UNIVERSIDADE DO SUL DE SANTA CATARINA

ALCIENY NUNES

RENATA APARECIDA DUARTE

MELHORIAS NO PROCESSO DE LEVANTAMENTO DE REQUISITOS PARA A QUALIFICAÇÃO DE CASOS DE TESTE

Palhoça

2015

ALCIENY NUNES
RENATA APARECIDA DUARTE

MELHORIAS NO PROCESSO DE LEVANTAMENTO DE REQUISITOS PARA A QUALIFICAÇÃO DE CASOS DE TESTE

Trabalho de Conclusão de Curso apresentado ao Curso de Graduação em Ciência da Computação da Universidade do Sul de Santa Catarina, como requisito parcial à obtenção do título de Bacharel em Ciência da Computação.

Orientador: Profa. Maria Inés Castiñeira, Dra

Palhoça

2015

ALCIENY NUNES
RENATA APARECIDA DUARTE

MELHORIAS NO PROCESSO DE REQUISITOS PARA A QUALIFICAÇÃO DE CASOS DE TESTE

Este Trabalho de Conclusão de Curso foi julgado adequado à obtenção do título de Bacharel em Ciência da Computação e aprovado em sua forma final pelo Curso de Graduação em Ciência da Computação da Universidade do Sul de Santa Catarina.

Palhoça, 19 de novembro de 2015.

Professor e orientador Maria Inés Castiñeira, Dra.
Universidade do Sul de Santa Catarina

Prof. Flavio Seci, Dr.
Universidade do Sul de Santa Catarina

Prof. Roberto Fabiano Fernandes, Me.
Universidade do Sul de Santa Catarina

AGRADECIMENTOS

Agradecemos a nossa orientadora Maria Inés Castiñeira, pelo apoio ao longo deste ano, pelas correções e por responder as incontáveis dúvidas que surgiram no decorrer deste trabalho.

ALCIENY NUNES

Agradeço primeiramente a Softplan, pelas oportunidades que me deram e pelos conhecimentos adquiridos em apenas 9 meses. Estar trabalhando nesta empresa me faz crescer na área que tanto quero exercer para o resto da minha vida.

Agradeço também aos meus colegas de trabalho que me ajudam a adquirir conhecimento todos os dias e que aceitaram me ajudar incentivando no andamento desta monografia, responderam aos questionários aplicados de forma sincera e responderam minhas dúvidas sobre o processo aplicado na empresa.

A minha família vai o meu mais sincero agradecimento por todo esse tempo me incentivando a estudar, me dando o maior apoio em tudo que desejo fazer da minha vida, aceitando as decisões que tomo e me impulsionando a sempre crescer e nunca passar por cima de ninguém para alcançar meus objetivos. Aos meus pais que foram muito pacientes pois não pude visitá-los muitas vezes, por que precisava focar nesta monografia para finalizar da forma melhor possível. Obrigada pelo carinho, pelo apoio, pelo amor e principalmente por terem orgulho de mim e da pessoa que me tornei, pois fazer vocês sentirem orgulho de mim era o principal objetivo da minha vida.

Agradeço também principalmente a minha melhor amiga Renata Duarte, pois sem você eu não teria finalizado esta monografia da melhor forma possível. Obrigada por me aturar, por me apoiar, por me entender e por ser essa excelente pessoa que você é. Muito obrigada por concluir mais esta etapa comigo.

Agradeço também a Unisul pelos conhecimentos adquiridos ao longo desses anos e pelo próximo ano que ainda está por vir.

RENATA APARECIDA DUARTE

Agradeço a Softplan, por todas as oportunidades que me deram e por me deixar trabalhar no tema deste trabalho dentro da empresa.

Ao meu colega e orientador na Softplan, André Fernando Faggion, que me ajudou na escolha do tema e pelas dicas sugeridas no decorrer do desenvolvimento deste trabalho.

Aos meus pais, por sempre apoiar minhas decisões, mesmo que não lhes agrade, por estarem sempre ao meu lado em todas as minhas conquistas e também nos momentos difíceis. Por todo o incentivo aos meus estudos e objetivos, que não são poucos. Por toda a educação maravilhosa que me deram, por tudo que ainda me ensinam e pelo amor sempre explícito que sentem.

A minha irmã, minha eterna bebezinha, por ser minha amiga, companheira, ouvinte e pela lealdade de sempre.

A minha melhor amiga Alcieny Nunes, por ter paciência nos meus surtos e desesperos, por sempre me ouvir e por finalizar junto comigo mais essa etapa da minha vida.

Ao meu gato, Tibe, que mesmo inconsciente me dá toda a paz que eu preciso, me faz companhia em todos os meus momentos em casa, me tira alguns sorrisos e me faz amá-lo cada vez mais.

A minha fiel amiga Chirley Domingues, que sempre me incentivou a estudar antes de tudo, principalmente antes de ter filhos, pelas risadas durante o trabalho, que por muitas vezes era estressante e por todas as lembranças boas que eu tenho ao lado dela.

Aos professores do Programa de Línguas da Unisul, que durante 5 anos aguentaram minhas cobranças de diários de classe e planos de ensino. Por todos os

conhecimentos que eu pude adquirir em língua portuguesa, inglesa e espanhola, e pelos momentos de cantoria, alegria e risadas nos eventos anuais do SARAU.

A todas as pessoas que são especiais ou que foram especiais em algum momento na minha vida. As que participaram diretamente ou indiretamente da minha formação, o meu muito obrigado.

RESUMO

Este trabalho tem por finalidade realizar um estudo de caso no processo de desenvolvimento de software da empresa Softplan. O objetivo do trabalho é apontar melhorias no processo de levantamento de requisitos da empresa para, assim, qualificar as especificações de testes. A monografia trata de uma pesquisa aplicada, um estudo de caso, na área de Engenharia de Software. Após definição dos objetivos foi realizada uma pesquisa bibliográfica com foco em processo de levantamento de requisitos, artefatos utilizados, tipos e processo de teste de software. Foi aplicado também, um questionário para 22 colaboradores da empresa com o intuito de coletar informações sobre o processo de software e suas características. Após a coleta dos dados foi realizada uma descrição do processo utilizando a notação BPMN. No processo de gestão de produto, foram considerados os subprocessos de elaboração do documento inicial de requisitos, estimativa preliminar e especificação de requisitos. Considerando a teoria estudada na revisão bibliográfica, as fragilidades do processo e sugestões apontadas pelos colaboradores no questionário aplicado, o processo de desenvolvimento de software foi analisado, a partir dessa análise foram propostas melhorias no processo de levantamento de requisitos e nos artefatos utilizados nesse processo, com o objetivo de qualificar as especificações de testes de software.

Palavras-chave: Análise de Requisitos. Testes. Processo de Requisito.

ABSTRACT

The main goal of this paper is to make a case study about the software developing process on the company called Softplan. This work's objective is to indicate improvements on the requirements gathering of the company in order to qualify the test specifications. The term paper is about an applied research, a case study, on Software Engineering. After the objective's definition a bibliographic research was made focusing the requirements gathering, used artifacts, types and the software testing process. Also, a questionnaire was sent to 22 company employees in order to gather information about the software process and its aspects. After the data gathering a process description was made using the BPMN notation. In the product management process, the sub-processes of the initial requirements document, the preliminary estimates and the requirements specification were taken into account. With the objective of qualifying the software tests specifications, considering the bibliographic revision's studied theory, the process weaknesses and the employees suggestions on the applied questionnaire, the software developing process was analysed and from this analysis improvements on the requirements gathering process and on the artifacts used on this process were proposed.

Key-words: Requirement Analysis, Tests, Requirement Process.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

Sumário

1. INTRODUÇÃO

A utilização de softwares tem se tornado essencial para execução das funções dentro de uma organização. É possível encontrar software em todos os lugares do mundo, serviços nacionais e internacionais, que são controlados por um sistema computacional (por exemplo: aeroportos, sinaleiras, fiscalizações rodoviárias, jogos de computador, cinema e televisão) e todos eles possuem pelo menos um computador, pois sem ele, o sistema é incapaz de funcionar sozinho. (SOMMERVILLE, 2011)

E para um sistema ser bem desenvolvido e bem especificado é preciso que ele passe por todo um processo de engenharia de software, e todo processo precisa ser, de alguma maneira, documentado. Conforme diz Ian Sommerville (2011, p. 3):

> Essa é a diferença importante entre desenvolvimento de software profissional e amador. Se você está escrevendo um programa para si mesmo, que ninguém mais usará, você não precisa se preocupar em escrever o manual do programa, documentar sua arquitetura etc. No entanto, se você está escrevendo um software que outras pessoas usarão e no qual outros engenheiros farão alterações, então você provavelmente deve fornecer informação adicional, assim como o código do programa.

A qualidade de software é uma área que está cada vez mais sendo considerada pelo setor de desenvolvimento e vem demonstrando sua importância conforme o cliente é fidelizado pela qualidade do produto de software. Entregar um produto de boa qualidade deve ser um dever de toda organização, sendo assim a empresa deve submeter o seu produto às ações de garantia de qualidade em todas as etapas de desenvolvimento de software. (PRESSMAN, 2011). Esse autor também esclarece que uma das atividades da garantia de qualidade são os testes de software.

Existem diversas atividades, métodos e documentações que procedem ao desenvolvimento de software. Todo projeto começa com uma demanda ou com uma nova ideia. No caso de atendimento de demandas o processo de software pode ser composto das seguintes fases ou atividades: entendimento da necessidade e das regras de negócio a partir do levantamento de requisitos, documentação (requisitos, casos de teste e demais documentação necessária), desenvolvimento, testes, homologação e implantação.

Conforme Roger Pressman (2011), a análise de requisitos resulta na especificação de características operacionais do software, indica a interface do software com outros elementos do sistema e estabelece restrições que o software deve atender.

Para o correto desenvolvimento de um software é necessário inicialmente que sejam levantados os requisitos do sistema. É através destes requisitos que a equipe do projeto pode entender as necessidades que devem ser implementadas. Esses requisitos devem ser bem levantados e definidos, pois eles são o cerne das necessidades do usuário, sendo também utilizados como base para a criação dos casos de testes garantindo uma melhor qualidade no sistema.

Segundo Roger Pressman (2011), é importante antes de iniciar qualquer trabalho técnico, aplicar um conjunto de tarefas de engenharia de requisitos. Estas levam a um entendimento de qual será o impacto do software sobre o negócio, o que o cliente quer e como os usuários finais irão interagir com o software.

Um requisito em linhas gerais é aquilo que o sistema deve atender, ou seja, o que o cliente deseja que o sistema faça. Por exemplo: O dono de uma farmácia deseja ter um sistema que cadastre os remédios, e que cadastre também os genéricos para cada um dos remédios. Cadastrar remédios é um requisito funcional. Um requisito pode ter diversos níveis de complexidade sendo de simples compreensão ou até mesmo com uma grande complexidade. Por isso é de grande importância que todo requisito seja bem definido e que fique muito claro para que qualquer pessoa, pertencente ao projeto ou não, consiga através da leitura dos requisitos compreendê-los. Uma especificação de requisitos bem documentada garante boa parte da qualidade de software. Se um requisito não for bem especificado pode levar a falhas no sistema, se estas forem de fácil detecção podem ser identificadas nos testes, caso contrário, possivelmente será percebido apenas no cliente, e isso deve ser evitado o máximo possível.

A importância de um profissional da área de testes está na qualidade do produto, quanto mais testes forem feitos, maior a garantia de um software de qualidade, pronto para o mercado. Para todo teste ser bem sucedido deve-se garantir que este seja planejado. O planejamento não inicia quando o software é entregue a equipe de teste para validar, ele deve começar assim que a equipe de desenvolvimento recebe os requisitos para implementação do sistema. É com base nos requisitos que são feitos os casos de testes e desenvolvido o sistema, o

levantamento dos casos de teste dessa forma, deve ser feito paralelamente ao desenvolvimento do sistema. Assim quando finalizar o desenvolvimento do sistema este deve ir direto para validação pela equipe de testes.

Este trabalho tem como foco a qualidade no levantamento e documentação dos requisitos em uma empresa de desenvolvimento de software, com vistas a facilitar e melhorar o processo de testes.

1.1. PROBLEMÁTICA

Ao realizar casos de teste na empresa Softplan foi constatada a dificuldade no entendimento das especificações de requisitos, devido às ambiguidades e divergências nas especificações. Originando, a partir destas dificuldades, a necessidade de estudo para este problema. A proposta é realizar uma análise do processo, dos documentos e dados obtidos no processo de requisitos, para apresentar sugestões de melhorias no processo de requisitos e assim, posteriormente facilitar o entendimento da elaboração de casos de teste.

Dessa forma, foi verificado que o problema não é só na Softplan muitas empresas passam pela mesma dificuldade e esse trabalho tem por intuito ajudar a Softplan e outras empresas para que melhorem seu processo de levantamento de requisitos para diminuir a dificuldade na elaboração de casos de testes.

