 2009, p. 53)

Baldmam (2007) cita que uma das grandes vantagens do uso do padrão BPMN é que o mesmo oferece uma notação facilmente compreendida e usada por todos os envolvidos nos processos de negócio, ou seja, pela sua simplicidade, se torna fácil o seu entendimento, mesmo por pessoas que não estão acostumadas com os formalismos das técnicas de especificação de software.

O BPMN propõe as seguintes definições:

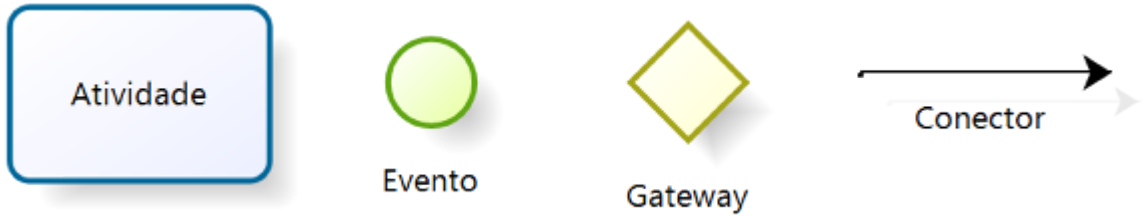
Tabela 6 – Conceitos básicos segundo o BPMN

CONCEITO	DEFINIÇÃO BPMN
Atividade	Termo genérico para o trabalho desempenhado pela empresa. Processos, subprocessos e tarefas são tipos de atividades.
Tarefa	Tarefa (*task*) é uma atividade atômica incluída num processo. No modelo de processos, a tarefa é o desdobramento máximo do trabalho executado no processo.
Processo	Qualquer atividade desempenhada no interior da organização. No modelo de processos, é retratada como uma rede constituída por outras atividades em fluxo e por seus respectivos controles de sequenciamento (eventos e junções). Um processo de negócio contém um ou mais processos.
Evento	Algo que “acontece” no curso do processo de negócio, influenciando o seu fluxo. Há o evento inicial, o evento final e eventos intermediários.

Fonte: (VALLE; OLIVEIRA, 2009, p. 9)

O padrão BPMN é representado pelo Diagrama de Processos de Negócio (DPN), que possui basicamente quatro tipos de elementos, representados pela Figura 11.

Figura 11 – Elementos BPMN



Fonte: Adaptado de (VALLE; OLIVEIRA, 2009, p. 81)

Porém, cada um desses elementos pode possuir algumas variações, tais como:

Atividade: pode representar uma tarefa, um processo ou um subprocesso, além de haver a possiblidade de uma especificação mais ilustrada, como tarefas de envio e recepção. Ver Figura 12.

Figura 12 – Exemplos de representação de tarefas

Fonte: o autor

Evento: tem basicamente três representações, início, intermediário e fim, que se diferenciam pelas suas bordas, borda fina, borda dupla e borda grossa respectivamente, a Figura 13 ilustra exemplos de representação de eventos.

Figura 13 – Exemplos de representação de eventos

Fonte: o autor

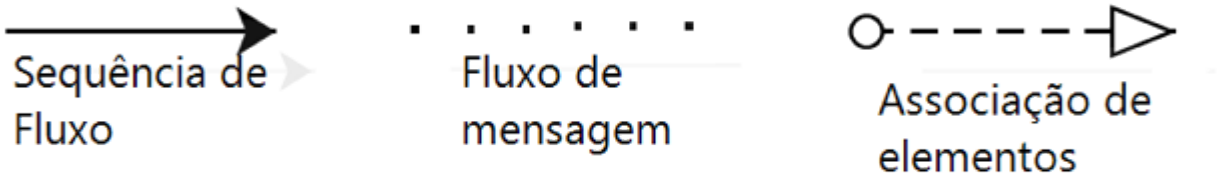
Gateways: são utilizados para controlar a sequência de fluxo dos processos, podendo ser convergentes ou divergentes, ou seja, tem a finalidade de unir ou separar o fluxo dos processos. Também podem ser representados conforme o tipo de decisão, onde o gateway exclusivo garante que após uma decisão, apenas uma das condições será válida, seguindo apenas um fluxo na sequência. O gateway paralelo indica que a partir daquele ponto, as tarefas subsequentes serão executadas de forma paralela. A Figura 14 ilustra a representação dos gateways citados.

Figura 14 – Exemplo de representação de gateways

Fonte: o autor

Conectores: servem para realizar as ligações necessárias para garantir a sequência de fluxo dos processos, além de realizar ligações entre tarefas e mensagens, ou também, associação de elementos.

Figura 15 – Exemplo de representação de conectores



Fonte: adaptado de (VALLE; OLIVEIRA, 2009, p. 88)

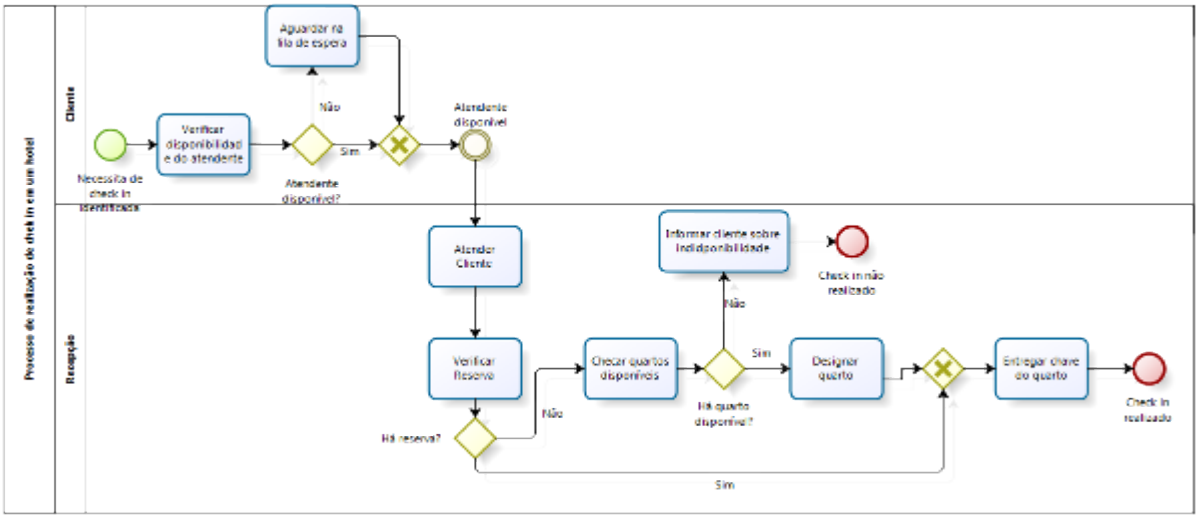
Os diagramas BPMN também podem fazer uso de *Swimlanes* (raias), que, segundo (VALLE; OLIVEIRA, 2009, p. 89) pode ser de dois tipos: *Pools* (piscinas) e *Lane* (raia). Onde *Pools* representam entidades que estão separadas fisicamente no diagrama, e *Lane* para separar atividades associadas para uma função ou papel específico. A Figura 16 representa um exemplo de *Swimlane*.

Figura 16 – Exemplo de *swimlane*

Fonte: adaptado de (VALLE; OLIVEIRA, 2009, p. 89)

A Figura 17 ilustra um exemplo de processo de negócio de um *Check in* de hotel.

Figura 17 – Diagrama de processo de negócio (exemplo)



Fonte: adaptado de (VALLE; OLIVEIRA, 2009, p. 92)

2.3.4.3 Estórias de usuários

Alguns autores citam o termo “estória”, e outros usam o termo “história”. Nesse trabalho utilizaremos o termo “estória”. Semelhante aos cenários, o uso de estória de usuários é uma prática utilizada em métodos ágeis, mais precisamente na XP. As funcionalidades são descritas em alto nível, evitando o uso de termos técnicos. No livro de (SOMMERVILLE, 2011, p. 45) é apresentado um exemplo de estória, descrevendo uma prescrição médica.

Tabela 7 – Estória de prescrição de medicamentos

Prescrição de medicamentos
Kate é uma médica que deseja prescrever medicamento para um paciente de uma clínica. O prontuário do paciente já está sendo exibido em seu computador, assim, ela clica o campo “medicação” e pode selecionar “medicação atual”, “nova medicação”, ou “formulário”. Se ela selecionar “medicação atual”, o sistema pede que ela verifique a dose. Se ela quiser mudar a dose, ela altera esta e em seguida, confirma a prescrição. Se ela escolher “nova medicação”, o sistema assume que ela sabe qual medicação receitar. Ela digita as primeiras letras do nome do medicamento. O sistema exibe uma lista

de possíveis fármacos que começam com essas letras. Ela escolhe a medicação requerida e o sistema responde, pedindo-lhe para verificar se o medicamento selecionado está correto. Ela insere a dose e, em seguida, confirma a prescrição. O sistema sempre verifica se a dose está dentro da faixa permitida. Caso não esteja, Kate é convidada a alterar a dose. Após Kate confirmar a prescrição, esta será exibida para verificação. Ela pode escolher “OK” ou “Alterar”. Se clicar em “OK”, a prescrição fica gravada nos bancos de dados da auditoria. Se ela clicar em “Alterar”, reinicia o processo de “Prescrição de Medicamentos”.

Fonte: (SOMMERVILLE, 2011, p. 45)

Como visto no exemplo da Tabela 7, essa estória pode ser dividida em várias tarefas. Essas tarefas serão escritas em cartões e esses, por sua vez, são distribuídos aos programadores, que se encarregam de implementar as funcionalidades. Koscianski & Soares (2007) ilustra um exemplo de como podem ser representados as estórias de usuário e os cartões (Tabela 8 e Tabela 9).

Tabela 8 – Um padrão de estória de usuário

Data:	Tipo de estória: Nova Melhoria Debug		
Número:	Prioridade:	Risco:	Estimativa:
Descrição:			
Notas:			
Acompanhamento:			
Data	Status	A fazer	Comentário

Fonte: (KOSCIANSKI; SOARES, 2007, p. 199)

Tabela 9 – Um padrão de cartão de tarefas

Data:	Número:	Estimativa:
Responsável 1:		Responsável 2:
Descrição:		
Notas:		
Acompanhamento:		

Data	Status	A fazer	Comentário

Fonte: (KOSCIANSKI; SOARES, 2007, p. 199)

2.3.5 Gerência de Requisitos

> O propósito do processo Gerência de Requisitos é gerenciar os requisitos do produto e dos componentes do produto do projeto e identificar inconsistências entre os requisitos, os planos do projeto e os produtos de trabalho do projeto. (MPS.BR, 2011, p. 29)

Os requisitos de um projeto mudam constantemente, principalmente em projetos mais longos. No transcorrer do tempo, os *stakeholders* podem mudar a sua visão sobre um determinado problema, alguma funcionalidade deixa de ter importância e outras podem surgir ao longo do caminho. Mudanças de tecnologias, tanto de *software*, quanto de *hardware* também podem exigir mudanças nos requisitos. Além de mudanças de legislação e outras mais.

Sommerville (2011, p. 78) cita que o “planejamento é o primeiro estágio essencial no processo de gerenciamento de requisitos”. Cita inclusive que é necessário definir algumas premissas quanto ao seu detalhamento, são elas:

Identificação de requisitos: os requisitos devem ser identificados de forma única, ou seja, procurar identificá-los de forma sistemática para, caso haja alguma alteração, se consiga identificar exatamente o que deve ser alterado.

Processo de gerenciamento de mudanças: é o conjunto de tarefas que devem ser observadas visando o controle do impacto advindo das alterações no projeto, identificando custos e prazo.

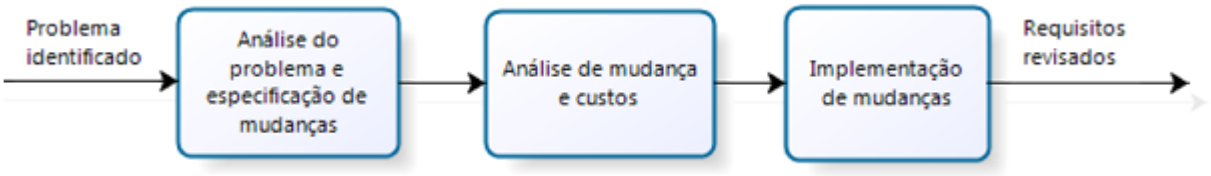
Políticas de rastreabilidade: identificar quais os requisitos que estão direta ou indiretamente ligados ao requisito alterado.

Ferramentas de apoio: Identificar as ferramentas que auxiliarão no gerenciamento de requisitos. Sommerville (2011, p. 78) cita que “o gerenciamento

de requisitos precisa de apoio automatizado, e as ferramentas de *software* para esse gerenciamento podem ser escolhidas durante a fase de planejamento".

A Figura 18 mostra um processo que pode ser utilizado caso haja a necessidade de alteração de algum requisito.

Figura 18 – Gerenciamento de mudança de requisitos



Fonte: Adaptado de (SOMMERVILLE, 2011, p. 79)

2.4 Métricas

Mudanças em processos de software podem causar falsas impressões quanto à sua real eficiência. Visto que muitas vezes empresas utilizam técnicas *ad hoc* de desenvolvimento, copiam técnicas de outras empresas ou apenas seguem as novas tendências de mercado, fica difícil para um gerente de projeto de desenvolvimento de software conseguir mensurar se essas mudanças realmente surtiram o efeito esperado.

Para que seja possível essa mensuração de forma quantitativa, se faz necessário utilizar alguma técnica eficiente e eficaz, que garanta que essas impressões sejam ratificadas.

Sommerville (2011) cita que existem três tipos de métricas de processo que podem ser coletadas:

a) O tempo necessário para um processo específico ser concluído;
b) Os recursos necessários para um determinado processo;
c) O número de ocorrências de um determinado evento.

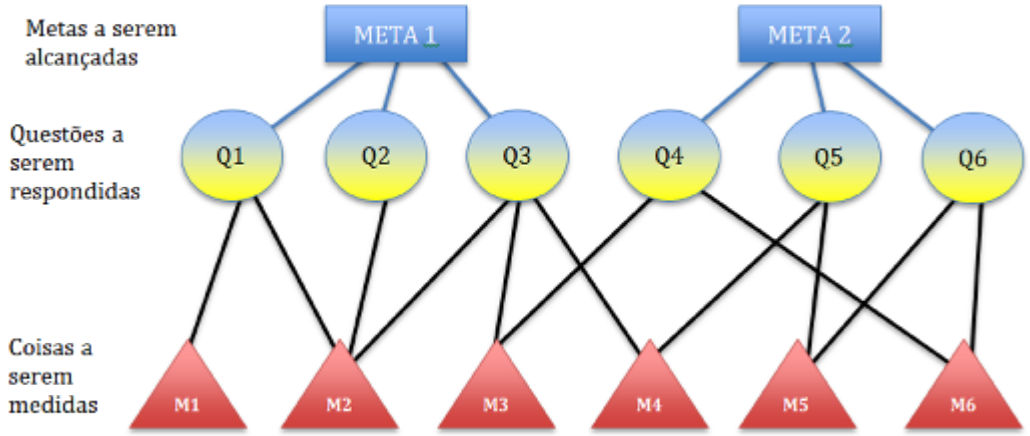
Como o objetivo deste trabalho é uma mudança em processos de trabalho, para que se avalie a eficiência e eficácia desse processo, se faz valer o estudo de metodologias de medição, cuja escolhida é o paradigma GQM, ao qual é descrito no próximo capítulo.

2.4.1 Paradigma GQM

A grande dificuldade do processo de medição é determinar o que deve ser medido, para que os resultados obtidos por essa medição sejam um reflexo real do panorama atual dos processos. Em virtude dessas dificuldades, (BASILI; ROMBACH, 1988) propuseram o que eles chamam de paradigma GQM (Meta-Questão-Métrica, do inglês *Goal-Question-Metric*). Esse padrão de métrica vem sendo amplamente utilizado para medição de processos na área de software.

O paradigma GQM é baseado em metas, onde, em um primeiro momento são definidas metas a serem alcançadas, a partir dessas metas, são elaboradas questões que garantem o alcance dessas metas, e, por fim, são definidas as métricas necessárias para responder às questões. A definição do plano de metas é realizada de forma *top-down*, e a análise/interpretação, é *bottom-up*. A Figura 19 representa um diagrama do paradigma GQM.

Figura 19 – Paradigma GQM



Fonte: Adaptado de (SOMMERVILLE, 2011, p. 498)

Gresse (2000) cita que programas de mensuração baseados em GQM devem ser planejados e executados de acordo com os seguintes princípios:

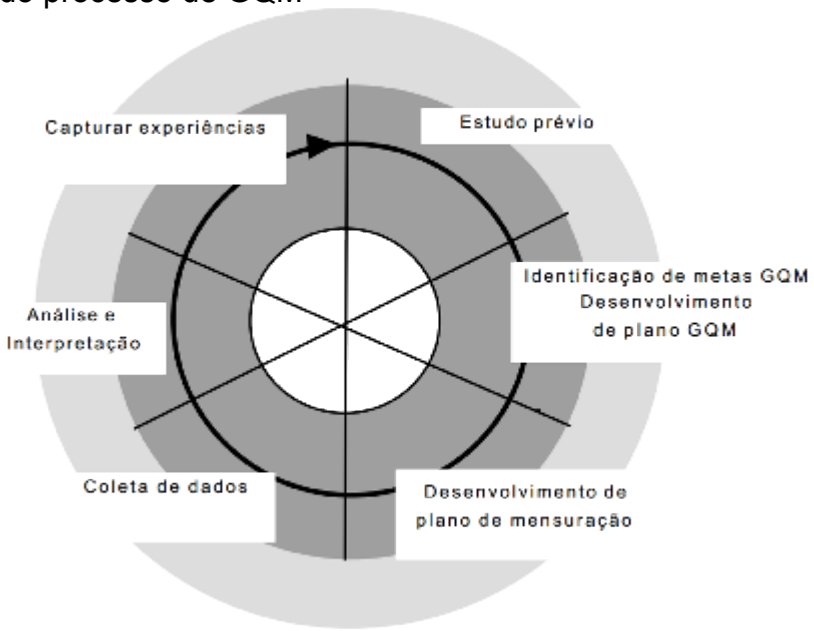
a) A tarefa de análise a ser executada precisa ser especificada precisamente e explicitamente através de uma meta de mensuração;
b) Medidas precisam ser derivadas de uma forma *top-down* baseada em metas e perguntas. Uma estrutura de metas e perguntas não pode ser adaptada de forma retroativa a um conjunto de medidas existente;
c) Cada medida precisa ter um fundamento lógico subjacente que é documentado explicitamente. O fundamento lógico é usado para justificar a coleta dos dados e para guiar a análise e interpretação;

d) Os dados que são coletados com respeito às medidas precisam ser interpretados de uma forma *bottom-up* num contexto das metas GQM e das perguntas. Isso dá suporte à interpretação dos dados para as limitações e suposições do objeto através de um fundamento lógico de cada medida;

e) As pessoas que vão utilizar os resultados do programa de mensuração e a partir de cujo ponto de vista a meta de mensuração é formulada, precisam ser envolvidas profundamente na definição e interpretação do programa de mensuração. Elas são os reais peritos com respeito ao objeto e ao enfoque de qualidade investigado no programa de mensuração e, portanto, proveem interpretações válidas no ambiente específico.

A Figura 20 ilustra os 6 (seis) passos principais do processo GQM propostos por (GRESSE; RUHE, 1999), são eles: Estudo prévio, identificação das metas e desenvolvimento do plano GQM, desenvolvimento do plano de mensuração, coleta de dados, análise e interpretação dos dados e, por fim, captura de experiências.

Figura 20 – Ciclo do processo de GQM



Fonte: (GRESSE, 2000, p. 11)

3 TRABALHOS RELACIONADOS

Neste capítulo são apresentados alguns trabalhos relacionados com a área de Engenharia de Requisitos que justificaram o assunto desse trabalho.

3.1 Metodologia para Equipes de Desenvolvimento de Requisitos de Sistemas de Informação

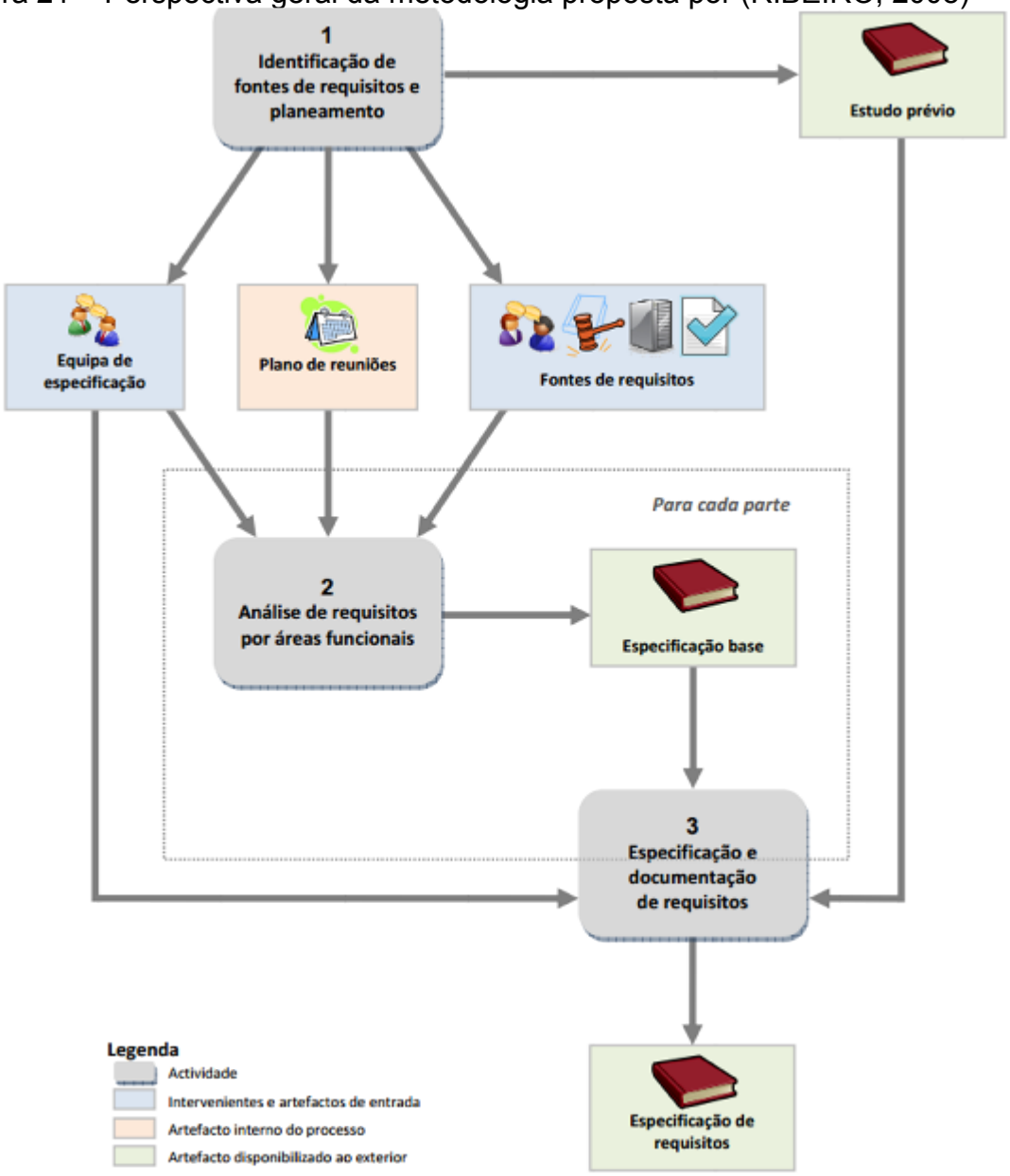
Em sua dissertação, (RIBEIRO, 2008) propõe um processo de especificação de requisitos para o Projeto de Sistemas de Informação da Faculdade de Engenharia da Universidade do Porto (FEUP).

Nesse trabalho, o autor especifica alguns pressupostos para que sua metodologia seja aplicada, tais como: alocar um conjunto de pessoas, com dedicação de algumas horas por semana, para a preparação e a realização de atividades relacionadas ao processo de desenvolvimento de requisitos, e atribuir as funções de engenheiro de requisitos a uma pessoa com dedicação exclusiva aos projetos.

Ribeiro (2008) cita que, devido ao grande tamanho dos projetos de sistemas desenvolvidos, há a necessidade de envolver várias pessoas na sua especificação, além de ser necessário envolver pessoas com diferentes perspectivas do sistema, para garantir uma visão única e consensual. Nesse ponto, o mesmo identifica que é essencial que as pessoas envolvidas no processo tenham atribuídas algumas horas, de trabalho semanal, para que participem do processo de desenvolvimento de requisitos.

A Figura 21 ilustra a perspectiva proposta por (RIBEIRO, 2008), onde se divide basicamente em três fases: Identificação de fontes de requisitos e planejamento, análise de requisitos por áreas funcionais, e por fim, a especificação e documentação dos requisitos.