A proposta é fazer um levantamento teórico de padrões, técnicas e metodologias usadas na engenharia de software para a documentação das especificações de requisitos. Essa documentação deve proporcionar fácil entendimento, tanto para o cliente, quanto para os desenvolvedores e usuário interno que produz os casos de testes. A não utilização de padrões e métodos, origina divergências e ambiguidades e isso se torna uma grande “bola de neve”. Um dos autores da área de engenharia de software destaca: “A falta de adoção de métodos, ferramentas e procedimentos no desenvolvimento de software e a difícil relação de entendimento entre o usuário com o desenvolvedor como possíveis causadores de tais absurdos” (PRESSMAN, 2002, p.5).

Assim, o questionamento de pesquisa que norteia este trabalho é: como melhorar, em uma empresa de desenvolvimento de software, o processo e a documentação dos requisitos, com a finalidade de facilitar e qualificar o desenvolvimento dos testes?

1.2. OBJETIVOS

A seguir é abordado o objetivo geral deste trabalho seguido pelos objetivos específicos.

1.2.1. Objetivo Geral

Propor melhorias nas especificações e no processo de levantamento de requisitos da empresa Softplan.

1.2.2. Objetivos Específicos

- ✓ Fazer um levantamento da literatura sobre engenharia de software, com ênfase em engenharia de requisitos e teste de software.
- ✓ Descrever e analisar o processo e os artefatos utilizados pela equipe da organização no processo de requisitos.
- ✓ Aplicar um questionário dentro da empresa referente aos processos utilizados e suas características.
- ✓ Descrever o processo de testes da organização.

- ✓ Sugerir melhorias no processo de requisitos a partir das teorias abordadas no desenvolvimento deste trabalho e dos questionários aplicados, bem como melhorias na especificação de requisitos.
- ✓ Descrever onde essas melhorias facilitam e qualificam o processo de teste.

1.3. JUSTIFICATIVA

O desenvolvimento de software é constantemente modificado devido às novidades tecnológicas. Um sistema de qualidade precisa passar por várias etapas dentro do desenvolvimento de software, desde o levantamento de requisitos com o cliente até a entrega final.

Para que um sistema seja considerado de qualidade, satisfaça o cliente interno e o cliente final, não é suficiente simplesmente desenvolver um sistema que atenda determinadas funcionalidades. De acordo com Ian Sommerville “A qualidade de software não implica apenas se a funcionalidade de software foi corretamente implementada, mas também depende dos atributos não funcionais de sistemas.” (SOMMERVILLE, 2011. p.457). Um sistema deve atender a todos os requisitos e funcionalidades que o cliente espera. O sistema deve ser intuitivo, prático, ágil e o produto final entregue dentro do prazo estipulado com o cliente. Para que tudo isso aconteça, todas as etapas do desenvolvimento de software devem ser completadas e realizadas de forma eficaz. Os testes devem ser executados a partir dos casos de testes realizados, e para que um caso de teste esteja entendível, a especificação de requisitos precisa estar clara.

O estudo de caso proposto nesse trabalho tem fundamental importância para a empresa Softplan, pois ao sugerir melhorias nas especificações de requisitos pode-se melhorar o entendimento para todos os demais componentes deste processo, e por consequência, melhorar todo o desempenho da equipe de desenvolvimento de software.

1.4. ESTRUTURA DA MONOGRAFIA

Este trabalho de conclusão de curso está estruturado em cinco capítulos:

- O primeiro capítulo é dividido em apresentação do tema, a problemática, objetivos, justificativa e estrutura da monografia.
- O segundo capítulo apresenta a revisão literária com conteúdo referente à engenharia de software, desenvolvimento de software (processos) especificações de requisitos, padrões, casos de teste, processos de teste e a importância do teste no desenvolvimento de software.
- No terceiro capítulo apresentamos os métodos utilizados para a elaboração desde trabalho.
- No quarto capítulo é apresentado o estudo de caso, com a descrição da organização, o processo de requisitos e de teste dentro da organização. São apresentados os questionários aplicados e a análise dos resultados obtidos. Neste capítulo são introduzidas as recomendações para a melhoria na especificação de requisitos e no processo de requisitos. E a descrição de como as melhorias sugeridas impacta no teste.
- O quinto e último capítulo contém as conclusões sobre o trabalho desenvolvido juntamente com os trabalhos futuros.

2. REVISÃO BIBLIOGRÁFICA

Este capítulo apresenta assuntos que dão o embasamento teórico para a realização deste trabalho de conclusão de curso. São utilizadas as obras bibliográficas de diversos autores que ajudaram no desenvolvimento deste trabalho.

2.1. ENGENHARIA DE SOFTWARE

Segundo Ian Sommerville (2011), a engenharia de software tem por objetivo “Apoiar o desenvolvimento profissional de software, mais do que a programação individual. Ela inclui técnicas que apoiam especificação, projeto, e evolução de programas, que normalmente não são relevantes para o desenvolvimento de software pessoal”. A engenharia de software não se trata somente da programação, mas também, de toda a documentação durante o processo de desenvolvimento de software, incluindo, inclusive, manual do usuário, sites para downloads de informações recentes do produto, entre outros. (SOMMERVILLE, 2011)

> Engenharia de Software é a área interdisciplinar que engloba vertentes tecnológica e gerencial visando abordar, de modo sistemático, os processos de construção, implantação e manutenção de produtos de software com qualidade assegurada por construção, segundo cronogramas e custos previamente definidos. (MAFFEO, 1992)

A Engenharia de software pode ser definida como sendo uma tecnologia em camadas, que passa por ferramentas, métodos, processo e foco na qualidade. A base para a engenharia de software é a camada de processos. Um processo de engenharia de software pode ser a liga que irá manter as camadas de tecnologia coesas e dessa forma possibilitando o desenvolvimento de software de forma racional e dentro do prazo. Um processo vai definir uma metodologia a ser estabelecida para a entrega de tecnologia de engenharia de software. Um processo de software constitui a base para o controle do gerenciamento de projetos de

software e estabelece o contexto no qual são aplicados métodos técnicos, estabelecidos marcos e mudanças são geridas de forma apropriada. (PRESSMAN, 2011)

2.1.1. Processo de software

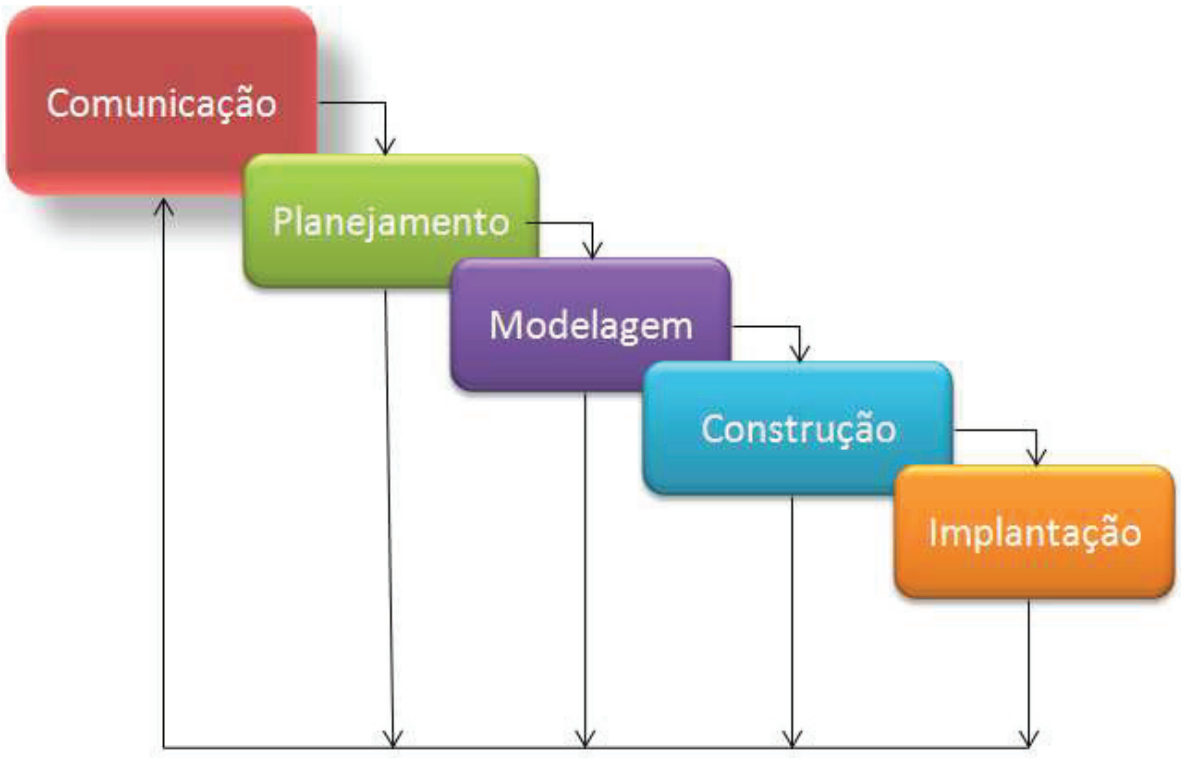
Segundo Roger Pressman (2011), um processo de software é um conjunto de atividades, ações e tarefas realizadas na criação de algum produto de trabalho. Uma atividade esforça-se para atingir um objetivo amplo e pode ser utilizada independentemente do campo de aplicação, do tamanho do projeto ou até mesmo da complexidade de esforços. De acordo com Ian Sommerville (2011, p. 18), “Um processo de software é um conjunto de atividades relacionadas que levam a produção de um produto de software.”. Ambas as definições tratam de atividades que são desenvolvidas dentro de processos de software, e essas atividades e ações podem envolver o desenvolvimento de um software do início até o fim em uma linguagem de programação padrão. (SOMMERVILLE, 2011)

Segundo James Peters (2001), em um desenvolvimento de software o engenheiro se envolve em uma sequência de atividade que produzem uma variedade de documentos, chegando a um programa satisfatório e executável. Essas atividades de engenharia englobam aquilo que conhecemos por processo de software (uma sequência de etapas com *feedback* que resultam na produção e na evolução do software).

Segundo Roger Pressman (2011), uma metodologia de processo genérica para engenharia de software estabelece cinco atividades metodológicas: comunicação, planejamento, modelagem, construção e entrega. Além disso, um conjunto de atividades de apoio são aplicadas ao longo do processo, como o

acompanhamento e controle do projeto, a administração de riscos, a garantia da qualidade, o gerenciamento das configurações, as revisões técnicas e outras.

Figura 1: Processo de Software



Fonte: PRESSMAN (2010).

No contexto geral de engenharia de software, um processo não é prescrição rígida de como desenvolver um software, é ao contrário uma abordagem adaptável que possibilita ás pessoas realizarem o trabalho de selecionar e escolher o conjunto apropriado de ações e tarefas. (PRESSMAN, 2011)

Mesmo que não exista um processo ideal de software, muitas empresas possuem lacunas nos processos de softwares, que podem ser melhorados. Os processos podem ser melhorados sendo padronizados, possibilitando melhor comunicação e reduzindo treinamentos, tornando-se mais econômico o apoio ao processo automatizado. A padronização é importante, utilizando novos métodos e técnicas de engenharia de software, como também, as boas práticas de engenharia de software. (SOMMERVILE, 2011)

Conforme cita Ian Sommerville (2011, p.24), “Processos reais de software são intercalados com sequências de atividades técnicas, de colaboração e de gerência, com o intuito de especificar, projetar, implementar e testar um sistema de software.” O processo de software possui quatro atividades básicas, especificação, desenvolvimento, validação e evolução, que são organizadas de formas diferentes, conforme o processo de desenvolvimento de software. (SOMMERVILLE, 2011)

De acordo com Wilson de Pádua Paula Filho (2001) na engenharia de software “processos podem ser definidos para atividades como desenvolvimento, manutenção, aquisição e contratação de software. Pode-se também definir subprocessos para cada um destes.”. Nesse caso é possível definir subprocessos para o desenvolvimento de software, sendo que nesses subprocessos compreendem: definição de requisitos, análise, estrutura (desenho, protótipo), implementação e testes. (PAULA FILHO, 2001). A seguir serão detalhados os subprocessos de requisitos e análise contidos no processo de desenvolvimento de software.

2.1.2. Engenharia de Requisitos

A engenharia de requisitos é uma atividade importante na engenharia de software, que se inicia no primeiro contato de definição de negócio do sistema, ou seja, nos interessados (clientes, usuários finais) ou necessidade a ser desenvolvida. Ela analisa as necessidades que o projeto e sua construção devem atender, assim como as prioridades e a ordem das atividades a serem executadas. (PRESSMAN, 2011)

De acordo com Roger S. Pressman (2011, p.127), “A engenharia de requisitos estabelece uma base sólida para o projeto e sua construção. Sem ela, o software resultante tem grande probabilidade de não atender as necessidades do cliente.”.

Segundo Felipe Nery Machado (2011) e Roger S. Pressman (2011), A engenharia de requisitos preocupa-se em buscar a qualidade e é responsável por entender da melhor maneira possível às necessidades do cliente, efetuar uma

negociação, avaliar a viabilidade, especificar os requisitos, elaborar documentos, revisar e validar os requisitos do sistema.

Conforme Felipe Nery Machado (2011, p.101),

> A literatura relacionada a requisitos evidencia uma falta de consenso na definição de termos como engenharia de requisitos, gerência de requisitos, ou análise de requisitos. Estes termos podem aparecer como sinônimos ou como componentes um dos outros.

De acordo com Roger S. Pressman (2011), a gestão ou gerência de requisitos, faz parte da engenharia de requisitos, sendo assim, neste trabalho fica definido que a engenharia de requisitos é responsável pelas atividades de: levantamento de requisitos, especificação, validação de requisitos e gerência de requisitos.

2.1.2.1. Levantamento de requisitos

Um levantamento de requisitos parece simples, basta realizar uma entrevista com os interessados, procurar saber o objetivo do sistema, bem como as funcionalidades que o sistema deve ter para atender as necessidades da empresa, porém, não é tão simples assim o levantamento de requisitos e alguns problemas comuns são encontrados nessa atividade. (PRESSMAN, 2011)

De acordo com Christel e Kang (1992, apud PRESSMAN, 2011) os problemas comuns encontrados na etapa de levantamento de requisitos são:

- Problemas de escopo: detalhes que não são necessários no levantamento de requisitos e acabam confundindo ao invés de esclarecer os objetivos do sistema.
- Problemas de entendimento: os clientes não são claros em suas necessidades, tem dificuldades de transmitir as necessidades, ou não passam todas as informações necessárias achando que algumas já estão implícitas, sendo assim, acaba resultando em requisitos ambíguos ou impossíveis de serem testados.

- Problemas de volatilidade: com o tempo os requisitos mudam, dessa forma, o levantamento de requisitos, deve ser feito de forma organizada.

Antes de iniciar um ciclo de desenvolvimento de software é imprescindível o entendimento das necessidades do cliente pelo engenheiro. O levantamento de requisitos é a principal tarefa da engenharia de software, é onde, os desenvolvedores tomam ciência das necessidades e restrições dos clientes. (THAYER, 1997 apud ALVES e FERREIRA, 2006)

De acordo com Ian Sommerville (2011, p. 57), “Os requisitos de um sistema são as descrições do que o sistema deve fazer, os serviços que oferece e as restrições a seu funcionamento.”.

De acordo com Felipe Nery Machado (2011), “O stakeholder é qualquer pessoa materialmente afetada pelo resultado do projeto: clientes, usuários diretos e indiretos, investidores, acionistas, fornecedores”.

Ian Sommerville (2011, p. 72) destaca também:

> A descoberta de requisitos (às vezes, chamada elicitação de requisitos) é o processo de reunir informações sobre o sistema requerido e os sistemas existentes e separar dessas informações os requisitos de usuários e de sistema. Fontes de informação durante a fase de descoberta de requisitos incluem documentação, *stakeholders* do sistema e especificações de sistemas similares. Você interage com os *stakeholders* por meio da observação e de entrevistas e pode usar cenários e protótipos para ajudar os *stakeholders n*o que o sistema vai ser.

As entrevistas com os *stakeholders* do sistema são parte da maioria dos processos de engenharia. É a partir das entrevistas que os analistas levantam os requisitos de software. As entrevistas podem ser fechadas ou abertas, sendo que na entrevista fechada o *stakeholder* responde a uma determinada sequência de perguntas e na entrevista aberta o *stakeholder* responde a uma série de duvidas e questionamento da equipe de engenharia de requisitos, desenvolvendo, assim, uma melhor compreensão das necessidades. (SOMMERVILLE, 2011)

Segundo James F. Peters e Witold Pedrycz (2001, p. 102), “Um requisito de software é uma descrição dos principais recursos de um produto de software, seu fluxo de informações, comportamento e atributos.”, ou seja, um requisito de software é a base para o desenvolvimento de um software. (PETERS, 2001)

Segundo Wilson de Pádua Paula Filho (2001, p. 53), “Uma boa Engenharia de Requisitos é um passo essencial para o desenvolvimento de um bom produto, em qualquer caso”.

Levantamento e documentação de requisitos são um dos problemas básicos da engenharia de software. Quanto mais bem definido é um requisito em uma documentação, maior a chance de o desenvolvedor entender o documento e realizar um bom desenvolvimento. (PÁDUA, 2001)

Wilson de Pádua Paula Filho (2001) destaca também que “Requisitos de alta qualidade são claros, completos, sem ambiguidade, implementáveis, consistentes e testáveis. Os requisitos que não apresentem essas qualidades são problemáticos”.

Os requisitos, geralmente são classificados em requisitos funcionais e não funcionais. (SOMMERVILLE, 2011)

2.1.2.1.1. Requisitos funcionais

Requisitos funcionais são explicações de como o sistema deve se comportar ou reagir em determinadas situações, e pode também, em alguns casos, descrever o que o sistema não deve fazer. (SOMMERVILLE, 2011)

André Koscianski e Michel do Santos Soares (2007) entendem como requisitos funcionais as funcionalidades que o sistema tenha depois de pronto, como entradas e saídas.

Segundo Ian Sommerville (2011, p. 59), “Os requisitos funcionais de um sistema descrevem o que ele deve fazer.” Explica também que os requisitos funcionais do sistema “variam de requisitos gerais, que abrangem o que o sistema deve fazer, até requisitos muito específicos, que refletem os sistemas e as formas de trabalho de uma organização.”.

A especificação de requisitos funcionais deve conter todos as funcionalidades solicitadas pelo usuário e não devem ser contraditórias. Quanto maior a complexidade do sistema mais difícil de atingir a consistência e completude de todos

os requisitos. Porém, a medida que forem descobertos problemas, o documento de requisitos deve ser atualizado para contemplar a solução. (SOARES e KONCIANSKI, 2007).

Os requisitos funcionais geralmente podem ser visualizados no casos de uso dos sistemas. (ENGHOLM JUNIOR, 2010).

2.1.2.1.2. Requisitos não funcionais

Os requisitos não funcionais são descritos como restrições de serviços ou de funções oferecidos pelo sistema, ou seja, restrição de *timing,* performance, sistema operacional, entre outros. Requisitos não funcionais como o nome diz, não são requisitos que descrevem a necessidade do cliente e sim relacionado à funcionalidade do sistema, propriedades, confiabilidade e tempo de resposta. Eles restringem ou especificam as características do sistema como um todo. (SOMMERVILLE, 2011)

André Koscianski e Michel do Santos Soares (2007) também entendem que requisitos não-funcionais não são diretamente ligados as funções desempenhadas no software, mas sim a restrições de forma geral. No entanto isto não impede que eles se relacionem com requisitos funcionais do sistema. "Por exemplo, limitar o tamanho do código de máquina para uma aplicação(...) pode tornar-se incompatível com o número de funções que se deseja implementar".

Kotonya e Sommerville (2000) citado em André Koscianski e Michel do Santos Soares (2007) classificam os principais tipo de requisitos não funcionais:

Quadro 1: Classificação dos principais requisitos não-funcionais.

Desempenho	Interface	Interoperabilidade
Verificação	Eficiência	Robustez
Portabilidade	Qualidade	Confiabilidade
Recursos	Segurança	Manutenibilidade

Fonte: Kotonya e Sommerville (2000) apud André Koscianski e Michel do Santos Soares (2007)

Hélio Engholm Jr (2010) também cita alguns tipos de requisitos-não funcionais e os define:

- Extensibilidade: facilidade de alterar o sistema para contemplar novas funcionalidades e adaptar mudanças de software
- Usabilidade: considerando a facilidade de uso do sistema pelo usuários
- Confiabilidade: considerando o quão confiável é a utilização do sistema e a confiabilidade das informações apresentadas
- Desempenho: relacionado ao desempenho esperado do sistema
- Escabilidade: capacidade do sistema de ser utilizado por número crescente de usuários simultaneamente
- Reusabilidade: utilização ou implementação de código reutilizável
- Capacidade de manutenção: facilidade de se manter o sistema
- Reutilização de código: facilidade de reutilizar código já desenvolvido de forma eficaz
- *Performance:* processamentos com tempo de resposta adequado às necessidades do usuário
- Eficiência no desenvolvimento: desenvolvimento sofisticado e eficaz
- Confiabilidade nos dados apresentados: confiança dos usuários de que as informações apresentadas pelo sistema encontram-se corretas.

2.1.2.1.3. Casos de Uso

À medida que o levantamento de requisitos é realizado, pode-se expressar as atividades por meio da modelagem de diagramas de casos de uso. Esses diagramas tem como objetivo representar antecipadamente o que deve existir no software muito antes de como será implementado. (TONSIG, 2003)

Para Roger S. Pressman (2011), enquanto os requisitos são levantados, um visão geral das características e funções começa a ter forma. Porém ainda é difícil evoluir nas atividades de engenharia de software sem entender de forma bem clara

a solução proposta. "Para tanto desenvolvedores e usuários podem criar um conjunto de cenários que identifique um roteiro de caso de uso para o sistema a ser construído. Os cenários normalmente são chamados de casos de uso." (PRESSMAN, 2011, p. 136).

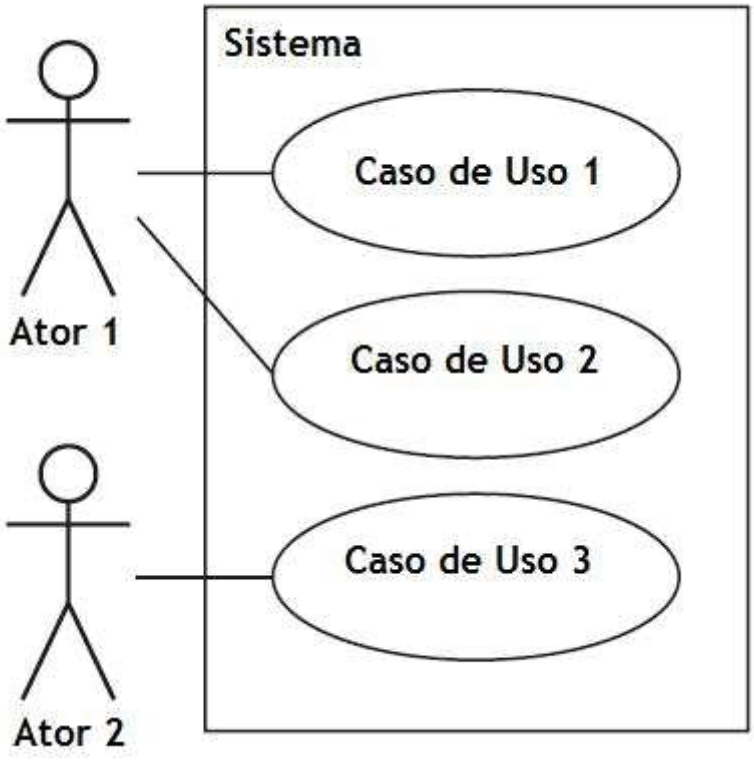
De acordo com Sérgio Luiz Tonsig (2003), o diagrama de casos de uso são aplicados para capturar os requisitos solicitados pelos clientes, tendo assim um primeiro nível de abstração acerca do sistema. Os clientes e usuários finais também tem bastante interesse nesse tipo de modelagem pois é de fácil interpretação e retrata como serão as funcionalidades do sistema.

Sérgio Luiz Tonsig (2003) afirma que a construção do caso de uso é visto como uma *caixa preta* de funcionalidades: não se sabe como vai ser feito, mas apenas o que vai ser feito. Os principais objetivos dessa representação são:

- Captar(entender) a funcionalidade necessária para resolução dos problemas existentes, sob ótica do cliente ou usuários.
- Mostrar uma visão funcional coesa sobre tudo o que o software deverá fazer, pois o diagrama será a base para todo o processo de desenvolvimento.
- Deverá ser aplicado para testes de validação (o software quando pronto realmente possui a funcionalidade inicialmente planejada).
- Propiciar facilidade para a transformação dos requisitos funcionais em classes e operações reais do software.

Os componentes deste diagrama são os atores e os casos de uso. Pode-se observar um exemplo de um diagrama de caso de uso na figura 2.

Figura 2 – Exemplo de diagrama de caso de uso



Fonte: http://www.macoratti.net/11/09/net_ioop.htm (2015).