Figura 21 – Perspectiva geral da metodologia proposta por (RIBEIRO, 2008)



Fonte: (RIBEIRO, 2008, p. 48)

3.2 Pesquisa da Engenharia de Requisitos em Empresas de Desenvolvimento de Software de Micro, Pequeno e Médio Porte de Joinville

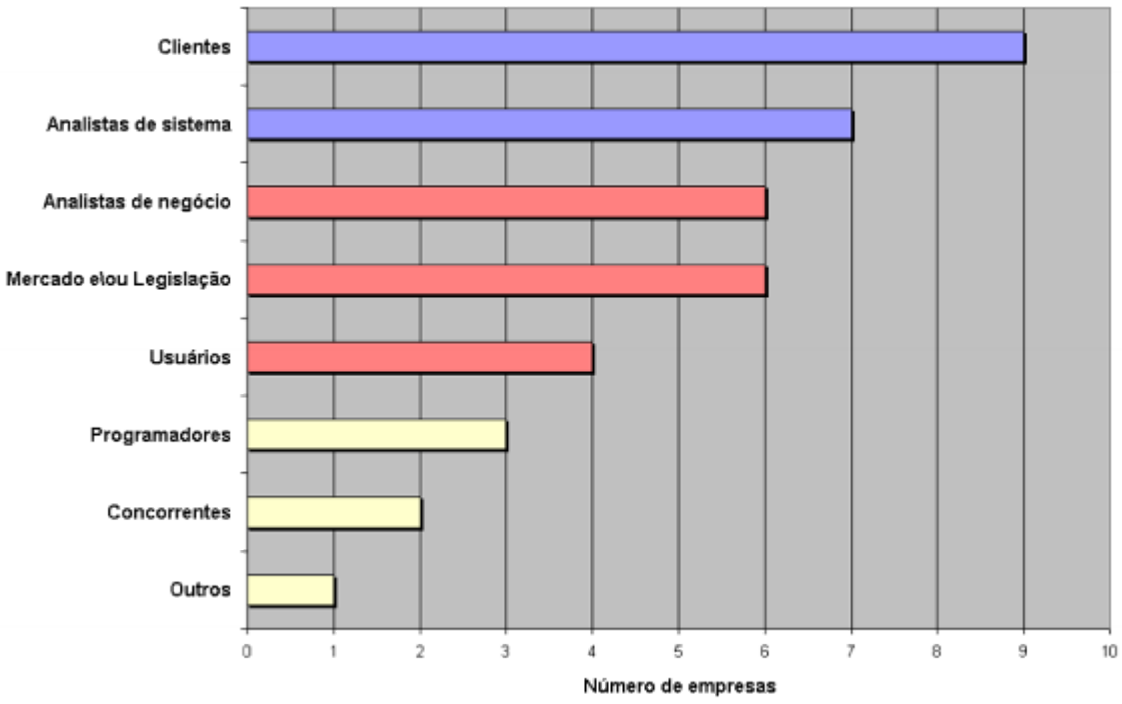
Em seu trabalho de pesquisa, (GIANESINI, 2008), realiza um levantamento bibliográfico sobre a área de engenharia de requisitos, além de descrever as boas

práticas citadas pelo *Capability Maturity Model* (CMM[3]) e *Capability Maturity Model Integration* (CMMI[4]).

De posse desses conceitos, a autora, foca a pesquisa em avaliação de micro, pequenas e médias empresas da cidade de Joinville-SC. Foram enviados questionários a 14 (quatorze) empresas, sendo que, pelos critérios adotados pela autora, apenas os resultados de 10 (dez) estavam realmente aptos a serem considerados. Apresento, logo abaixo, alguns gráficos extraídos desse trabalho que ratificam a necessidade do estudo de Engenharia de Requisitos no processo de desenvolvimento de software.

A Figura 22 apresenta uma análise dos envolvidos no levantamento de requisitos. Nota-se que em quase todas as empresas os clientes estão diretamente envolvidos nesse quesito. A autora cita também que quantidade de empresas em que os analistas de sistemas estão envolvidos no processo, está em menor número, evidenciando que nem sempre um profissional com formação técnica parta tal está diretamente ligado à definição de requisitos.

Figura 22 – Pessoas envolvidas em definição de requisitos

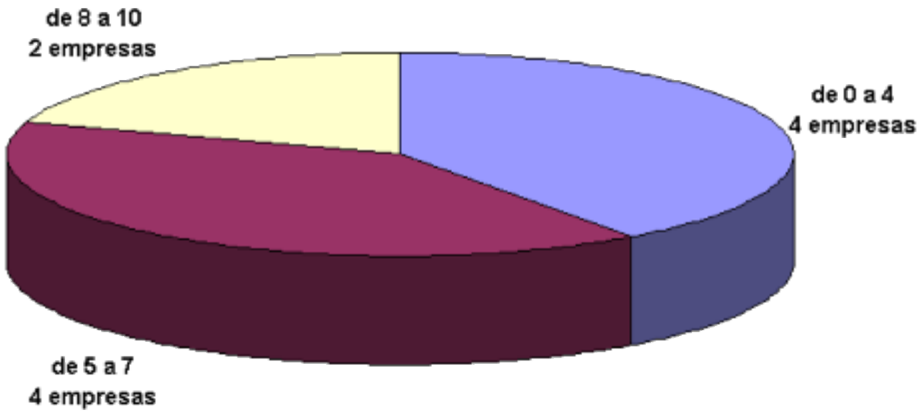


Fonte: (GIANESINI, 2008, p. 59)

[3] CMM – Modelo Maturidade e Capacidade- Desenvolvido pela *Software Engineering Institute* (SEI)
[4] CMMI – Modelo Maturidade e Capacidade Integrado – Desenvolvido pelo SEI

Quanto à padronização de processos, a autora solicitou que as empresas definissem o grau de padronização, que varia de 0(zero) a 10 (dez), onde zero indicaria sem padronização e dez a processo totalmente padronizado. Através da Figura 23 nota-se que a grande maioria não possui um processo totalmente padronizado no desenvolvimento.

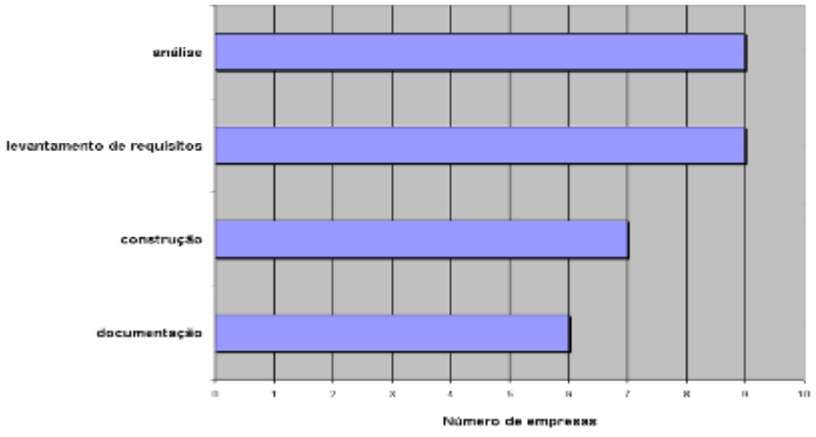
Figura 23 – Distribuição das empresas por faixa da escala de padronização dos processos de desenvolvimento



Fonte: (GIANESINI, 2008, p. 60)

Quanto à utilização de modelos de referência, foi verificado que a grande maioria das empresas utiliza modelos para os seus processos, mesmo que por vezes não é garantida a utilização em sua totalidade. A Figura 24 ilustra que esses modelos são mais utilizados nos processos de análise e de levantamento de requisitos.

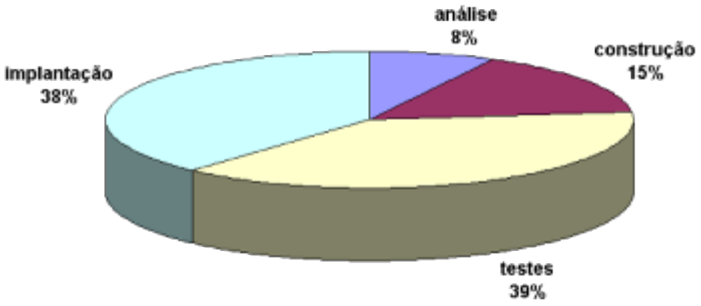
Figura 24 – Fases que utilizam modelos de referência



Fonte: (GIANESINI, 2008, p. 61)

O próximo gráfico, Figura 25, mostra um grande problema relativo a requisitos, a grande maioria das empresas não consegue identificar falhas nos requisitos no início do projeto, apenas 8% conseguem identificar na fase inicial, ou seja, na análise, e a grande maioria (77%), identifica apenas nos testes ou na implantação, fases essas em que o custo de alteração de um requisito é extremamente alto.

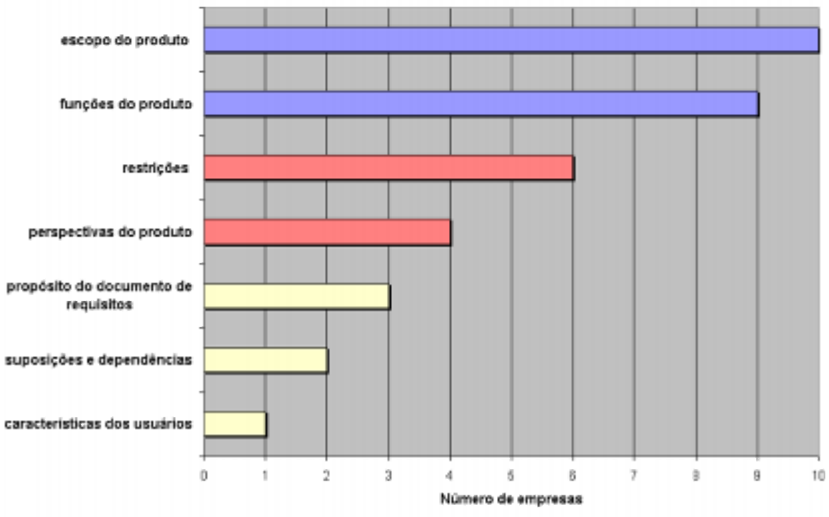
Figura 25 - Distribuição percentual das fases em que as inconsistências dos requisitos são encontradas nas empresas



Fonte: (GIANESINI, 2008, p. 62)

A Figura 26 representa a qualidade do documento de requisitos, onde 100% das empresas afirma que documenta o escopo do produto, porém, outros itens, também importantes, são deixados de lado por algumas empresas.

Figura 26 - Número de empresas por tipo de informação registrada no documento de registro de requisitos



Fonte: (GIANESINI, 2008, p. 65)

Esse trabalho auxiliou para identificar quais os itens mais sensíveis, apresentados pela autora, no contexto de engenharia de requisitos que influenciam nos resultados do desenvolvimento de software.

3.3 Uma Experiência de Engenharia de Requisitos em Empresas de Software

Semelhante ao trabalho citado no capítulo 3.2, a autora desse artigo realizou uma pesquisa com 13 empresas de Recife-PE, Brasil, com foco em engenharia de requisitos.

Nessa pesquisa, é citado que 75% das empresas, envolvidas no estudo, possuem um processo de engenharia de requisitos deficiente, e que 58,3% das empresas afirmam que não possuem um processo bem definido de engenharia de requisitos.

A partir dos resultados, a autora afirma que os problemas mais frequentes relacionados ao processo de engenharia de requisitos são a falta de processo definido de ER, dificuldade de entender as reais necessidades dos usuários, marketing deficiente, dificuldade de interação com clientes e dificuldade em gerenciar requisitos.

Ressalta-se que a maioria das empresas entrevistadas reconhece a importância de se ter um processo de engenharia de requisitos definido, porém, justificam que a falta de recursos financeiros e de pessoal capacitado em exercer atividades de engenharia de software dificultam a adoção de certas práticas.

A Figura 27 ilustra um modelo de processos que envolve desde a elicitação até a validação dos requisitos, citados pela autora.

A autora ainda conclui que as empresas de produtos de software podem ter grandes benefícios se adotarem boas práticas de ER, sendo que as práticas devem ser ágeis e precisam se adaptar as necessidades e recursos disponíveis pela empresa.

Baseando-se nesse trabalho, foi identificado a necessidade de modelar um processo que vise principalmente a validação dos requisitos antes de entrarem para o desenvolvimento do software propriamente dito.

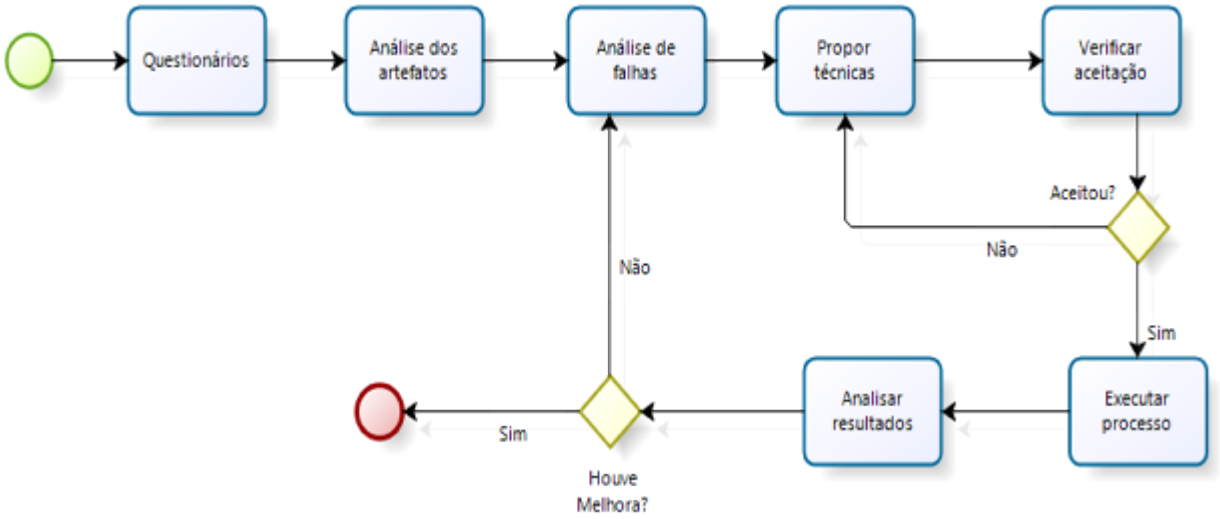
Figura 27 – Esquema de especificação de requisitos

Fonte: (ALVES, 2008)

4 METODOLOGIA

O foco do trabalho será a avaliação de qual é o impacto do uso de Engenharia de Requisitos no processo de desenvolvimento de software, visando a diminuição de retrabalho, principalmente pela equipe de desenvolvimento. Para isso, foi utilizado como ambiente de teste o setor produção de software do NTIC. Primeiramente, foi realizada uma análise da situação atual, verificando as técnicas utilizadas na elicitação dos requisitos, verificando a documentação dos requisitos e tabulando as falhas ocorridas em virtude de problemas no processo. A Figura 28 ilustra a sequência prevista para o trabalho. A partir dessas informações, foi criado um modelo de processo e desenvolvimento, onde foram propostas técnicas que possam se adaptar ao contexto do NTIC, e após a aceitação dessa proposta pelo NTIC, foi escolhido e executado um projeto piloto, utilizando o processo proposto por este trabalho (apresentado no capítulo 5). Após a execução desse processo, foram analisados os resultados obtidos e comparados com outros projetos executados pelo NTIC.

Figura 28 – Sequência de atividades previstas para a realização deste trabalho



Fonte: Elaborada pelo autor

4.1 Questionários de avaliação

Gianesini (2008, p. 44) cita em seu trabalho a aplicação de CMM e do modelo de Melhoria de Processos de Software Brasileiro (MPS-BR) na avaliação de empresas, porém, em sua pesquisa, utiliza apenas algumas premissas de CMM.

Em virtude de o NTIC estar disposto a buscar certificação MPS-BR, neste trabalho, tomaremos como base para a elaboração de um questionário, a avaliação prevista pelo MPS-BR, nível G (Parcialmente Gerenciado), no processo de Gerência de Requisitos (GRE), cujo propósito é gerenciar os requisitos do produto e dos componentes do produto e do projeto, além de identificar inconsistências entre os requisitos, os planos do projeto e os produtos de trabalho do projeto (MPS.BR, 2011, p. 29).

Esse processo proposto pelo MPS-BR possui os seguintes resultados esperados:

Tabela 10 – Resultados esperados MPS-BR – Gerência de Requisitos

GRE 1	O entendimento dos requisitos é obtido junto aos fornecedores de requisitos;
GRE 2	Os requisitos são avaliados com base em critérios objetivos e um comprometimento da equipe técnica com estes requisitos é obtido;
GRE 3	A rastreabilidade bidirecional entre os requisitos e os produtos de trabalho é estabelecida e mantida;
GRE 4	Revisões em planos e produtos de trabalho do projeto são realizadas visando identificar e corrigir inconsistências em relação aos requisitos;
GRE 5	Mudanças nos requisitos são gerenciadas ao longo do projeto.

Fonte: (MPS.BR, 2011)

Para esse processo, foi montado um questionário com as seguintes perguntas:

Tabela 11 – Questionário 01 – Gerência de requisitos

a) É estabelecida uma lista de critérios para identificar os provedores de requisitos?
b) São estabelecidos critérios objetivos para a validação dos requisitos?
c) São realizadas reuniões para validar os critérios de aceitação?
d) Os acordos são documentados sobre os requisitos e seus compromissos?
e) São realizadas analises dos requisitos visando o comprometimento da equipe técnica?
f) Como é realizado o entendimento dos requisitos pelas partes envolvidas?
g) Os requisitos são numerados com identificador único?

h) é determinada a caracterização do requisito?
i) São determinadas prioridades de requisitos?
j) São realizadas reuniões visado o andamento do projeto comparando-o com seus requisitos?
k) São realizadas ações necessárias para a correção de possíveis inconsistências?
l) São documentadas todas as mudanças dos requisitos?
m)É mantido um histórico das mudanças de requisitos e da linha de raciocínio utilizada?
n) É avaliado o impacto das mudanças de requisitos do ponto de vista das partes interessadas relevantes?

Fonte: Elaborado pelo autor

Também foi utilizado como base para o questionário, o processo de Desenvolvimento de Requisitos – DRE, também proposto pelo MPS-BR, no nível D, cujo propósito é definir os requisitos do cliente, do produto e dos componentes do produto (MPS.BR, 2011, p. 39).

Os resultados esperados são:

Tabela 12 - Resultados esperados MPS-BR – Desenvolvimento de Requisitos

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;
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;
DRE 4	Os requisitos funcionais e não funcionais de cada componente do produto são refinados, elaborados e alocados;
DRE 5	Interfaces internas e externas do produto e de cada componente do produto são definidas;
DRE 6	Conceitos operacionais e cenários são desenvolvidos;
DRE 7	Os requisitos são analisados, usando critérios definidos, para balancear as necessidades dos interessados com as restrições existentes;

DRE 8	Os requisitos são validados.

Fonte: (MPS.BR, 2011)

Para obter informações referentes ao processo descrito na Tabela 12, foi elaborado outro questionário, contendo as perguntas abaixo:

Tabela 13 – Questionário 02 – Desenvolvimento de requisitos

a) É criada uma lista com as necessidades e suas prioridades? Quais as práticas?
b) Os itens da lista são classificados como funcionais e não funcionais?
c) São criados conceitos operacionais e cenários para detalhar os requisitos? Quais os artefatos?
d) A lista de necessidade é cruzada coma as restrições para o projeto?
e) Com a lista de requisitos em sua versão final é validada? Como?

Fonte: Elaborado pelo autor

Esses questionários foram ser respondidos pelos responsáveis da CAU, visto que a definição dos requisitos é de sua responsabilidade.

A avaliação do questionário é apresentada no APÊNDICE A.

4.2 Análise dos artefatos

Foram analisados os documentos de especificação de requisitos elaborados pela CAU e executados pela CODEV, do projeto de Identificação Institucional, conforme os critérios elencados nos capítulos 2.3.3 e 2.3.4 deste trabalho.

A avaliação desta análise é apresentada no APÊNDICE B.

4.3 Análise de falhas

A partir dos resultados obtidos nos questionários e na análise dos artefatos, foram tabulados os erros/falhas/inconsistências do processo que acabam ocasionando o retrabalho tanto por parte da equipe CODEV, quanto da equipe CAU. Para a análise, foram tabulados os resultados obtidos por três projetos de módulos do sistema GURI[5], sendo que dois módulos, Portal do Professor e Identificação Institucional, foram realizados sem o modelo proposto por esse trabalho, e o

[5] Gestão Unificada de Recursos Institucionais – software de gestão desenvolvido pela UNIPAMPA

terceiro, módulo Ouvidoria, foi o caso piloto, em que as equipes utilizaram o modelo de processo proposto por este trabalho.

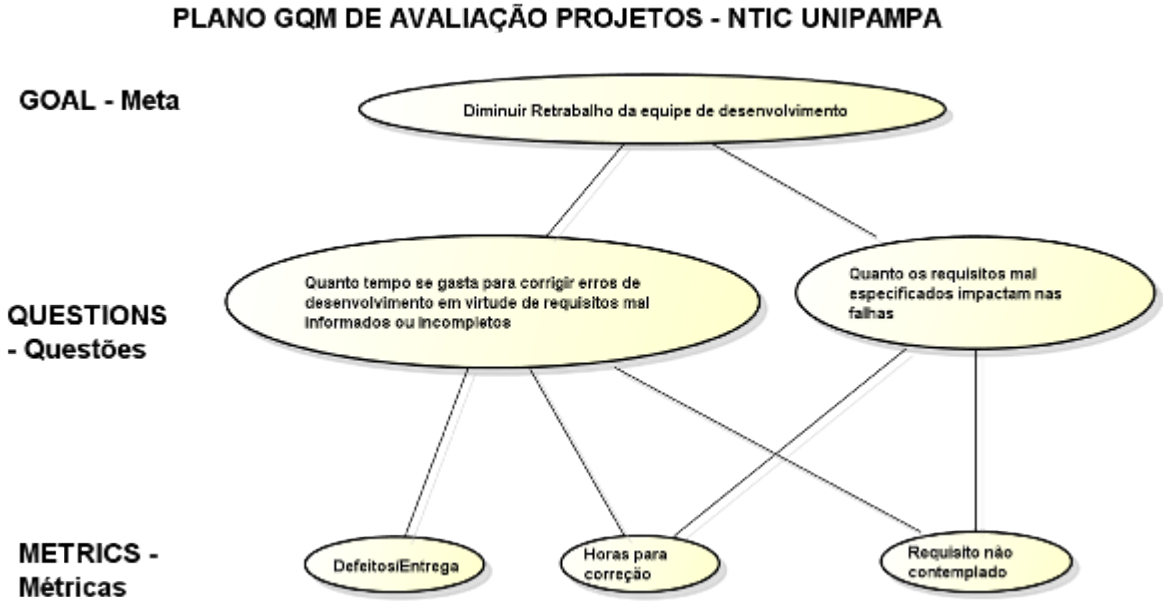
Para essa análise, foi utilizado o paradigma GQM, apresentado anteriormente, cuja primeira tarefa é definir um programa de mensuração. A Tabela 14 apresenta esse programa de mensuração, onde são definidas as dimensões da avaliação, ou seja, estabelece os objetivos e as fronteiras em que irá se basear tal avaliação. A finalidade desse programa de mensuração é justificar, a todos os envolvidos no processo, a necessidade de cada métrica a ser utilizada.

Por conseguinte, foi criado um plano de avaliação de projetos, visando identificar a quantidade de esforço desperdiçada em virtude de falhas na especificação dos requisitos. A Figura 29 ilustra esse plano GQM.

Tabela 14 – Programa de mensuração

Dimensão	Definição	Objeto
Objeto de Estudo	O que será analisado?	Processos de desenvolvimento de software no âmbito do NTIC - UNIPAMPA
Objetivo	Porque o objeto será analisado?	Existe muito retrabalho em virtude de problemas nas especificações dos requisitos
Enfoque de Qualidade	Qual atributo do objeto será analisado?	Quantidade de esforço dispensado para corrigir erros/falhas de desenvolvimento em virtude de inconsistências nas especificações de requisitos
Ponto de Vista	Quem vai usar os dados coletados?	Gerentes do projeto
Contexto	Em qual ambiente está localizado?	Projetos de módulos do sistema GURI – NTIC – Unipampa

Figura 29 – Plano GQM de avaliação de projetos – NTIC UNIPAMPA



Fonte: Elaborado pelo autor

A Tabela 15 apresenta o plano de mensuração propriamente dito, estabelecendo exatamente o que será mensurado, quais os propósitos de cada indicador, além de deixar claro como será avaliado cada resultado.