Qualquer entidade externa ao sistema, ou seja, clientes, funcionários, equipamentos e sistemas de terceiros são considerados atores do sistema. Os casos de uso são desenvolvidos de acordo com os eventos que ocorrem entre as atividades externas e o sistema. (TONSIG, 2003 e PRESSMAN, 2011)

Os casos de uso só podem ser desenvolvidos após a definição dos atores. (PRESSMAN, 2011)

Para facilitar a escrita de casos de uso, Jacobson (PRESSMAN, 2011) define algumas perguntas a serem respondidas:

- Quem é(são) o(s) ator(es) primário(s) e o(s) ator(es) secundário(s)?
- Quais são as metas do ator?
- Que precondições devem existir antes de uma história começar?
- Que tarefas ou funções principais são realizadas pelo ator?

- Que exceções deveriam ser consideradas à medida que uma história é descrita?
- Quais são as variações possíveis na interação do ator?
- Que informações de sistema o ator adquire, produz ou modifica?
- O ator terá de informar o sistema sobre mudanças no ambiente externo?
- Que informações o ator deseja do sistema?
- O ator gostaria de ser informado sobre mudanças inesperadas?

Ainda sobre caso de uso, Sérgio Luiz Tonsig (2003) fala que este pode ser definido como um conjunto de ações que o sistema executa, geralmente acionado por um ator, e produz resultados para os objetivos do sistema. Algumas características de caso de uso:

- Um caso de uso modelo a interação entre os atores e o sistema, ou mesmo entre casos de uso.
- Um caso de uso é ativado por um ator ou por outro caso de uso para acionar uma certa função do sistema, como por exemplo "cadastrar fornecedor".
- Um caso de uso é fluxo de eventos completo e consistente (conjuntos de operações que se completam, atingindo um objetivo).
- Todos os casos de uso juntos representam todas as situações possíveis de utilização do sistema de mostram toda a funcionalidade existente disponível no sistema.

Os casos de uso auxiliam muito e são de certa forma imprescindíveis para uma boa elaboração de requisitos de um sistema. (TONSING, 2003)

2.1.2.1.4. Especificação de Software

Conforme Ian Sommerville (2011, p. 24), especificação de software é “o processo de compreensão e definição dos serviços requisitados do sistema e

identificação de restrições relativas à operação e ao desenvolvimento do sistema.” Todas as atividades de especificação de software, assim como as atividades relacionadas à manutenção e adaptação dessa especificação são chamadas de engenharia de requisitos.

O processo de engenharia de requisitos tem como objetivo a produção de um documento de requisitos que especifica um sistema em detalhes, satisfazendo o que o cliente deseja. Esses requisitos geralmente são apresentados de duas formas, com um documento em alto nível para o cliente e um documento de forma mais técnica e mais detalhada para os desenvolvedores. (SOMMERVILLE, 2011)

Engenharia de requisitos possui três atividades principais.

Estudo de viabilidade: basicamente é o estudo para informar uma decisão de avançar ou não no desenvolvimento do sistema, levando em consideração o que o cliente necessita e se é possível ser desenvolvido no âmbito das atuais restrições orçamentais.

Elicitação e Análise de requisitos: essa atividade faz uma análise de tarefas, discussões com o suposto comprador do sistema e análise de requisitos com base nos sistemas já desenvolvidos, para embasamento do que será desenvolvido.

Especificação de requisitos: faz a tradução de informações obtidas durante a atividade de análise em um documento de requisitos. Validação de requisitos: é feita a validação do documento especificado. Após descoberta de erros nessa etapa, o documento é corrigido. (SOMMERVILLE, 2011)

2.1.2.1.4.1. Documento de requisitos

Segundo Ian Sommerville (2011, p. 63), “O documento de requisitos de software, é uma declaração oficial de o que os desenvolvedores do sistema devem implementar.”.

De acordo com Roger S. Pressman (2011, p. 129), “A formalidade e o formato de uma especificação variam com o tamanho e complexidade do software a ser construído.”.

Conforme Ian Sommerville (2011), a estrutura de um documento de requisitos é composta da seguinte forma:

- Prefácio: histórico de versão do documento com resumo das mudanças.
- Introdução: Deve descrever a necessidade do sistema, funções do sistema e também como o sistema atende os objetivos de negócio do cliente.
- Glossário: Termos técnicos usados no documento.
- Definição de requisitos de usuário: Descrição da funcionalidade fornecida ao usuário. Também podem ser descritos qualquer diagrama ou notação que for relevante para o entendimento da funcionalidade.
- Arquitetura do sistema: Deve descrever uma visão geral sobre a arquitetura prevista do sistema, descrevendo a distribuição de funções entre os módulos do sistema.
- Especificação de requisitos do sistema: Descrição detalhada de requisitos funcionais e não funcionais.
- Modelos do sistema: Modelos gráficos, de fluxo de dados ou modelos semânticos.
- Evolução do sistema: Descrição de previsão de mudanças ou previsão de mudanças de necessidade.
- Apêndices: Descrição de desenvolvimento, hardware e banco de dados.
- Índice: Um ou mais índice. Índice por diagrama, funções, entre outros.

Roger S. Pressman (2011), apresenta um modelo de especificações de requisitos que pode ser estruturado da seguinte forma:

- Introdução (propósito, convenções do documento, público-alvo e sugestões de leitura, escopo do projeto e referências).
- Descrição geral (perspectiva e características do produto, classes de usuários e características, ambiente operacional, documentação para usuários, restrições de projeto e implementação e hipóteses e dependências).
- Características do sistema

- Requisitos de interfaces externas (interfaces do usuário, interfaces de hardware, interfaces de software e interfaces de comunicação).
- Outros requisitos não funcionais (necessidade de desempenho, proteção, segurança e atributos de qualidade de software).
- Outros requisitos
- Apêndice A: Glossário
- Apêndice B: Modelos de análise
- Apêndice C: Lista de problemas

Um documento de requisito vai variar de acordo com o projeto. Para um projeto mais complexo, pode-se realizar um documento mais completo e para um projeto mais simples, pode-se elaborar um documento mais enxuto. (PRESSMAN, 2011)

2.2.2.2. Verificação e validação de requisitos

Na etapa de verificação e validação de requisitos, que ocorre após a documentação de requisitos, é verificado se os requisitos estão claros, sem ambiguidade e possíveis de serem implementados e testados, é feito também, uma análise referente à qualidade do documento elaborado. (MACHADO, 2011)

De acordo com Roger S. Pressman (2011), um documento de requisitos pode ser revisado após a resposta das seguintes questões:

- ✓ Cada um dos requisitos é consistente com os objetivos globais para o sistema/produto?
- ✓ Todos os requisitos foram especificados no nível de abstração apropriado? Ou seja, algum dos requisitos fornece um nível de detalhe técnico inapropriado no atual estágio?
- ✓ O requisito é realmente necessário ou representa um recurso adicional que talvez não seja essencial para o objetivo do sistema?
- ✓ Cada um dos requisitos é limitado e sem ambiguidade?
- ✓ Cada um dos requisitos possui atribuição? Ou seja, uma fonte (em geral, um individuo específico) é indicada para cada requisito?

- ✓ Algum dos requisitos conflita com outros requisitos?
- ✓ Cada um dos requisitos é atingível no ambiente técnico que irá abrigar o sistema ou produto?
- ✓ Cada um dos requisitos pode ser testado, uma vez implementado?
- ✓ O modelo de requisitos reflete, de forma apropriada, a informação, função e comportamento do sistema a ser construído?
- ✓ O modelo de requisitos foi 'dividido' para expor progressivamente informações mais detalhadas sobre o sistema?
- ✓ Os padrões de requisitos foram utilizados para simplificar o modelo de requisitos? Todos os padrões foram validados adequadamente? Todos os padrões são consistentes com os requisitos do cliente?

Essas e outras perguntas podem ser feitas na validação de requisitos para garantir que o documento possua a qualidade necessária para atender as necessidades do cliente e servir de base para o projeto. (PRESSMAN, 2011)

2.2.2.3. Gestão de requisitos

De acordo com Ian Sommerville (2011, p. 78):

> O gerenciamento de requisitos é o processo de compreensão e controle das mudanças nos requisitos do sistema. Você precisa manter a par das necessidades individuais e manter ligações entre as necessidades dependentes para conseguir avaliar o impacto das mudanças nos requisitos. Você precisa estabelecer um processo formal para fazer propostas de mudanças e a ligação destas às exigências do sistema. O processo formal de gerenciamento de requisitos deve começar assim que uma versão preliminar do documento de requisitos estiver disponível.

Segundo Ian Sommerville, os principais aspectos que fazem parte da gerência de requisitos são:

- Gerenciar as mudanças em requisitos existentes.
- Gerenciar o relacionamento entre os requisitos.
- Gerenciar as dependências entre o documento de requisitos e outros documentos produzidos durante o desenvolvimento de software.

Após a aprovação de um documento de requisitos, havendo propostas de alteração nos requisitos, deve-se fazer o gerenciamento de mudança de requisito para estas mudanças. Esse gerenciamento, basicamente, é avaliar se os benefícios para a implementação desses novos requisitos justificam os custos dessa implementação. (SOMMERVILLE, 2011)

2.2.2.4. Rastreabilidade

Segundo Hélio Engholm Jr. (2010), a rastreabilidade serve para auxiliar na manutenção do sistema, dos testes a serem executados e os impactos que essas mudanças podem causar. A partir do momento que o processo de engenharia de requisitos inicia, deve-se começar também a rastreabilidade, para atualizar os atributos e avaliar os impactos. Para realizar uma rastreabilidade criando uma matriz de rastreabilidade, devem-se seguir os seguintes processos: objetivos, responsável por atualizar essa matriz, produtos de entrada (documentos de requisitos, documentos de visão entre outros), critérios de entrada (especificação de requisitos aprovada), produtos de saída (documento atualizado de rastreabilidade contendo os atributos atualizados), critérios de saída (matriz de rastreabilidade consistente) e atividades.

De acordo com Wilson Pádua de Paula Filho (2001), há dois tipos de rastreabilidade que devem ser analisados:

- Rastreabilidade para trás: deve ser possível saber de onde surgiram determinados requisitos, o que originou o requisito. Essa rastreabilidade ajuda para avaliar possíveis impactos e também para facilitar a interpretação.
- Rastreabilidade para frente: análise de onde os requisitos impactam no sistema. Essa rastreabilidade ajuda no desenvolvimento, no teste para garantir que todas as partes importantes impactadas por esses requisitos sejam testadas e na análise de requisitos. Sabendo estes impactos, ajuda também caso houver mudança nos requisitos, quais itens serão afetados.

Conforme Wilson Pádua de Paula Filho (2001, p. 58), “Uma especificação dos requisitos é rastreável se permite a fácil determinação dos antecedentes e consequências de todos os requisitos”.

2.2. QUALIDADE DE SOFTWARE

Segundo Roger Pressman (1995, p. 724), qualidade de software é definida como: “Conformidade a requisitos funcionais e de desempenho explicitamente declarados, a padrões de desenvolvimento claramente documentados e a características implícitas que são esperadas de todo software.” (PRESSMAN, 1995).

Qualidade de software pode ser afetada por alguns fatores, estes podem ser divididos em dois amplos grupos, o primeiro está ligado diretamente aos fatores que podem ser medidos diretamente (por exemplo: erros/unidade de tempo) e o segundo são aqueles que podem ser medidos apenas indiretamente (por exemplo: usabilidade ou manutenibilidade). Em cada caso, deve ocorrer a medição para poder aferir a qualidade. (PRESSMAN, 1995)

Segundo Roger Pressman (2011), a qualidade de software não aparece simplesmente do nada. Ela é o resultado de um bom gerenciamento de projeto e uma prática consistente de engenharia de software. O gerenciamento e a prática são aplicados no contexto de quatro grandes atividades que ajudam uma equipe de software a atingir alto padrão de qualidade de software: métodos de engenharia de software, técnicas de gerenciamento de projeto, ações de controle de qualidade e garantia da qualidade de software.

2.2.1. Garantia de qualidade de software

Conforme Roger Pressman (1995), a garantia de qualidade é uma atividade fundamental para qualquer negócio que gere produtos que são usados por outros.

Antes do século XX, a garantia da qualidade era uma responsabilidade exclusiva do artesão que desenvolvia o produto. Durante os primórdios da computação, a qualidade era uma responsabilidade exclusiva do programador.

A garantia da qualidade de software (SQA) compreende uma variedade de tarefas. Ela é uma atividade abrangente aplicada a cada passo do processo de engenharia de software, envolve procedimentos para a efetiva aplicação de métodos e ferramentas, revisões técnicas formais, estratégias e técnicas de teste, procedimentos para o controle de mudanças, procedimentos para assegurar o cumprimento de padrões e mecanismos de medição e reportagem. (PRESSMAN, 1995)

O principal fator que torna a SQA difícil é a natureza complexa da qualidade de software, um atributo dos programas de computador que é definido como “conformidade a requisitos explicitamente definidos”. Mas quando considerada de forma geral, a qualidade de software abrange muitos fatores diferentes de processo e produto e métricas relacionadas. (PRESSMAN, 1995).