Tabela 15 – Plano de mensuração

Id	Propósito	Unidade	Periodi-cidade	Respon-sável	Apresen-tação	Interpreta-ção
Quantidade de defeitos encontrados por entrega	Identificar a quantidade e o momento da identificação dos defeitos	Unidade	Por entrega	Gerente	Gráfico de colunas	Quanto menor o número de defeitos, melhor
Horas dispensadas para corrigir defeitos	Identificar a quantidade de tempo dispensado para corrigir defeitos	Horas de correção/ total de horas do projeto	Por entrega	Gerente	Gráfico de colunas	Quanto menor, melhor
Requisito não contemplado	Identificar requisitos não contemplados por falta/incon-sistência de especificação	Unidade	Por entrega	Gerente	Gráfico de colunas	Quanto menor, melhor

Fonte: Elaborado pelo autor

5 PROPOSTA

Segundo resultados da pesquisa realizada por (GIANESINI, 2008), a presença do cliente no momento da definição de requisitos é primordial, inclusive é utilizado pela maioria das empresas entrevistadas, além de apresentar que a padronização dos processos de desenvolvimento é meta para a maioria das empresas e que a falta do mesmo pode acarretar baixa eficiência nesse processo.

Assim sendo, a partir do levantamento bibliográfico, dos trabalhos elencados e dos resultados obtidos através dos processos citados nos capítulos 4.1, 4.2 e 4.3, foram realizadas reuniões com as duas equipes, CODEV e CAU, de forma separada, para propor técnicas e boas práticas, já discutidas neste trabalho, que melhor se adaptam a realidade operacional do NTIC.

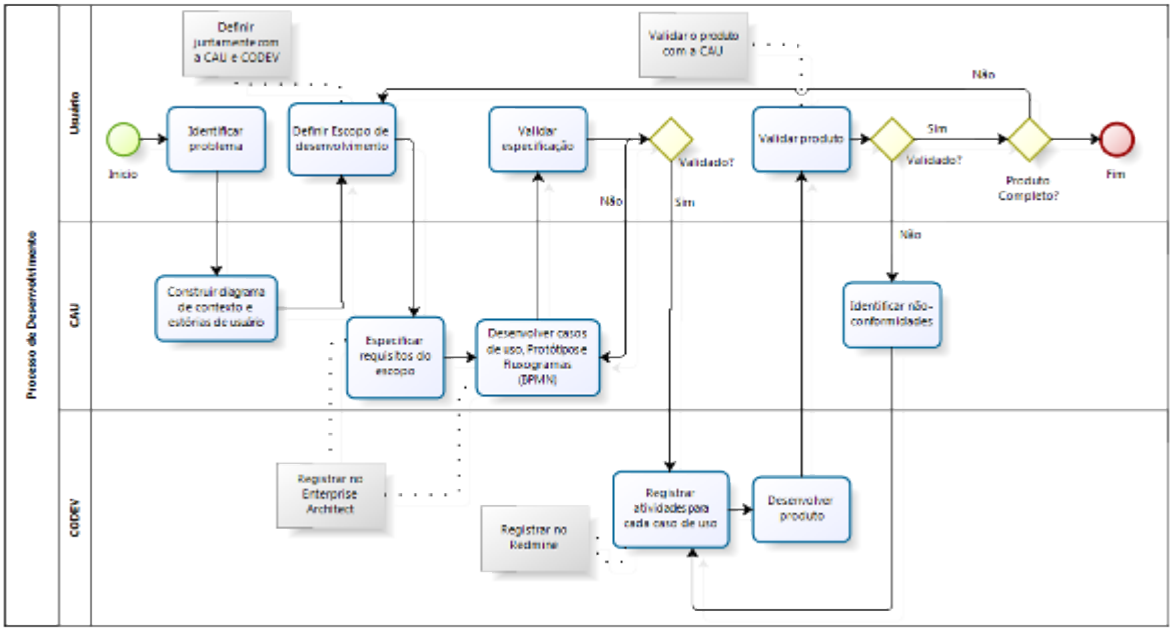
Primeiramente foi realizada a reunião com a equipe da CODEV, visando identificar quais os artefatos de especificação que mais agregariam valor ao seu trabalho e que não estariam sendo realizados pela CAU.

De posse das informações obtidas pela primeira reunião, foi realizada uma reunião com os integrantes da CAU, na cidade de Bagé-RS, onde foi apresentado aos mesmos as necessidades da CODEV e consequentemente discutidos quais os artefatos que poderiam ser construídos por essa equipe.

Após as reuniões, foi modelado e, consequentemente, proposto um processo de trabalho para que ambas as equipes utilizem-no em um projeto piloto, módulo Ouvidoria, para no final do projeto, identificar os resultados obtidos pelo mesmo. A Figura 30 ilustra o modelo proposto, onde estão definidas as tarefas para cada um dos envolvidos no processo (Usuário, CAU e CODEV).

Juntamente com o modelo, foi repassado um manual explicativo sobre cada tarefa a ser realizada. O conteúdo completo desse manual encontra-se no APÊNDICE C deste trabalho.

Figura 30 – Modelo de processo de desenvolvimento proposto



Fonte: Elaborado pelo autor

A seguir, é apresentado o esclarecimento de cada tarefa a ser desenvolvida.

a) **Identificar Problema** (Usuário) – *Identificar uma necessidade de desenvolvimento de software e solicitar junto à CAU a abertura de um processo de desenvolvimento.* Essa tarefa tem a finalidade de envolver o usuário, principalmente nas fases iniciais. Conforme citado por (RIBEIRO, 2008, p. 46), onde se deve haver o envolvimento de várias pessoas no processo, e que é essencial que seja possível atribuir a essas pessoas algumas horas de trabalho semanal para o processo de desenvolvimento de requisitos;

b) **Construir diagrama de contexto e estórias de usuário** (CAU) – *A partir das demandas identificadas junto ao usuário (stakeholder), desenvolver um diagrama de contexto contendo essas necessidades de forma resumida, além de descrever através de estórias de usuário essas necessidades, com o intuito de facilitar a definição do escopo de desenvolvimento.* Em virtude do NTIC propor o uso de metodologias ágeis, o uso de diagramas de contexto serão úteis para ter uma visão geral do problema, e que a utilização de estórias de usuário serão utilizadas para se dar um maior esclarecimento sobre os processos executados pelos usuários, ao qual se tem a intenção de automatizar.

c) **Definir escopo de desenvolvimento** (Usuário, CAU, CODEV) - *A partir dos requisitos apresentados no diagrama de contexto e nas estórias de usuário,*

definir os subprodutos de trabalho que tenham maior prioridade e que sejam possíveis de serem desenvolvidos no período definido como Escopo de Desenvolvimento. É interessante que seja realizada uma reunião com representantes dos setores envolvidos (Usuário, CAU e CODEV) para que seja definido o escopo a ser desenvolvido. Ratificando o uso de métodos ágeis, durante a execução das tarefas *c* e *d*, são definidas as prioridades e o produto de trabalho a ser entregue ao final de um escopo de desenvolvimento, sendo que esse escopo deverá ser de no máximo 30 (trinta) dias

d) **Especificar requisitos do escopo** (CAU) - *A partir da definição dos requisitos a serem trabalhados no escopo, a CAU deverá realizar o levantamento de requisitos juntamente com os stakeholders para especificar o produto a ser desenvolvido no escopo de desenvolvimento. Toda a especificação deverá ser registrada no software Enterprise Architec[6]t.*

e) **Desenvolver casos de uso, Protótipos e Fluxogramas (BPMN)** (CAU) - *A CAU deverá desenvolver os fluxogramas que representem o fluxo das atividades e os casos de uso referentes ao escopo, além de desenvolver os protótipos de telas (wireframes[7]). Essa documentação deverá ser escrita de forma que os stakeholders consigam validar o que foi especificado, e, que os casos de uso possuam detalhes que sejam suficientes para que a equipe da CODEV consiga desenvolver o produto.* Tendo em vista a necessidade de entendimento por parte do cliente, foi definido que a CAU deverá utilizar diagramas BPMN para especificar a sequência do processo do produto a ser desenvolvido. Os *wireframes* serão utilizados para que o cliente possua uma visualização melhor do resultado esperado do produto. Os casos de uso serão úteis para descrever com maiores detalhes da sequência de atividades previstas de uso por parte do cliente, incluindo o fluxo principal e os fluxos alternativos e de exceção;

f) **Validar especificação** (Usuário e CAU) - *O usuário solicitante deverá verificar se a especificação confere com a sua real necessidade. Caso exista alguma não-conformidade, a CAU deverá refatorar a especificação até que a mesma seja validada com os stakeholders.* O cliente irá validar a documentação especificada pela CAU, e, caso positivo, entrar na linha de desenvolvimento propriamente dita,

[6] *Enterprise Architect* – Software de modelagem conceitual (UML, SysML, BPMN, etc)
[7] *Wireframe* – termo utilizado para definir protótipos de tela

caso contrário, será solicitada junto à CAU uma refatoração dessa especificação até que seja validada. Essa tarefa, e as demais anteriores, seguem o modelo citado por (ALVES, 2008, p. 15), onde a não aceitação do documento implica no retorno à tarefa inicial (elicitação), seguindo a sequência até a sua completa validação;

g) **Registrar atividades para cada caso de uso** (CODEV) - *A partir da especificação do escopo de desenvolvimento, a CODEV deverá registrar os casos de uso do escopo e definir, a partir desses casos de uso, as tarefas de projeto, desenvolvimento, testes e documentação, que deverão ser distribuídas à equipe de desenvolvimento. Todas essas tarefas deverão ser registradas no software Redmine*[8]*.* Para que se possa realizar o registro e controle das tarefas a serem realizadas e, consequentemente o tempo de trabalho, foi definido que a CODEV deverá registrar todas as atividades necessárias para o desenvolvimento do produto previsto no escopo em software específico de gestão de projetos (*Redmine*);

h) **Desenvolver produto** (CODEV) - *Nessa fase, a equipe de desenvolvimento deverá executar as tarefas registradas no Redmine. Ao final de cada expediente de trabalho, cada membro da equipe de desenvolvimento deverá atualizar no Redmine as informações relativas às tarefas desenvolvidas no expediente, tais como: Tempo despendido, porcentagem concluída, etc. para cada tarefa.*

i) **Validar produto** (Usuário e CAU) - *Ao final do desenvolvimento do escopo, a equipe de desenvolvimento passa para a CAU o produto de trabalho e esse, por sua vez, realiza a validação do produto desenvolvido com o setor solicitante do produto (stakeholder). Caso seja observada alguma não-conformidade, essas alterações deverão ser repassadas à CAU.* Após realizar o desenvolvimento, será necessária a realização da validação do produto junto ao cliente, e nesse momento identificar os possíveis defeitos/falhas do produto;

j) **Produto Completo?** (Usuário e CAU) - *Após a validação, verificar se todas as necessidades levantadas pelos stakeholders foram concluídas com êxito. Enquanto o produto não esteja completo, será aberto um novo escopo de*

[8] *Redmine* – Software de gestão de projetos

desenvolvimento para dar continuidade ao processo de desenvolvimento desse produto.

Cabe ressaltar que as tarefas de arquitetura, desenvolvimento, testes e documentação não foram modeladas, em virtude de fugir do escopo deste trabalho. O tempo e custo dessas tarefas estão incluídas na tarefa Desenvolver produto.

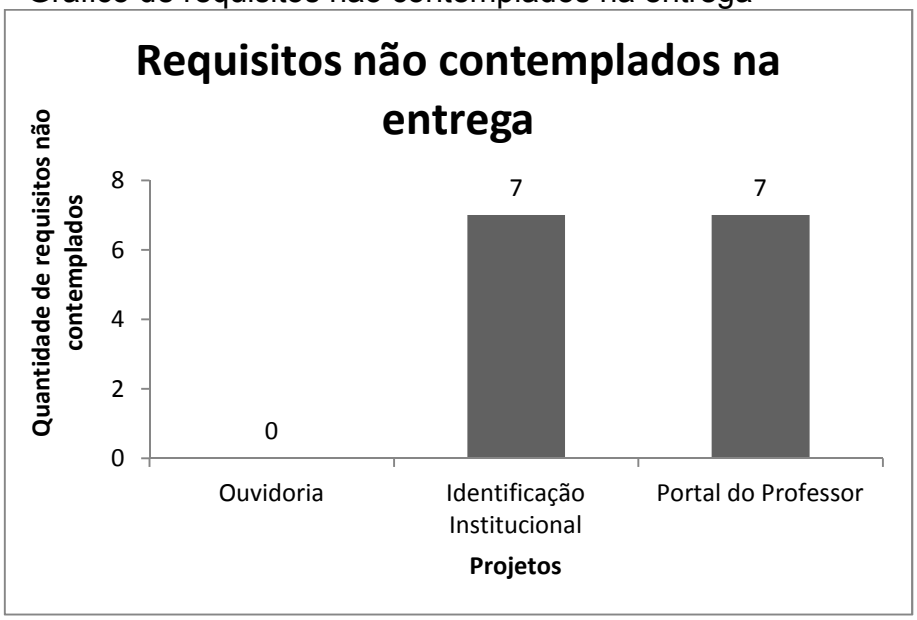
6 RESULTADOS

Após a aprovação do modelo de processo proposto, por ambas as equipes, CAU e CODEV, foi definido o projeto piloto a ser desenvolvido utilizando esse processo. O projeto escolhido foi o módulo de Ouvidoria, visto que o tempo para a avaliação não era muito extenso, e a quantidade de requisitos do módulo geraria um esforço suficiente para o trabalho de não mais que dois meses.

Após desenvolver o projeto piloto, foram coletados os dados dos três projetos conforme previsto pelo plano GQM, apresentado no capítulo 4.3, onde se constatou uma considerável melhora no processo de desenvolvimento.

O gráfico ilustrado pela Figura 31 mostra que, em virtude de uma especificação mais completa, e uma validação com o cliente antes de passar para a fase de desenvolvimento, garantiu que não houvesse nenhum requisito não contemplado, ou seja, todos os requisitos esperados pelo cliente, no módulo Ouvidoria, foram contemplados logo na primeira entrega, ao contrário dos outros dois projetos, que após a entrega foi identificado a necessidade de mais 7 (sete) requisitos em cada um dos projetos. Evidenciando um produto de maior qualidade para o cliente.

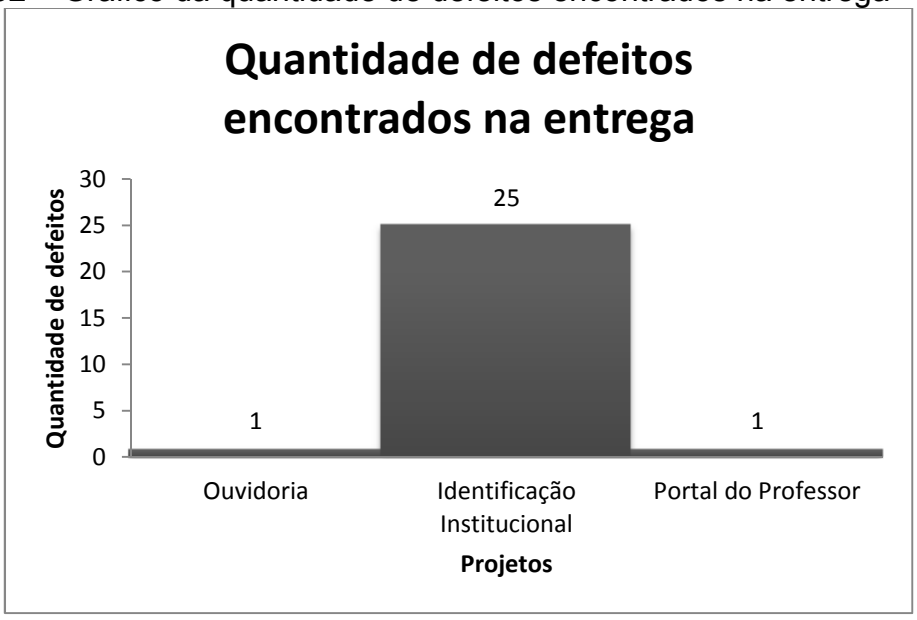
Figura 31 – Gráfico de requisitos não contemplados na entrega



Fonte: Elaborado pelo autor

Quanto à quantidade de defeitos encontrados, foram tabulados apenas os defeitos referentes à má especificação, ou seja, requisitos desenvolvidos conforme especificado, porém, que não era o esperado pelo cliente, gerando retrabalho. A Figura 32 ilustra o gráfico que representa esse indicador. Pode-se observar que foi identificado apenas um defeito no projeto piloto, e que no projeto de Identificação Institucional obteve um valor extremamente alto, comparando aos demais. Ou seja, essa especificação garantiu à equipe de desenvolvimento um melhor entendimento das regras de negócio do produto a ser entregue, ficando a maior parte do tempo de correção para pequenas alterações, principalmente visuais e textuais, não precisando afetar na estrutura do produto.

Figura 32 – Gráfico da quantidade de defeitos encontrados na entrega

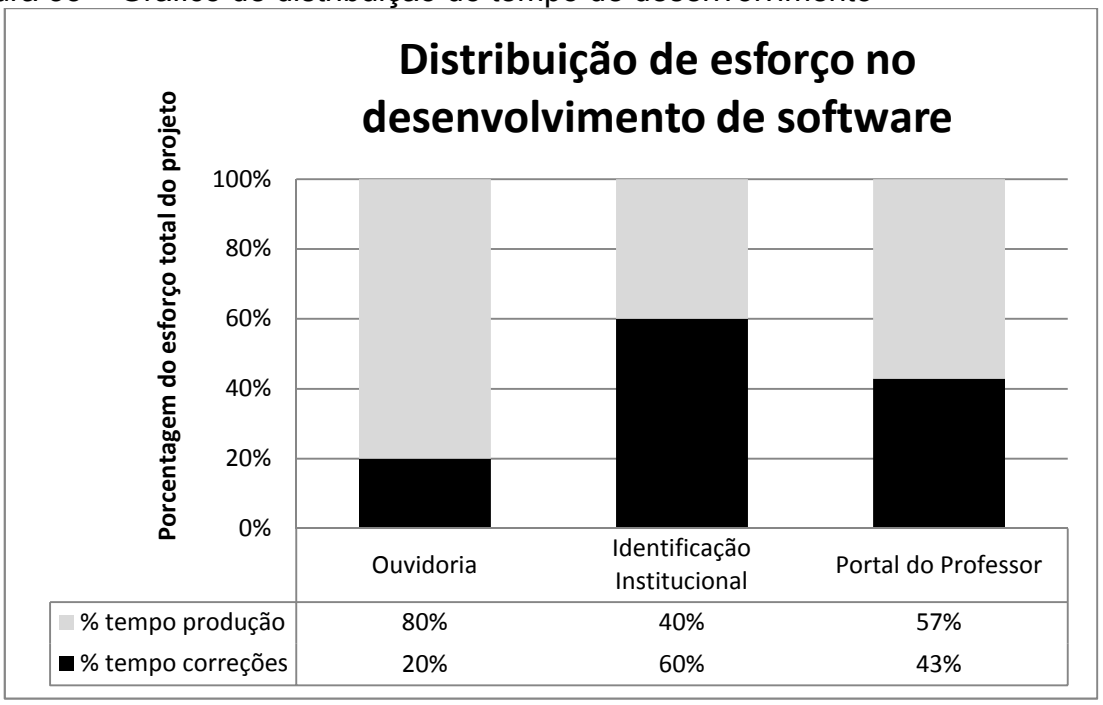


Fonte: Elaborado pelo autor

Em virtude da diferença de tamanho nos projetos, para que seja realizada uma comparação mais justa, nos próximos gráficos são apresentados os valores em forma percentual, visando garantir a proporcionalidade entre os projetos. Assim sendo, mesmo na forma percentual, o gráfico ilustrado pela Figura 33 corrobora com os dois anteriores, onde se pode observar que, no projeto Ouvidoria, utilizou-se muito menos tempo do projeto, para realizar correções, em contra partida, observa-se que no projeto de Identificação Institucional utilizou-se mais tempo para correções (60%) do que para o desenvolvimento propriamente dito (40%), ou seja, a má especificação desse projeto acarretou muito mais retrabalho, principalmente em

virtude de mudanças de requisitos que não haviam sido validadas, junto aos clientes, antes de entrar no processo de desenvolvimento.

Figura 33 – Gráfico de distribuição do tempo de desenvolvimento



Fonte: Elaborado pelo autor

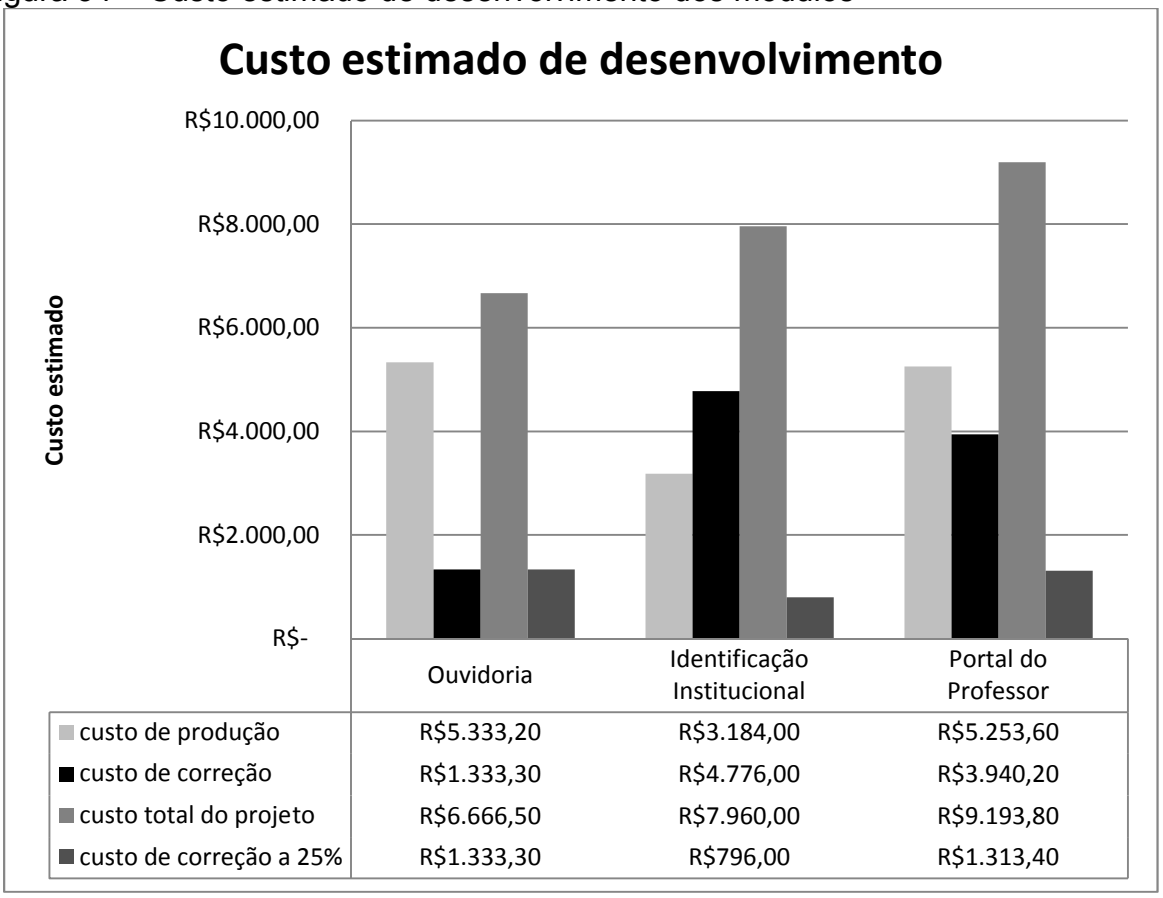
Tomando como base a média salarial dos integrantes da CODEV, o custo por hora de trabalho de cada colaborador é o equivalente a R$ 19,90 (dezenove reais e noventa centavos), conforme informado em (Portal da Transparência, 2014), foi elaborado o gráfico, ilustrado pela Figura 34, onde se verifica o valor estimado do custo das correções em cada módulo.

Com base nesses números, é identificado que no módulo Ouvidoria, o custo de correção é o equivalente a 20% do total do projeto. Olhando por outro ângulo, esse número representa 25% do custo de produção, e que nos módulos de Identificação Institucional e Portal do Professor, esses custos são de 150% e 75%, respectivamente.

Caso o índice atingido pelo módulo Ouvidoria seja replicado aos demais módulos, ou seja, caso houvesse um percentual de 25% sobre o tempo de produção para correções, ao invés de dispender o valor de R$ 4.776,00 (quatro mil, setecentos e setenta e seis reais), gastar-se-ia apenas R$ 796,00 (setecentos e noventa e seis reais), ou seja, haveria uma economia de R$ 3.980,00 (três mil, novecentos e oitenta reais) no módulo de Identificação Institucional, e R$ 2.626,80

(dois mil, seiscentos e vinte e seis reais, e oitenta centavos) no módulo Portal Professor.

Figura 34 – Custo estimado de desenvolvimento dos módulos



Fonte: Elaborado pelo autor

7 CONSIDERAÇÕES FINAIS

> Criamos hábitos ou respostas programadas para enfrentar a complexidade da vida. Quando nos defrontamos com a mudança, essa tendência de reagir conforme de costume transforma-se em fonte de resistência. (ROBBINS, JUDGE; SOBRAL, 2010, p. 569).