2.3. TESTES

Segundo Roger Pressman (1995), a atividade de teste de software é um elemento crítico da garantia de qualidade de software e representa uma revisão da especificação, projeto e codificação.

O principal objetivo do teste é encontrar erros, e um bom teste é aquele que tem alta probabilidade de encontrar um erro. Portanto, um engenheiro de software deve projetar e implementar um sistema ou produto baseado em computador tendo em mente a “testabilidade”. Ao mesmo tempo, os próprios testes devem ter uma série de características que permitam atingir o objetivo de encontrar o maior número de erros com o mínimo de esforço. (PRESSMAN, 2011)

Segundo Myers (1976 apud PETERS 2001), teste é o processo de execução de um programa com a intenção de encontrar erros.

De acordo com Roger Pressman (1995), a importância do teste de software e sua implicação para a qualidade de software não pode ser enfatizada em demasia.

> O desenvolvimento de sistemas de software envolve uma série de atividades de produção em que as oportunidades de infecção de falhas humanas são enormes. Erros podem começar a acontecer logo no começo do processo, onde os objetivos… podem estar errônea ou imperfeitamente especificados, além de erros que venham a ocorrer em fases de projeto e desenvolvimento posteriores… Por causa da incapacidade que os seres humanos têm de executar e comunicar com perfeição, o desenvolvimento de software é acompanhado por uma atividade de garantia de qualidade. (DEUTSCH apud PRESSMAN, 1995, p. 786)

Segundo Roger Pressman (2011), o principal motivo de um software ser testado é para revelar erros cometidos inadvertidamente quando projetado e construído. Uma estratégia para teste de software é desenvolvida pelo gerente de projeto, pelos engenheiros de software e pelos especialistas em testes. O teste muitas vezes requer mais trabalho de projeto do que qualquer outra ação da engenharia de software.

> As atividades de teste são muitas vezes realizadas de maneira pouco estruturada ao final do projeto, quando não existe mais solução para os problemas. Segundo Pressman, a atividade de teste seria um dos elementos críticos da garantia da qualidade de software e pode assumir até 40% do esforço gasto em seu desenvolvimento. (CAETANO, 2011, p. 58)

Conforme James Peters (2001), o teste de software determina quando um sistema pode ser liberado e prevê um desempenho no futuro. O teste é uma fonte importante de *feedback* e fornece uma base para a interação com os participantes no projeto. O teste de software envolve muitas estratégias de teste. Elas incluem teste estático versus o teste dinâmico e um teste de caixa branca versus o teste de caixa preta. Ás vezes, o teste dinâmico é denominado teste de software ou simplesmente análise dinâmica. A análise dinâmica requer que o software seja executado com os dados de teste. Ela se baseia na utilização de provas inseridas em um programa. Os testes de caixa preta e caixa branca são apenas exemplos das classes fundamentais de abordagens ao teste de software.

Uma vez que a detecção de erros é um propulsor importante do teste de software, ela é responsável pelo aumento no interesse no desenvolvimento de conjuntos de dados de teste adequados para sensibilizar erros. Do mesmo modo, é preciso alocar uma determinada quantidade de tempo para concluir o teste, e desenvolver diferentes estratégias de teste que tendem a implementar um compromisso perfeitamente seguro entre o próprio teste e o tempo destinado a ele. (PETERS, 2001).

Segundo Melissa Barbosa Pontes (2011), os testes de Software estão muito mais relacionados com o que o testador pode fazer no seu dia-a-dia do que com

documentações e realização de atividades em seqüência. Apesar da enorme importância de ter um processo como base, é imprescindível o desenvolvimento das habilidades das pessoas, para que elas saibam realizar bem suas atividades até mesmo em empresas que não têm nenhum processo bem definido.

É importante a consciência de que testar um software exige experiência, e que isso faz a diferença no momento em que existem pressões de prazo, orçamento, ausência de ferramentas, enfim, limitações de recursos no projeto. Em suma, trata-se de uma atividade que requer um perfil próprio para o profissional, diferenciado do perfil do desenvolvedor, e que requer também muita especialização, tendo em vista a quantidade de áreas distintas que existem dentro da área de testes de software, por exemplo: Testes de Unidade, Testes de Funcionalidades, Testes de Desempenho e Testes de Segurança, dentre varias outras áreas. (PONTES, 2011)

2.3.1. Tipos de teste de software

Segundo James Peters (2001), ao levar em consideração a diversidade do teste de software existente, é vantajoso considerar os tipos de testes, á medida que se tornam disponíveis a um projetista. Isso também ajudará a identificar a abrangência de um determinado teste e explicar as principais vantagens e desvantagens, além de esclarecer o desenvolvedor sobre as suas limitações.

2.3.1.1. Teste de Caixa preta

O teste de caixa preta também pode ser denominado de teste funcional, o ponto de partida deste teste é uma especificação ou código. O teste de caixa branca é também denominado teste estrutural, nesse teste as instruções, os caminhos e as ramificações são verificados para fins de execução correta.(PETERS, James 2001)

Segundo Roger Pressman (2011), o teste caixa preta, focaliza os requisitos funcionais do software. As técnicas de teste caixa preta permitem derivar séries de condições de entrada que utilizarão completamente todos os requisitos funcionais para um programa. O teste caixa preta não é uma alternativa ás técnicas caixa branca. Em vez disso, é uma abordagem complementar, com possibilidade de descobrir uma classe de erros diferente daquela obtida com métodos caixa branca.

2.3.1.2. Teste Caixa branca

Para um entendimento mais profundo do termo teste de caixa branca, Roger Pressman (2011) diz que este teste é uma filosofia de projeto de casos de teste que usa a estrutura de controle descrita como parte do projeto no nível de componentes para derivar casos de teste. Usando métodos de teste caixa branca, o engenheiro de software pode criar casos de teste que (1) garantam que todos os caminhos independentes de um módulo foram exercitados pelo menos uma vez, (2) exercitam todas as decisões lógicas nos seus estados verdadeiro e falso, (3) executam todos os ciclos em seus limites e dentro de suas fronteiras operacionais, e (4) exercitam estruturas de dados internas para assegurar a sua validade.

2.3.1.3. Teste de Comparação

Ainda temos o teste de comparação que segundo Roger Pressman (1995), se aplica quando a confiabilidade do software é absolutamente crítica. Em tais aplicações, muitas vezes são usados hardware e software redundantes para minimizar a possibilidade de erros. Quando um software redundante é desenvolvido, equipes de engenharia de software distintas produzem versões independentes de uma aplicação usando a mesma especificação. Em tais situações, cada versão pode ser testada com os mesmos dados de teste para garantir que todas ofereçam saídas

idênticas. Então, todas as versões são executadas em paralelo com uma comparação em tempo real dos resultados a fim de se assegurar consistência.(PRESSMAN, 1995)

2.3.1.4. Teste de unidade

Conforme Roger Pressman (1995), o teste de unidade concentra-se no esforço de verificação da menor unidade de projeto de software – o módulo. Usando a descrição do projeto detalhado como guia, caminhos de controle importantes são testados para descobrirem erros dentro das fronteiras do módulo. A complexidade relativa dos testes e os erros detectados como resultado deles são limitados pelo campo de ação restrito estabelecido para o teste de unidade. O teste de unidade baseia-se sempre na caixa branca, e esse passo pode ser realizado em paralelo para múltiplos módulos.

2.3.1.5. Teste de Regressão

O principal objetivo deste teste de regressão é reexecutar automaticamente alguns testes em um software, sempre que uma pequena mudança ocorrer no produto. Existem duas atividades principais no teste de regressão segundo James Peters (2001), são elas:

- Captura de um teste para reprodução. A regra é que todos devem passar por um grupo de testes fortes.
- Comparação de novas saídas com respostas antigas para garantir que não haja mudanças indesejadas.

As duas etapas do teste são automaticamente executadas em segundo plano. Para um teste de regressão ser válido, são necessários alguns ajustes adicionais no grupo de testes. A eficiência deste teste é expressa em duas condições: (1) quão

difícil é construir e manter um grupo de testes respectivos e (2) quão confiável é o sistema de teste de regressão.

2.3.1.6. Teste de Estresse

Segundo Roger Pressman (1995), durante as primeiras etapas de teste de software, as técnicas de caixa branca e de caixa preta resultam numa avaliação cuidadosa das funções e do desempenho normais do programa. O teste de estresse é realizado para confrontar os programas com situações anormais. Essencialmente, a pessoa que realiza o teste de estresse pergunta: até que ponto podemos elevar isto antes que falhe?

O teste de estresse executa o sistema de uma forma que exige recursos em quantidade, frequência ou volume anormais. Uma variação do teste de estresse é uma técnica denominada teste de sensibilidade. Em algumas situações, uma variedade muito pequena de dados contida dentro dos limites de dados válidos para um programa pode provocar um processamento extremo e até mesmo errôneo, ou uma profunda degradação do desempenho. O teste de sensibilidade tenta descobrir combinações de dados dentro de classes de entrada válidas que possam causar instabilidade ou processamento impróprio. (PRESSMAN, 1995)

2.3.1.7. Teste de Desempenho

Os Testes de Desempenho são utilizados para determinar o desempenho amplamente definido do sistema de software como tempo de execução associado a várias partes do código, ao tempo de resposta e a utilização de dispositivo. O objetivo desse tipo de teste é identificar os pontos fracos de um sistema de software e quantificar as suas deficiências, levando, assim, a outras melhorias. (PETERS, 2001).

Segundo Roger Pressman (1995), o teste de desempenho é, ás vezes, combinado com teste de estresse e frequentemente exige instrumentação de hardware e de software. Ou seja, muitas vezes é necessário medir a utilização de recursos rigorosamente. A instrumentação externa pode monitorar intervalos de execução, registrar eventos quando eles ocorrerem e amostrar estados de máquina em base regular. Ao instrumentar um programa, o realizador de teste pode descobrir situações que levam à degradação e possível falha do sistema.

2.3.1.8. Teste de integração

Um neófito no mundo do software poderia fazer uma pergunta aparentemente legítima assim que todos os módulos tivessem sido testados no nível de unidade: “Se todos funcionam individualmente, por que se tem dúvida de que eles funcionarão quando colocados juntos:”. O problema é justamente colocá-los juntos, dados podem ser perdidos ao longo de uma interface; um módulo pode ter efeito inesperado, adverso, sobre outro; as subfunções, quando combinadas, podem não produzir a função principal esperada; uma imprecisão individualmente aceitável pode ser ampliada até níveis inaceitáveis; as estruturas de dados globais podem apresentar problemas. Infelizmente, a lista não se encerra aqui. (PRESSMAN, 1995)

Segundo Roger Pressman (1995), o teste de integração é uma técnica sistemática para a construção da estrutura de programa, realizando-se, ao mesmo tempo, testes para descobrir erros associados a interfaces. O objetivo é, a partir dos módulos testados no nível de unidade, construir a estrutura de programa que foi determinada pelo projeto.

Conforme Roger Pressman (1995), frequentemente, existe uma tendência para tentar a integração não-incremental; ou seja, construir o programa usando uma abordagem, “big bang”. Todos os módulos são combinados antecipadamente. O programa completo é testado como um todo. E o caos normalmente é o resultado! Um conjunto de erros é encontrado. A correção é difícil porque o isolamento das causas é complicado pela vasta amplitude do programa inteiro. Assim que esses

erros são corrigidos, surgem novos erros e o processo continua de um modo aparentemente infindável.

A integração incremental é a antítese da abordagem big bang, O programa é construído e testado em pequenos segmentos, onde os erros são mais fáceis de ser isolados e corrigidos; as interfaces têm maior probabilidade de ser testadas completamente; e uma abordagem sistemática ao teste pode ser aplicada. (PRESSMAN, 1995)

2.3.2. Processo de Teste

Em um excelente livro sobre teste de software, Glen Myers (1979) apud Roger Pressman (1995), estabelece uma série de regras que podem servir muito bem como objetivos de teste:

- A atividade de teste é o processo de executar um programa com a intenção de descobrir um erro.
- Um bom caso de teste é aquele que tem uma elevada probabilidade de revelar um erro ainda não descoberto.
- Um teste bem-sucedido é aquele que revela um erro ainda descoberto.

Os objetivos acima implicam uma mudança drásticas de ponto de vista. Eles apontam contrariamente ao ponto de vista comumente defendido de que um teste bem-sucedido é aquele em que nenhum erro é encontrado. Nosso objetivo é projetar testes que descubram sistematicamente diferentes classes de erros e façam-no com uma quantidade de tempo e esforço mínimos.(PRESSMAN, 1995)

No teste de um software podemos tomar como base algumas atividades chaves que podem ser descritas como parte do processo são elas: Planos de teste; Projeto de teste; Casos de teste; Procedimento de teste; Execução de teste; Relatório de Teste. Um plano de teste indica a abrangência, a abordagem, os recursos e a programação da atividade de teste. Nessa etapa, é preciso indicar o que deve ou não ser testado e quais tarefas devem ser executadas. Além disso, é necessário identificar as fontes e os níveis de risco durante o teste. O planejamento pode começar assim que os requisitos estejam aprovados. É muito difícil determinar

o momento de interrupção do teste ou quando um número razoável de falta é detectado. Por esses motivos, os critérios devem ser fornecidos como uma diretriz para a conclusão do teste. Os projetos de teste aperfeiçoam a abordagem em um plano de teste. (PETERS, 2001).