A pesquisa realizada por este trabalho ratifica a grande necessidade de investimento de tempo para garantir uma especificação de requisitos mais eficiente. Observou-se que, definindo um processo de desenvolvimento, conseguiu-se garantir um padrão de especificação, garantindo um documento mais eficaz e consequentemente fazendo com que o trabalho da equipe de desenvolvimento seja mais eficiente.

O custo do projeto, quanto ao retrabalho, diminuiu consideravelmente, podendo, com um maior entendimento de especificação, garantir um número ainda menor de tempo para correções. O tempo de entrega ficou bem mais perto do previsto inicialmente, em virtude da diminuição do retrabalho, notou-se que o tempo despendido para correção foi bem menor, assim, entregando o produto em menos tempo para o cliente. Além da qualidade, que certamente chegou a um índice maior, em virtude da entrega de todos os requisitos no prazo.

No decorrer do trabalho foram encontrados alguns obstáculos durante o trabalho, como a resistência à mudança do processo, principalmente no quesito especificação, visto que essa tarefa do processo passa a ser mais onerosa para a equipe da CAU. Outro empecilho foi o pouco conhecimento, dessa mesma equipe, em especificação UML, além da utilização de ferramentas para auxiliar os registros do processo de desenvolvimento, tanto na especificação, quanto na gerência do tempo, onde foram utilizadas as ferramentas Enterprise Architect e *Redmine*, respectivamente.

Quanto a trabalhos futuros, seria interessante se esse experimento pudesse ser replicado em mais projetos, inclusive fora do contexto da Unipampa, para verificar se os resultados obtidos possam ser validados em outras situações.

Também seria interessante para a avaliação, realizar algum tipo de métrica para definir o tamanho do projeto, podendo utilizar para isso, por exemplo, cálculos de pontos de função.

Espera-se que, a partir desse experimento, refine-se esse modelo e que o mesmo seja institucionado dentro do contexto de desenvolvimento de software no NTIC-Unipampa.

8 CRONOGRAMA

Tabela 16 – Cronograma do trabalho

Período / Atividades	2013							2014		
	Junho	Julho	Agosto	Setembro	Outubro	Novembro	Dezembro	Janeiro	Fevereiro	Março
Construção da proposta do TCC	✔									
Revisão bibliográfica		✔	✔							
Definição de processo de avaliação				✔						
Análise situação atual NTIC				✔						
Defesa TCC I					✔					
Definição de proposta de técnicas					✔	✔				
Execução de processos para analise							✔	✔	✔	
Analisar resultados									✔	✔
Escrita da Monografia		✔	✔	✔	✔	✔	✔	✔	✔	✔
Defesa TCC II										✔

. Fonte: Elaborada pelo autor

9 REFERÊNCIAS

ALVES, Carina F. **Uma Experiência de Engenharia de Requisitos em Empresas de Software**. Recife: 2008.

BALDAM, Roquemar et al. **Gerenciamento de Processos de Negócios - BPM - Business Process Management**. 2ª. ed. São Paulo: Érica, 2007.

BASILI, Victor R.; Rombach, H. Dieter. **THE TAME PROJECT? Towards Improvement-Oriented Software Environments**, 1988.

COCKBURN, Alistair. **Escrevendo Casos de Uso Eficazes**. Porto Alegre: Bookman, 2005.

GIANESINI, Débora. **Pesquisa da engenharia de requisitos em Empresas de Desenvolvimento de Software de Micro, Pequeno e Médio Porte de Joinville.** Joinville, 2008.

GRESSE, Christiane. **Utilização do GQM no Desenvolvimento de Software**. [S.l.]: [s.n.], 2000.

GRESSE, Christiane; RUHE, Günther. **Análise de Custo e Beneficio de Mensuração Baseada em GQM**: Um Estudo de Caso Replicado, 1999.

KOSCIANSKI, André; SOARES, Michel D. S. **Qualidade de Software:** aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. 2ª. ed. São Paulo: Novatec, 2007.

MELO, Ana C. **Desenvolvendo aplicações com UML 2.2:** do conceitual à implementação. Rio de Janeiro: Brasport, 2010.

MPS.BR. Softex. **Softex**, Julho 2011. Disponivel em: <http://www.softex.br/wp-content/uploads/2013/07/MPS.BR_Guia_de_Implementacao_Parte_1_2011.pdf>. Acesso em: 20 Setembro 2013.

PAULA FILHO, Wilson D. P. **Engenharia de Software:** fundamentos, métodos e padrões. 3ª. ed. Rio de Janeiro: LTC, 2009.

PFLEEGER, Shari L. **Engenharia de Software:** teoria e prática. São Paulo: Prentice Hall, 2004.

PORTAL da Transparência. **Portal da Transparência**, 2014. Disponivel em: <http://www.portaldatransparencia.gov.br/>. Acesso em: 09 mar. 2014.

PRESSMAN, ROGER S. **Engenharia de Software**. 7ª. ed. PORTO ALEGRE: AMGH, 2011.

RIBEIRO, Rodrigo C. S. **Metodologia para Equipas de Desenvolvimento de Requisitos de Sistemas de Informação**, 2008.

ROBBINS, Stephen P.; JUDGE, Timothy A.; SOBRAL, Filipe. **Comportamento Organizacional**. 14ª. ed. São Paulo: Pearson, 2010.

SOMMERVILLE, Ian. **Engenharia de Software**. Tradução de IVAN BOSNIC e KALINKA G. DE O. GONÇALVES. 9ª. ed. SÃO PAULO: PEARSON, 2011.

UNIPAMPA. NTIC - Núcleo de Tecnologia da Informação e Comunicação. Disponivel em: <http://ntic.unipampa.edu.br>. Acesso em: 20 Setembro 2013.

VALLE, Rogerio; OLIVEIRA, Saulo B. **Análise e Modelagem de Processos de Negócio**. 1ª. ed. São Paulo: Atlas, 2009.

APÊNDICE A – Resultados obtidos pelo questionário de avaliação

Questionário 01 - Gerência de requisitos

- Nem sempre são estabelecidos critérios objetivos para a validação dos requisitos;

- Os acordos relativos aos requisitos e seus compromissos são realizados informalmente;

- Não há uma garantia de entendimento do documento de requisitos, por parte da equipe de desenvolvimento;

- Não há um acompanhamento constante do andamento do projeto;

- As inconsistências somente são verificadas após a entrega do produto;

- Os impactos das mudanças dos requisitos no transcorrer do projeto não são calculados;

Questionário 02 - Desenvolvimento de requisitos

- Quanto à elicitação dos requisitos, são utilizadas as técnicas de entrevistas, perguntas e etnografia, porém não se utilizam cenários e diagramas de contexto, principalmente porque geralmente o cliente tem dificuldade em definir os processos que realiza;

- Os cenários operacionais são modelados parcialmente;

- É realizado o cruzamento dos requisitos com as restrições de projeto, porém sem uma técnica específica, além de não ser documentado;

- Quanto à validação do documento de requisitos, esse não é validado com o cliente, visto que o cliente não consegue entender o documento. Para a realização da validação, a CAU monta protótipos de tela (*wireframes*) e envia para a CODEV montar protótipos HTML (visto que a base dos sistemas é web), após isso, a CODEV envia o modelo de protótipo de tela para a CAU, e essa, por sua vez, valida com o cliente. Sendo que a CAU poderia validar diretamente com o cliente através dos *wireframes*.

APÊNDICE B – Avaliação dos artefatos do projeto de Identificação Institucional desenvolvido pelo NTIC – UNIPAMPA

Para analisar os artefatos, foi selecionado um dos projetos que estava em andamento no momento da pesquisa, o qual será citado logo abaixo.

O projeto consiste em criar, no sistema GURI, uma tela para inserção de dados e da foto para a confecção de crachás.

Foi criado um modelo de crachá, Figura 35, que consta como anexo em um e-mail enviado em 18 de fevereiro de 2013 para a CODEV.

Figura 35 – Primeiro modelo de crachá

Fonte: CODEV

Nota-se que não se tem as dimensões do crachá, muito menos a dimensão dos campos a serem preenchidos. Esse problema é evidenciado pela equipe de CODEV, que mais adiante envia um e-mail solicitando mais esclarecimentos, ilustrado pela Figura 36:

Figura 36 – e-mail enviado pela CODEV solicitando esclarecimentos

Em 26/02/2013 23:46, [redacted] escreveu:

Boa noite,
só para fins de esclarecimentos e para que eu possa ja ir estudando o Módulo,
levanto algumas questões:

- Já foi definido o padrão/layout e formato do arquivo?
- Se foi, como serão exportadas as imagens? como vamos fazer o link entre o indivíduo expotado e sua imagem??
- Existe limitação no tamanho do nome do cidadão, se existe, qual é esse valor?(20, 30, 40... caracters)

Pretendo até amanhã saber disso, para eu possa submeter o modulo para testes até sexta, porque sei que quando estourar vão querer o módulo para ontem, sabem como éh.

Hoje o [redacted] me explicou o esboço do DER do módulo, fizemos algumas alterações, e ja fiz os SQLs para buscar os dados do indivíduo no SIE, e também a sugestão de nome para o crachá
caso este estrapole o limite de tamanho(se existir alguma regra pra isso).

Fonte: CODEV

Em 22 de maio, é enviado um novo modelo de crachá, porém, continuando com os mesmos problemas citados anteriormente (Figura 37)

Figura 37 – Segundo modelo de crachá para servidores

Fonte: CODEV

Figura 38 – Modelo crachá para acadêmicos

Fonte: CODEV

Após, o e-mail (representado na Figura 36) foi respondido com o texto abaixo:

Figura 39 – e-mail de resposta

1º: Qual o tamanho máximo de caracteres que pode ser impresso no crachá para o nome do aluno/servidor?
Para os crachás acadêmicos os campos não terão limites.
Para os do servidores, devido o layout diferenciado, foi calculado o máximo de:
50 caracteres para o nome;
25 caracteres para local;
25 caracteres para cargo.
Os demais campos não deverão ter diferenciação de nº de caracteres.

2º. Qual o tamanho máximo de caracteres que pode ser impresso no crachá para o curso/cargo do aluno/servidor? (Não me recordo se irá o curso do aluno)
Respostas acima e no anexo.

3º. Qual o tamanho real da foto que vai no crachá?
Impresso ficará em 20x20mm para os crachás acadêmicos e 20x30mm para os

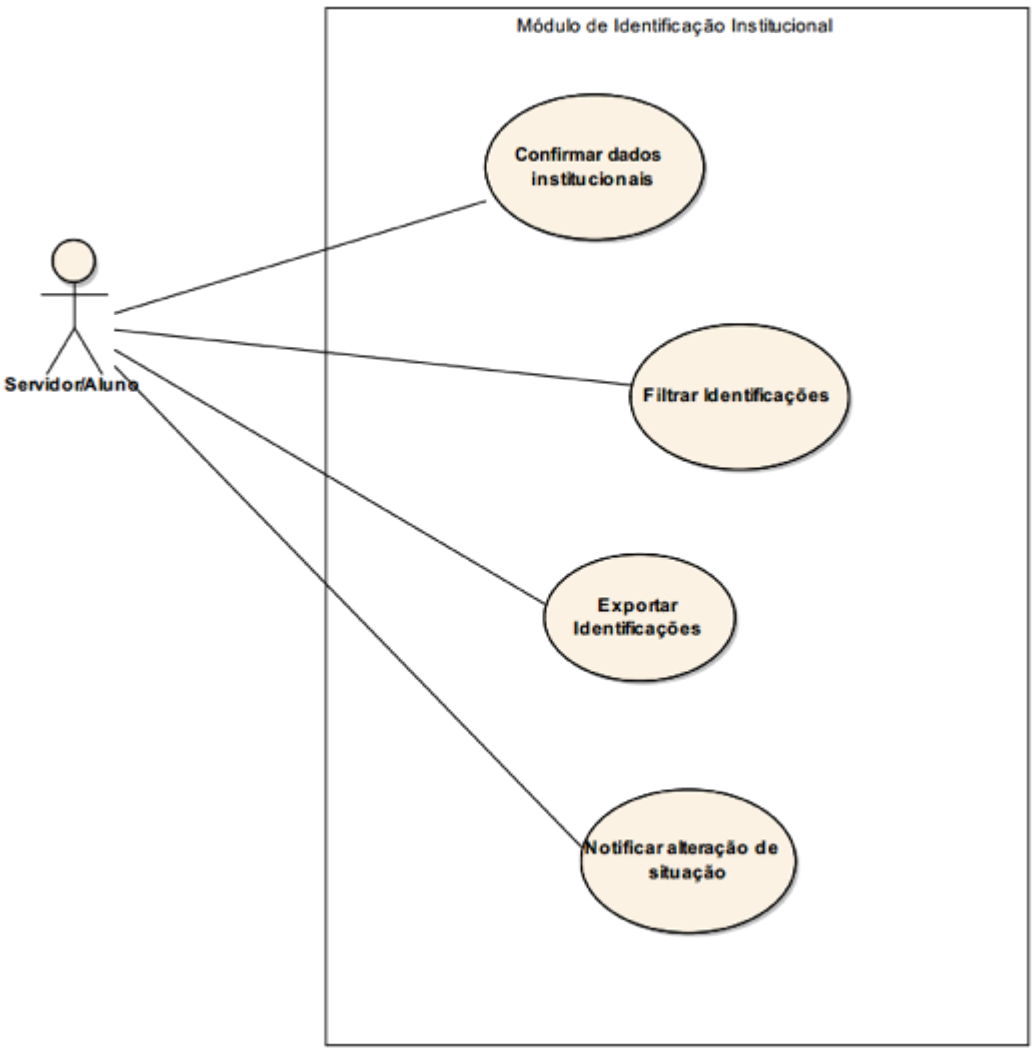
funcionais. Para uma qualidade aceitável o ideal seria trabalhar com fotos de no mínimo 500 pixels de altura e largura proporcional. Inclusive seria interessante incluir esta limitação no upload da imagem, se possível.

Respondendo à duvida do XXX, os crachás terão tamanho único de 54x86mm.

Fonte: CODEV

Cabe ainda ressaltar que, pelo menos para esse projeto em questão, não foi enviado para a CODEV um documento de requisitos explícito. A Figura 40 ilustra um diagrama de contexto utilizado para esse projeto. Porém, o próprio diagrama possui inconsistências, visto que os casos de uso apresentados estão ligados a um ator com as mesmas funções, tanto para alunos, quanto para servidor (funcionário público lotado na UNIPAMPA).

Figura 40 – Diagrama de contexto sistema de crachá



Fonte: CODEV

Para realizar a gerência dos projetos de software, no âmbito do NTIC, é utilizada como ferramenta o *software Mantis Bug Tracker*[9], onde são registradas as tarefas para execução pela equipe da CODEV. A Tabela 17 apresenta um resumo das ações definidas para o projeto em questão.

Tabela 17 - Registros no software Mantis

Núm.	**Resumo**	**Descrição**	**Data de envio**	**Anotações**	**Data anotação**
334	Foto SIE-GURI	A PRAEC perguntou se as fotos já adicionadas no SIE poderiam ser utilizadas na carteirinha do aluno e se o inverso também é viável, carregar no SIE as fotos enviadas para os crachás. Como resposta parcial informei que a foto do SIE é de muito baixa qualidade e que não seria conveniente oferecer como opção aos alunos. Foi pedido que analisasse a possibilidade	29/07/2013		
335	Tutorial de edição de imagens	Ficar disponível um tutorial bem simples que ensine o aluno a redimensionar sua imagem ao lado do campo de upload de foto. Devem ser observados os aplicativos mais usados (Paint, GIMP etc)	29/07/2013		
336	Preview	Está duplicado o Campus Deve constar o provável ano que o aluno vai se formar	29/07/2013		
338	Nome da carteirinha	Colocar um aviso no NOME(Se o nome completo for muito extenso abreviar o nome do meio)	29/07/2013	Qual seria o tamanho máximo? Então o sistema que vai decidir o nome para a identificação caso isso aconteça?	17/09/2013
				Isso foi definido nas primeiras reuniões. Já não sei o que ficou decidido	17/09/2013

[9] Software *open source* destinado a registro de bugs

339	Interface pública	Disponibilizar no GURI uma interface pública para que se consultando a matrícula do aluno ele informe se o vínculo é regular. Isso substituiria a necessidade do aluno carregar o comprovante de matrícula.	29/07/2013		
340	Perfil de ADM	Disponibilizar para que nos campi um ou mais servidores tenham privilégio de fazer carteirinhas pelos alunos. Quando eles organizarem mutirões para ajudar os alunos que não conseguiram pelo GURI acharam inconveniente o aluno ter que logar no sistema para que o servidor da Universidade insira os dados.	29/07/2013		
341	Notificaçõe s	O sistema deve notificar o aluno quando for reprovado e aprovado por e-mail, automaticamente.	29/07/2013		
342	Edição de lotes	Pediu-se que os lotes possam ser editados para que aluno/servidores que já foram inseridos possam ser recolocados, caso dê algum problema na impressão.	29/07/2013		
343	Falhas e Melhorias	em anexo doc com detalhes	21/08/2013		
403	Revisar identificaçõ es	Listar fotos dos funcionários e alunos, para avaliação. A avaliação será feita por campus e por tipo de identificação (aluno/servidor), cadastrar pessoas designadas em cada campus para cada tipo de identificação.	12/08/2013		
431	Desenvolve r manual do módulo	Desenvolver o manual do módulo de identificação institucional	19/08/2013	Ok	20/08/2013

Fonte: CODEV

Processo de Desenvolvimento

Processo de Desenvolvimento de Produtos de Software

Versão: 2.0

Autor: Nasser

Descrição

Alterado item 1.1.1.5

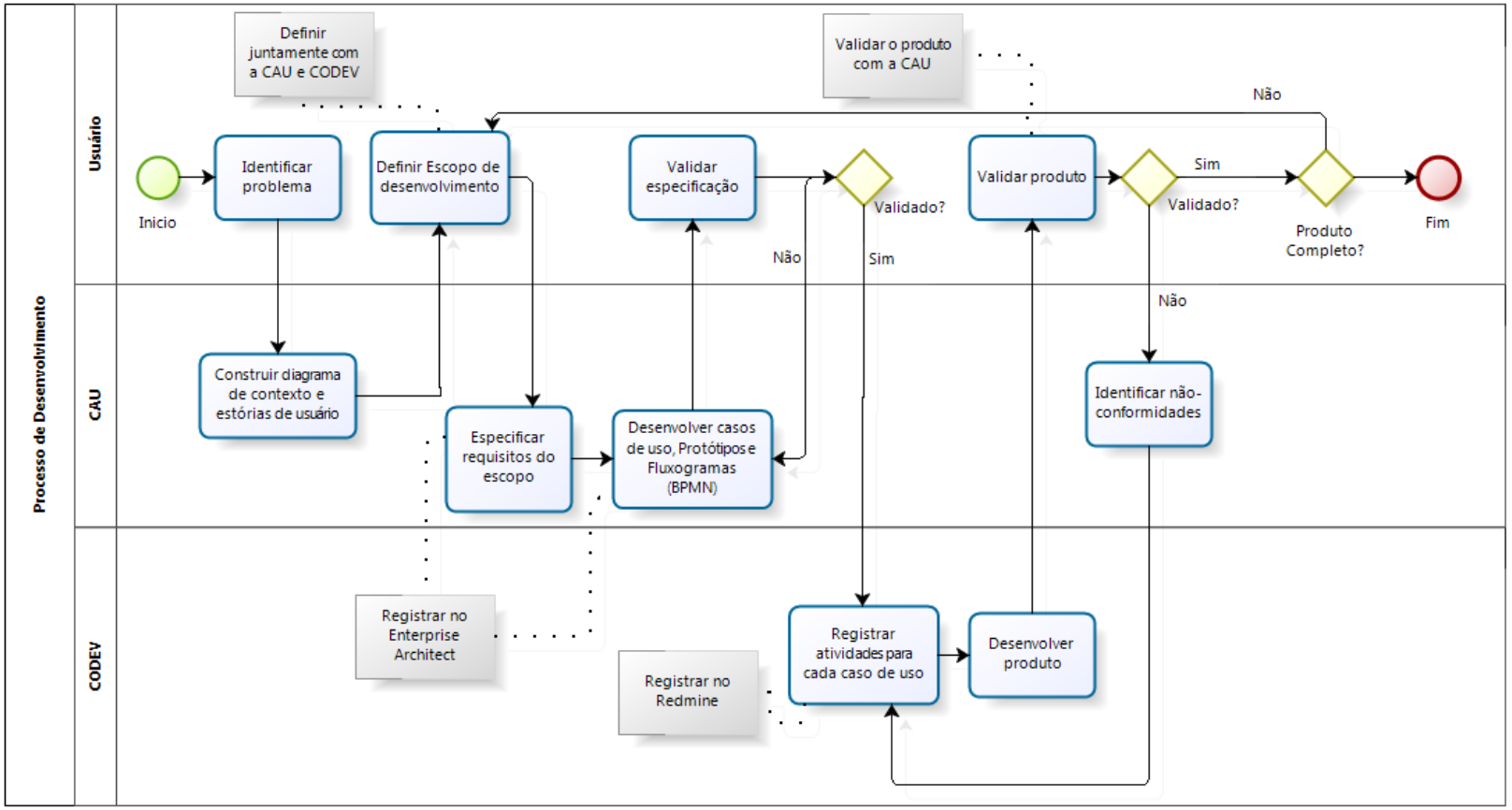
Incluido:

- Fluxograma BPMN

- Protótipos de tela

Excluido:

- Diagrama de Atividades



1.1 PROCESSO DE DESENVOLVIMENTO

Descrição

1.1.1 ELEMENTOS DO PROCESSO

1.1.1.1 Identificar problema

Descrição

Identificar uma necessidade de desenvolvimento de software e solicitar junto à CAU a abertura de um processo de desenvolvimento

1.1.1.2 Construir diagrama de contexto e estórias de usuário

Descrição

A partir das demandas identificadas junto ao usuário (stakeholder), desenvolver um diagrama de contexto contendo essas necessidades de forma resumida, além de descrever através de estórias de usuário essas necessidades, com o intuito de facilitar a definição do escopo de desenvolvimento.

1.1.1.3 Definir Escopo de desenvolvimento

Descrição

A partir dos requisitos apresentados no diagrama de contexto e nas estórias de usuário, definir os subprodutos de trabalho que tenham maior prioridade e que sejam possíveis de serem desenvolvidos no periodo definido como Escopo de Desenvolvimento. É interessante que seja realizada uma reunião com representantes dos setores envolvidos (Setor solicitante, CAU e CODEV) para que seja definido o escopo a ser desenvolvido.

1.1.1.4 Especificar requisitos do escopo

Descrição

A partir da definição dos requisitos a serem trabalhados no escopo, a CAU deverá realizar o levantamento de requisitos juntamente com os stakeholders para especificar o produto a ser desenvolvido no escopo de desenvolvimento. Toda a

especificação deverá ser registrada no software Enterprise Architect.

1.1.1.5 Desenvolver casos de uso, Protótipos e Fluxogramas (BPMN)

Descrição

A CAU deverá desenvolver os fluxogramas que representem o fluxo das atividades e os casos de uso referentes ao escopo, além de desenvolver os protótipos de telas (wireframes). Essa documentação deverá ser escrita de forma que os stakeholders consigam validar o que foi especificado, e, que os casos de uso possuam detalhes que sejam suficientes para que a equipe da CODEV consiga desenvolver o produto.

1.1.1.6 Validar especificação

Descrição

O usuário solicitante deverá verificar se a especificação confere com a sua real necessidade. Caso exista alguma não-conformidade, a CAU deverá refatorar a especificação até que a mesma seja validada com os stakeholders.

1.1.1.7 Registrar atividades para cada caso de uso

Descrição

A partir da especificação do escopo de desenvolvimento, a CODEV deverá registrar os casos de uso do escopo e definir, a partir desses casos de uso, as tarefas de projeto, desenvolvimento, testes e documentação, que deverão ser distribuidas à equipe de desenvolvimento. Todas essas tarefas deverão ser registradas no software Redmine.

1.1.1.8 Desenvolver produto

Descrição

Nessa fase, a equipe de desenvolvimento deverá executar as tarefas registradas no Redmine. Ao final de cada expediente de trabalho, cada membro da equipe de desenvolvimento deverá atualizar no Redmine as informações relativas às tarefas desenvolvidas no expediente, tais como: Tempo despendido, porcentagem concluída, etc. para cada tarefa.

1.1.1.9  Validar produto

Descrição

Ao final do desenvolvimento do escopo, a equipe de desenvolvimento passa para a CAU o produto de trabalho e esse, por sua vez, realiza a validação do produto desenvolvido com o setor solicitante do produto (stakeholder). Caso seja observada alguma não-conformidade, essas alterações deverão ser repassadas à CAU.

1.1.1.10  Produto Completo?

Descrição

Após a validação, verificar se todas as necessidades levantadas pelos stakeholders foram concluídas com êxito. Enquanto o produto não esteja completo, será aberto um novo escopo de desenvolvimento para dar continuidade ao processo de desenvolvimento desse produto.