Conforme James Peters (2001), os projetos de teste também identificam características específicas a serem testadas pelo projeto e definem os casos de teste associados. É preciso se esforçar para obter uma coleção de casos de teste mais compacta que continue a atender aos objetivos. Os bons casos de teste possuem uma grande probabilidade de detectar erros não descobertos. Um procedimento de teste identifica todas as etapas necessárias para operar o sistema e exercitar os casos de teste especificados para implementar o projeto de teste já definido. A execução do teste é o exercício dos procedimento de teste. A execução começa em nível de componente até a integração, o sistema e o nível de aceitação. Um relatório de teste resume todos os resultados e destaca as discrepâncias detectadas. As atividades de teste são distribuídas por todo o ciclo de vida do software. (PETERS, 2001).

2.3.3. Plano de teste

Segundo Andre Koscianski e Michel dos Santos Soares (2007), uma atividade de testes bem organizada pressupõe planejamento. O plano de teste facilita a comunicação entre os envolvidos no desenvolvimento do software ao propor um padrão de referência a ser seguido.

O padrão apresentado a seguir é uma adaptação do padrão de plano de teste do IEEE (1998), abrangendo o planejamento, especificação e documentação dos testes. Caso sejam necessários mais seções estas devem estar incluídas após a última. (KOSCIANSKI, SOARES, 2007)

2.3.3.1. Padrão IEEE 829-1998 para planos de teste

Conforme Andre Koscianski e Michel dos Santos Soares (2007), o plano de teste no Padrão IEEE (1998), pode ser visto da seguinte maneira:

Propósito: descreve o escopo, os recursos, a abordagem e o tempo alocado para as atividades de teste. Identifica os itens e funcionalidades a serem testados, os responsáveis e os riscos.

Identificador: associa um identificador único ao plano de teste específico.

Introdução: resume os itens e funcionalidades a serem testados. Pode haver referências a outros documentos do processo de desenvolvimento.

Itens: identifica os itens a serem testados, incluindo versão.

Funcionalidades: identifica as funcionalidades que serão testadas ou não assim como os motivos.

Abordagem: especifica as principais atividades, técnicas e ferramentas usada para o teste das funcionalidades. O detalhamento deve ser suficiente para permitir identificação das principais tarefas de teste e a estimativa de tempo para cada uma. As restrições significativas, como recursos e prazos, devem ser identificadas.

Critérios de aceite: especifica os critérios de aceite para cada item.

Suspensão: especifica os critérios para suspender parte ou toda a atividade de teste. Especifica as atividades que devem ser repetidas quando o teste for retomado.

Produtos: identifica os documentos produzidos, como planos, procedimentos, logs e relatórios.

Tarefas de teste: identifica as atividades necessárias para preparar e executar os testes, bem como todas as dependências entre as tarefas.

Ambiente: identifica as necessidades do ambiente, como características do hardware, software de comunicação, nível de segurança e ferramentas de auxílio.

Responsabilidades: identifica os grupos responsáveis por gerenciar, projetar, executar, verificar e resolver os testes. Identifica os grupos responsáveis pelo ambiente e pelos itens de teste.

Treinamento: especifica as necessidades de treinamento e identifica as opções.

Cronograma: identifica as atividades e os prazos de conclusão. Para cada recurso, como pessoas e ferramentas, especifica os períodos de alocação.

Riscos: identifica os maiores riscos e os planos de contingência.

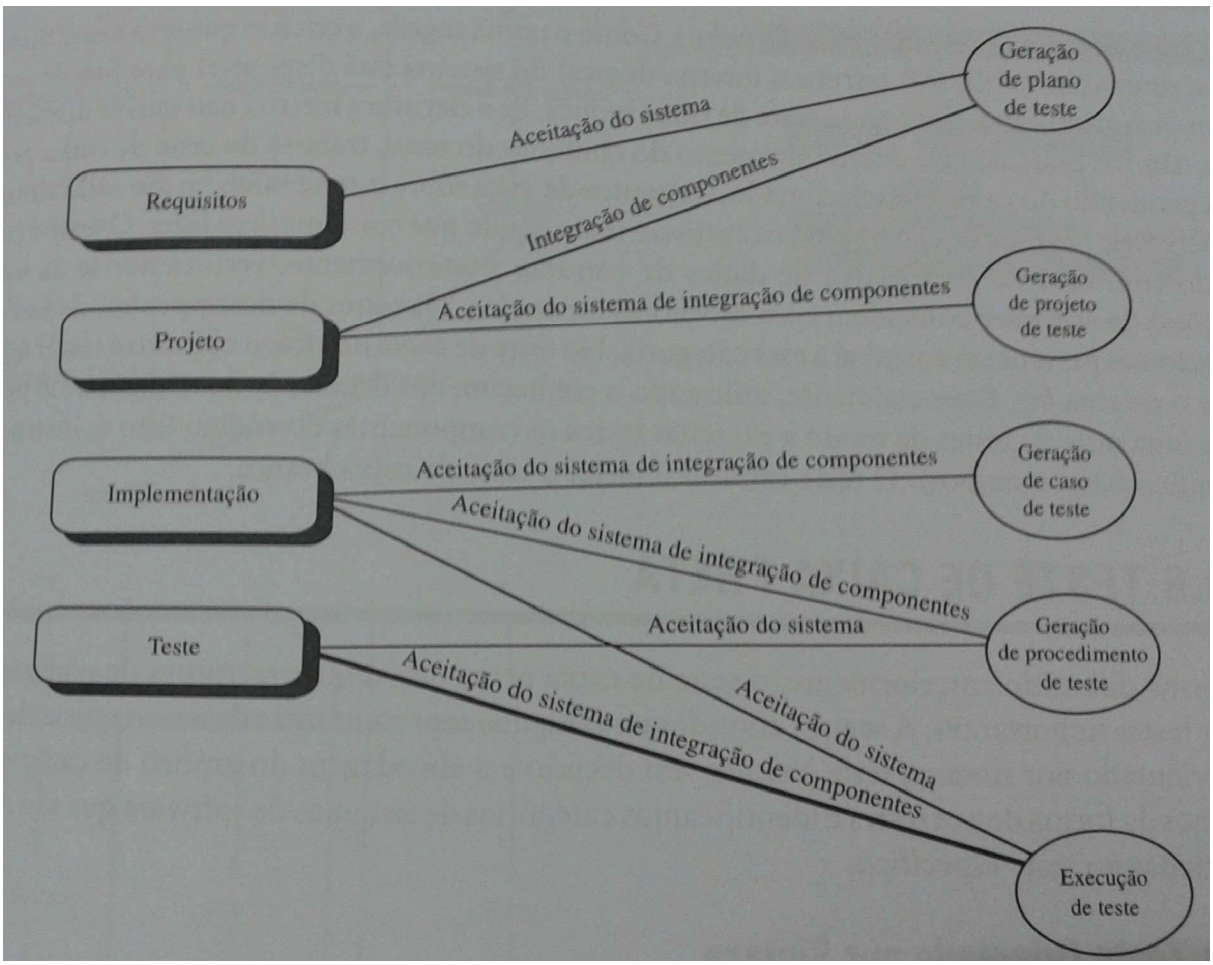
Aprovações: especifica os nomes e os cargos dos responsáveis por aprovar o plano. Devem ser inclusos espaços para assinaturas e datas. (KOSCIANSKI, SOARES, 2007)

2.3.4. Caso de teste

O Projeto de teste de software e de outros produtos trabalhados por engenharia pode ser tão desafiador quanto o projeto inicial do próprio produto. Contudo, por razões que já foram discutidas, os engenheiros de software muitas vezes tratam a atividade de teste como uma reflexão tardia, desenvolvendo casos de teste que podem “parecer certos”, mas que apresentam pouca garantia de estar completos. Relembrando os objetivos da atividade de teste, devemos projetar testes que tenham a mais alta probabilidade de descobrir a maioria dos erros com uma quantidade mínima de tempo e esforço. (PRESSMAN, 1995)

Os casos de teste e os procedimentos de teste são elaborados na fase de implementação. É preciso se esforçar para obter uma coleção de casos de teste (baterias) mais compacta (menor) que continue a atender aos objetivos. Os bons casos de teste possuem uma grande probabilidade de detectar erros não descobertos. Um procedimento de teste identifica todas as etapas necessárias para operar o sistema e exercitar os casos de teste especificados para implementar o projeto de teste já definido. A execução do teste é o exercício dos procedimentos de teste. Um relatório de teste resume todos os resultados e destaca as discrepâncias detectadas. As atividades de teste são distribuídas por todo o ciclo de vida do software assim como pode-se ver na Figura 3. (PETERS, 2001)

Figura 3 – Atividades de software



Fonte: James Peters, 2001.

Segundo Arilo Cláudio Dias Neto (2014), A atividade de teste é composta por alguns elementos essenciais que auxiliam na formalização desta atividade. Esses elementos são os seguintes:

• **Caso de Teste:** descreve uma condição particular a ser testada e é composto por valores de entrada, restrições para a sua execução e um resultado ou comportamento esperado (CRAIG e JASKIEL, 2002 apud DIAS NETO, 2014).

• **Procedimento de Teste:** é uma descrição dos passos necessários para executar um caso (ou um grupo de casos) de teste (CRAIG e JASKIEL, 2002 apud DIAS NETO, 2014).

• **Critério de Teste:** serve para selecionar e avaliar casos de teste de forma a aumentar as possibilidades de provocar falhas ou, quando isso não ocorre, estabelecer um nível elevado de confiança na correção do produto (ROCHA et al., 2001 apud DIAS NETO, 2014). Os critérios de teste podem ser utilizados como:

O **Critério de Cobertura dos Testes:** permitem a identificação de partes do programa que devem ser executadas para garantir a qualidade do software e indicar quando o mesmo foi suficientemente testado (RAPPS e WEYUKER, 1982 apud DIAS NETO, 2014). Ou seja, determinar o percentual de elementos necessários por um critério de teste que foram executados pelo conjunto de casos de teste.

O **Critério de Adequação de Casos de Teste:** Quando, a partir de um conjunto de casos de teste T qualquer, ele é utilizado para verificar se T satisfaz os requisitos de teste estabelecidos pelo critério. Ou seja, este critério avalia se os casos de teste definidos são suficientes ou não para avaliação de um produto ou uma função (ROCHA et al., 2001 apud DIAS NETO, 2014).

O **Critério de Geração de Casos de Teste:** quando o critério é utilizado para gerar um conjunto de casos de teste T adequado para um produto ou função, ou seja, este critério define as regras e diretrizes para geração dos casos de teste de um produto que esteja de acordo com o critério de adequação definido anteriormente (ROCHA et al., 2001 apud DIAS NETO, 2014).

Como visto anteriormente Roger Pressman (1995), focaliza que os testes de caixa branca focalizam a estrutura de controle do programa. Os casos de teste são derivados para garantir que todas as instruções do programa tenham sido exercitadas pelo menos uma vez durante os testes e que todas as condições lógicas tenham sido exercitadas. Os testes de caminho básico, uma técnica de caixa branca, faz uso de grafos de programa para derivar o conjunto de testes linearmente independentes que garantirá cobertura. Os testes de fluxo de dados e das condições põem à prova lógica do programa, e os testes de laços complementam outras técnicas de caixa branca ao proporcionar um procedimento para exercitar laços de vários graus de complexidade.

Conforme Hetzel (1984) citado por Roger Pressman (1995), os testes de caixa branca são definidos como “Teste de pequeno porte” *(testing in the small)*. A implicação dessa colocação é que os testes de caixa branca considerados são tipicamente aplicados a componentes de programa pequenos (por exemplo,

módulos ou pequenos grupos de módulos). Os testes de caixa preta, por outro lado, ampliam o nosso foco e poderiam ser denominados “testes em grande porte” (*testing in the large).*(PRESSMAN, 1995)

Os testes de caixa preta são projetados para validar os requisitos funcionais, sem se preocupar com o funcionamento interno de um programa. As técnicas de teste de caixa preta concentram-se no domínio de informações do software, derivando os casos de teste ao dividir a entrada e a saída de uma maneira que proporcione uma satisfatória cobertura de teste. O particionamento de equivalência divide o domínio de entrada em classes de dados que provavelmente exercitarão uma função de software específica. A análise do valor de fronteira prova a capacidade que um programa tem de manipular dados nos limites da aceitabilidade. O grafo de causa-efeito é uma técnica que possibilita que o analista valide conjuntos complexos de ações e condições. (PRESSMAN, 1995)

Os projetistas de software experientes frequentemente dizem: “A atividade de teste nunca termina; ela apenas é transferida do projetista [você] para o seu cliente. Toda vez que o seu cliente usa o programa, um teste é realizado”. Ao aplicar o projeto de casos de teste, o engenheiro de software pode realizar testes mais completos e, portanto, descobrir e corrigir o maior número de erros possível antes que os “testes do cliente” se iniciem. (PRESSMAN, 1995)

2.4. METODOLOGIA ÁGIL SCRUM

O Scrum é um *framework* para desenvolver e manter produtos complexos, seus princípios estão de acordo com o manifesto ágil e são utilizados para auxiliar um processo de desenvolvimento de software incorporando as atividades de requisitos, análise, projeto, evolução e entrega. Essas atividades possuem tarefas que são divididas por um padrão de projeto, ou seja, por um Sprint. O número de Sprint varia de acordo com o tamanho do projeto e, em alguns casos esse número muda no decorrer do processo, sendo decidido pela equipe Scrum. (PRESSMAN, 2011)

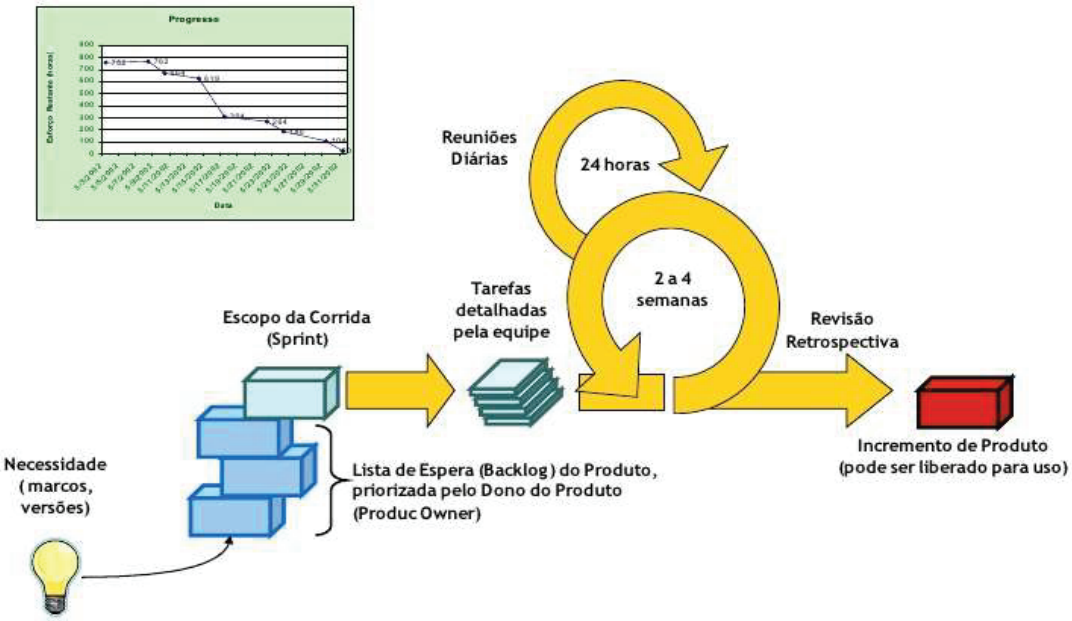
De acordo com Roger S. Pressman (2011, p. 95), “O Scrum engloba um conjunto de padrões de processos enfatizando prioridades de projeto, unidades de trabalho compartimentalizadas, comunicação e feedback frequente por parte dos clientes.”.

Segundo Roger S. Pressman (2011), cada padrão de processo possui um conjunto de ações de desenvolvimento:

1. **Backlog:** Uma lista de prioridades de necessidades que dão origem às alterações.
2. **Sprints:** Uma equipe para atender uma prioridade do backlog e entregar em um determinado prazo.
3. **Backlog work itens:** Permite que os membros da equipe trabalhem em um ambiente de curto prazo.
4. **Reuniões Scrum:** São reuniões curtas (de poucos minutos, normalmente 15 minutos) feitas rapidamente uma vez por dia, onde todos da equipe respondem algumas perguntas referentes ao andamento do projeto.

Na figura 4 pode-se observar como funciona o fluxo de um processo Scrum, as necessidades são inseridas no backlog, na sprint são selecionadas algumas dessas necessidades e as tarefas são realizadas em 2 a 4 semanas, as reuniões diárias são realizadas e após revisão retrospectiva, a funcionalidade está liberada para uso. (PRESSMAN, 2011)

Figura 4 – Fluxo de um processo Scrum



Fonte: http://pt.slideshare.net/manoelp/os-eis-e-os-perplexos-manoel-pimentel-jorge-diz-presentation, 2015

Na figura 4 além do fluxo de atividades do Scrum, direcionado pela Sprint, também se observa o gráfico de Burndown, no canto superior esquerdo, que pode ser usado para acompanhamento das atividades.

2.5. CONSIDERAÇÕES FINAIS DO CAPÍTULO

Este capítulo apresentou todos os conceitos de que precisamos para defender nosso principal objetivo do trabalho, que está focado no estudo da melhoria das especificações de requisitos.

Apresentam-se os conceitos básicos de Engenharia de Software, qualidade de software, engenharia de requisitos e testes.

3. PROCEDIMENTOS METODOLÓGICOS

Segundo Cleverson Leite Bastos e Vicente Keller (1991, p. 38) “Método é um procedimento de investigação e controle que se adota para o desenvolvimento rápido e eficiente de uma atividade qualquer. Não se executa um trabalho sem a adoção de algumas técnicas e procedimentos norteadores da ação.”.

Neste capítulo são descritos os métodos utilizados para alcançar os objetivos propostos neste Trabalho de Conclusão de Curso, além de apresentar o cronograma de planejamento deste trabalho. O cronograma das atividades encontra-se no Apêndice A.

3.1. CARACTERIZAÇÃO DO TIPO DE PESQUISA

De acordo com Cleverson Leite Bastos e Vicente Keller (1991, p. 55) “A pesquisa científica é uma investigação metódica acerca de um assunto determinado com o objetivo de esclarecer aspectos do objeto em estudo. O que poderia diferenciar a pesquisa de um estudando e de um cientista é basicamente o seu alcance ou grau.”.

Conforme Aidil Barros e Neide Lehfeld (2000, p. 70) “Segundo as formas de estudo do objeto de pesquisa, esta pode ser classificada em pesquisa descritiva (bibliográfica ou de campo), pesquisa experimental e pesquisa-ação”.

A pesquisa também possui duas tipologias, segundo Aidil Barros e Neide Lehfeld (2000) “no tocante aos fins ou destinação da pesquisa, encontramos a seguinte tipologia: a pesquisa pura e a pesquisa aplicada.”.

De acordo com Vilson Leonel e Alexandre de Medeiros Motta (2007, p. 99), “se classificarmos as pesquisas levando em conta a abordagem, teremos dois grupos: quantitativa e qualitativa.”.

Este Trabalho de Conclusão de curso é descrito como sendo uma pesquisa descritiva do tipo bibliográfica, de tipologia aplicada e de abordagem qualitativa.

3.1.1. Pesquisa Descritiva

A pesquisa descritiva o pesquisador descreve o objeto de pesquisa. Procura descobrir suas características, causas, relações e conexões com outros fenômenos. A pesquisa descritiva engloba dois tipos: pesquisa bibliográfica e pesquisa de campo. (BARROS e LEHFELD, 2000).

3.1.1.1. Pesquisa Bibliográfica

É a pesquisa que tenta resolver um problema ou buscar conhecimentos a partir de informações advindas de material gráfico, sonoro e informatizado. Nessa pesquisa, é fundamental que o pesquisador faça levantamento dos temas e tipos de abordagens já trabalhadas por outros estudiosos, sendo assim, é importante levantar e selecionar conhecimentos já catalogados em bibliotecas, editoras, internet e etc. (BARROS e LEHFELD, 2000)

3.1.2. Tipologia Aplicada

A pesquisa aplicada consiste em investigar e conhecer a necessidade do objeto de pesquisa a fim de proporcionar um resultado imediato e com o intuito de solucionar um problema encontrado na realidade. (BARROS e LEHFELD, 2000)

3.1.3. Abordagem Qualitativa

Conforme Vilson Leonel e Alexandre de Medeiros Motta (2007, p. 108), “O principal objetivo da pesquisa qualitativa é o de conhecer as percepções dos sujeitos pesquisados acerca da situação-problema, objeto da investigação.”.

Segundo D’Ambrosio (2004, apud LEONEL e MOTTA 2007, p.108):

> A pesquisa qualitativa requer do pesquisador uma atenção muito maior às pessoas e às suas ideias, procurando fazer sentido de discursos e narrativas que estariam silenciosas, tendo como foco entender e interpretar dados e discursos, mesmo quando envolve grupos de participantes e ficando claro que ela (a pesquisa qualitativa) depende da relação entre o observador e o observado.

3.2. ETAPAS METODOLÓGICAS

Para a realização deste Trabalho de Conclusão de curso são abordadas as etapas metodológicas a seguir:

- Revisão bibliográfica sobre Engenharia de Software: Especificações de requisitos e testes.
- Estudo de caso:
  a) Descrição da empresa, da equipe e/ou do produto estudo de caso. Descrição dos processos e documentos de requisitos e de testes adotados pela equipe/projeto selecionado.
  b) Aplicação de questionários para analistas de testes que utilizam especificações de requisitos para elaborar casos de testes e demais atividades relacionadas ao teste de software.
  c) Criação de um modelo para especificação de requisitos conforme a teoria estudada e as peculiaridades da empresa estudo de caso.

d) Avaliação desse modelo: aplicação de questionários e entrevistas com analistas de negócios/requisitos e de testes

e) Recomendações e conclusões.

3.3. PROPOSTA DA SOLUÇÃO

A proposta desse trabalho é apresentar melhorias no processo de requisitos e nas especificações de requisitos utilizadas nesse processo, com essas melhorias facilitar a elaboração dos casos de testes, possibilitando também, um melhor entendimento da solução para os demais processos dentro do desenvolvimento de software.

Para solucionar o problema descrito será definido um padrão de modelo de especificação de requisitos. Essa definição de um modelo vai se basear no referencial teórico, bem como, no resultado dos questionários aplicados. Dessa forma os casos de testes serão escritos de forma mais clara, diminuindo eventuais dúvidas e dificuldades e, assim, cumprindo o objetivo de facilitar a elaboração dos casos de testes, proporcionando o melhor entendimento e clareza na leitura desses documentos.

3.4. DELIMITAÇÕES

O trabalho tem como foco principal a sugestão de melhorias nas especificações de requisitos e no processo de requisitos que são analisadas no estudo de caso, com isso facilitar e qualificar os casos de testes a partir de alterações na especificação de requisitos.

Não será sugerido um novo documento de requisitos, ou seja, serão utilizados os mesmos artefatos de requisitos, porém com melhorias sugeridas.

4. ESTUDO DE CASO

Este capítulo apresenta o estudo de caso, com a descrição do processo de software da empresa. O processo de software e o estudo de caso descrito foi realizado com o consentimento da empresa citada, o termo de compromisso pode ser encontrado no apêndice H. Apresenta também uma pesquisa realizada, com colaboradores da empresa, em forma de questionário, sobre os processos de requisitos e de teste dentro da empresa. Finalmente são apresentadas modificações nos artefatos e no processo de requisitos da empresa, visando qualificar o processo de testes.

4.1. FERRAMENTAS E TECNOLOGIAS UTILIZADAS

No desenvolvimento deste trabalho foi utilizada a notação BPMN (*Bussiness Process Model and Notation*) para a descrição dos processos. Segundo OMG (2015) esta é uma notação para modelar e gerenciar os processos de negócio das organizações.

Para a elaboração deste trabalho também foram utilizadas algumas ferramentas para auxílio, são elas:

- *Highcharts*: para a elaboração dos gráficos em forma de pizza.
- Bizagi: para redesenhar os processos sugeridos.
- Google Formulário: para a criação dos questionários aplicados aos funcionários.

Figura 5. Ferramentas utilizadas

Fonte: das autoras, 2015

4.2. SOFTPLAN

A Softplan é uma empresa de tecnologia que desenvolve sistemas de gestão para instituições privadas e públicas, desde 1990. A empresa hoje, conta com mais de 2.300 clientes e 1.500 funcionários. (SOFTPLAN, 2015a)

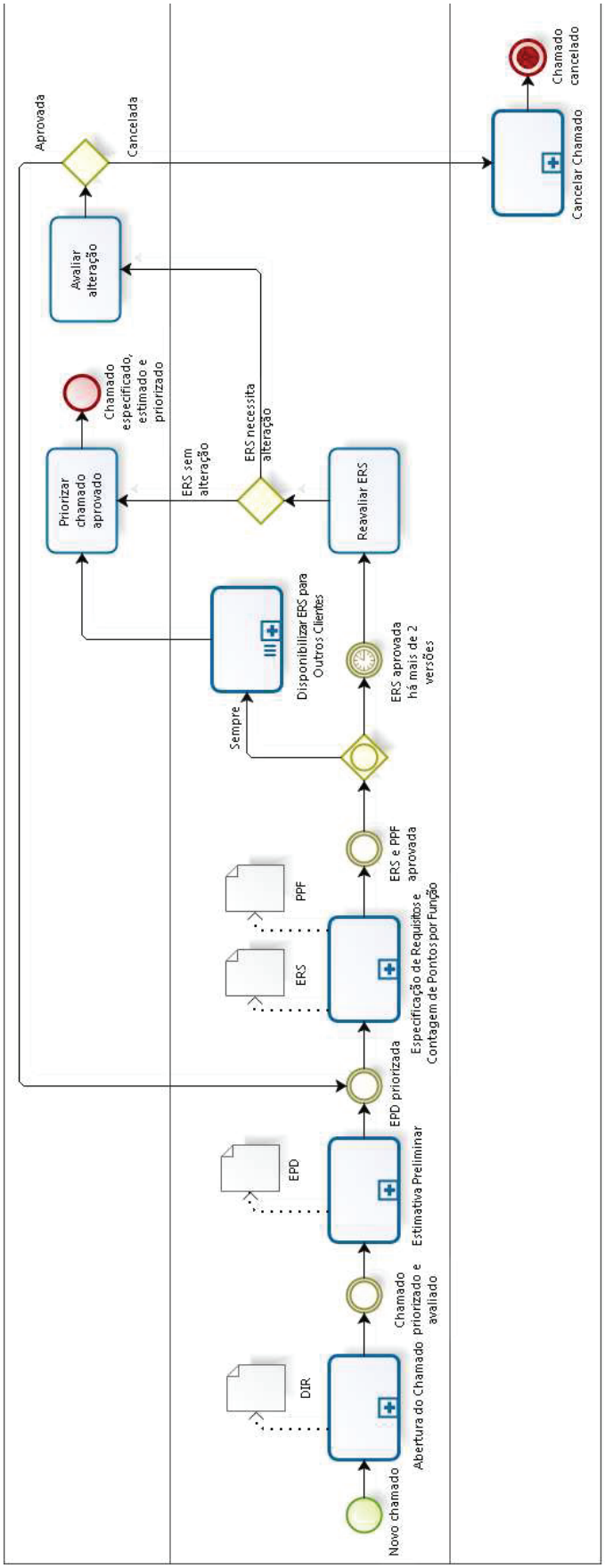
A empresa é dividida em três unidades: unidade da gestão pública, unidade da justiça e unidade da indústria da construção.

Conforme Softplan (2015a) em 1990, quando a empresa foi criada, deu-se início a unidade da indústria da construção, com softwares para a gestão de empresas privadas de construção civil. Em 1993 é iniciada a unidade da justiça, contratada pelo tribunal de justiça de Santa Catarina. Atualmente, a unidade da justiça trabalha com sistemas desktops para as Procuradorias, Tribunais de Justiça e Ministério Público. Em 1994, nasceu à unidade da gestão pública, com softwares para a gestão para Administração Física, Financeira e Contábil de Projetos cofinanciados por organismos internacionais.

4.3. PROCESSO DE GESTÃO DE PRODUTO DA EMPRESA

O processo descrito neste trabalho é o processo de manutenção evolutiva da gestão de produto utilizado na Unidade da Justiça – Softplan, o qual está em fase de implantação. Será trabalhado considerando este processo, analisando seu conteúdo e sugerindo melhorias no processo e nos artefatos que fazem parte deste processo. É apresentado também neste trabalho o processo de testes da empresa. A figura 6, a seguir, retrata o processo de manutenção evolutiva da parte de gestão de produto.

Figura 6. Processo de manutenção evolutiva – Gestão de produto



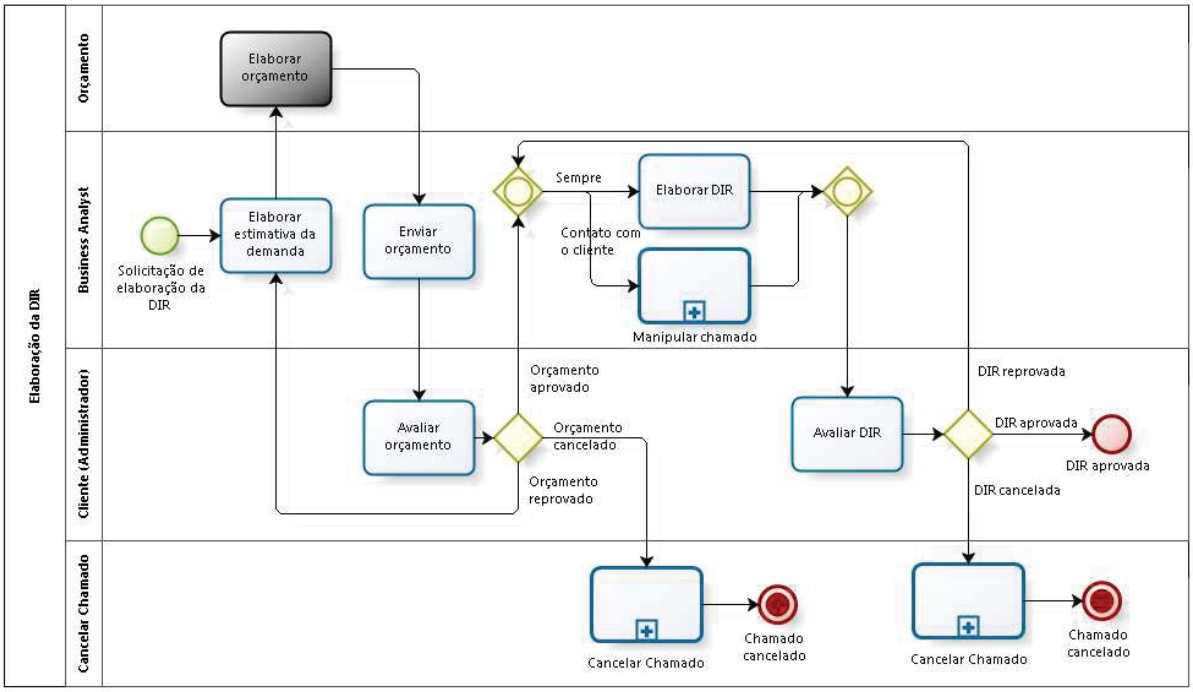
Fonte: SOFTPLAN, 2015b

Podemos verificar na figura 6 o fluxo do processo de manutenção evolutiva atual da empresa, nele são apresentadas todas as atividades e documentos gerados durante o processo.

4.3.1. Processo de Levantamento de Requisitos

O processo começa com a abertura de uma solicitação (chamado) de melhoria no sistema. É preenchido um registro chamado de DIR (Documento Inicial de Requisitos) o modelo deste documento se encontra no Anexo A desta monografia. Nesse documento será feito um registro detalhado sobre a solicitação. Na estrutura do documento é incluído: uma introdução referente à solicitação, as pessoas envolvidas no levantamento das informações, o escopo, o que ficará fora do escopo e detalhes das revisões (quem o elaborou, quem o revisou e etc). O preenchimento do DIR é feito pelo cliente, ou o mesmo pode solicitar que seja preenchido pela empresa, nesse caso, o analista de negócios primeiramente faz uma estimativa de custos, avaliando o esforço para a elaboração da DIR e envia o orçamento ao cliente. Se o cliente aprovar, então o analista de negócios elabora a DIR, caso contrário, a solicitação é cancelada. Na figura 7 a seguir é apresentado o processo de elaboração da DIR.

Figura 7. Elaboração da DIR



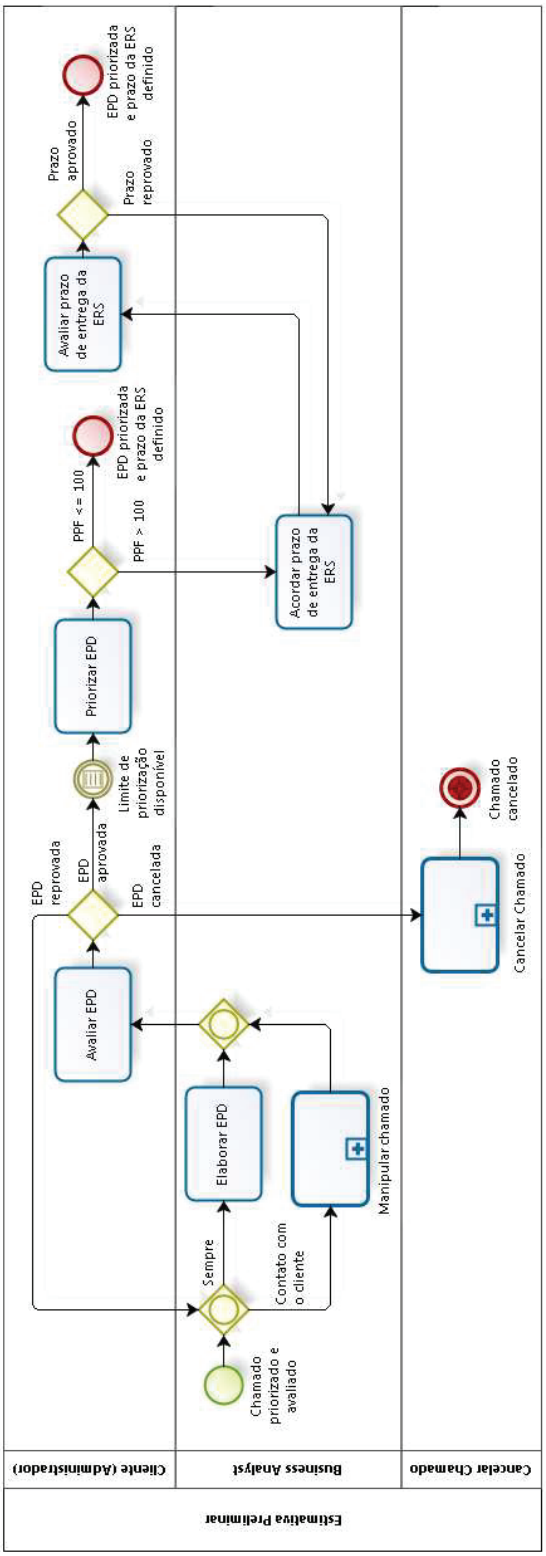
Fonte: SOFTPLAN, 2015b

Após o aceite do cliente a solicitação é priorizada, estimada e em seguida o analista de negócios realiza o EPD (Estimativa Preliminar de Demanda), este documento pode ser consultado no Anexo B desta monografia. Esse documento é uma pré-estimativa de contagem de ponto de função, que vai informar ao cliente um intervalo de pontos de função que serão atribuídos à futura especificação da solicitação aberta e em quanto tempo para a produção dessa especificação.

Ao finalizar a elaboração do documento de EPD, esse documento é enviado ao cliente que vai aprovar, reprovar ou cancelar. Ao ser reprovado o EPD volta ao analista de negócios para que sejam feitos os ajustes necessários. Quando o documento é cancelado, a solicitação também é cancelada. Se o EPD é aprovado pelo cliente, a solicitação é priorizada, levando em consideração o total de pontos de função pré estimado. Essa priorização é realizada usando a técnica de Pontos por função e ela é registrada no documento denominado PPF (Pontos Por Função). O template deste documento se encontra no Anexo C desta monografia. Se o resultado da estimativa for menos de 100 pontos de função então a solicitação é encaminhada para a elaboração da especificação de requisitos dentro do prazo estipulado no EPD. Se o resultado for mais de 100 pontos de função, o analista de negócios é responsável por acordar com o cliente o prazo para a elaboração da

especificação de requisitos. A seguir é apresentado o processo de estimativa preliminar de demanda.

Figura 8. Estimativa Preliminar de Demanda



Fonte: SOFTPLAN, 2015b

Após o EPD ser aprovado e com o prazo da ERS (Especificação de Requisitos de Software) definido, o analista de negócios avalia qual ciclo de

usabilidade será aplicado nessa especificação, em seguida, encaminha para o analista de sistemas fazer a elaboração da ERS o processo da elaboração de ERS pode ser visto na figura 9. Durante a realização desse documento o analista de sistema pode manter contato com o cliente para eventuais esclarecimentos.

Em paralelo a elaboração da especificação de requisitos, o analista de sistemas realiza o ciclo de usabilidade definido pelo analista de negócios com o apoio do usabilista. O analista de negócios pode definir um dos três ciclos de usabilidade: expresso, básico e o completo.

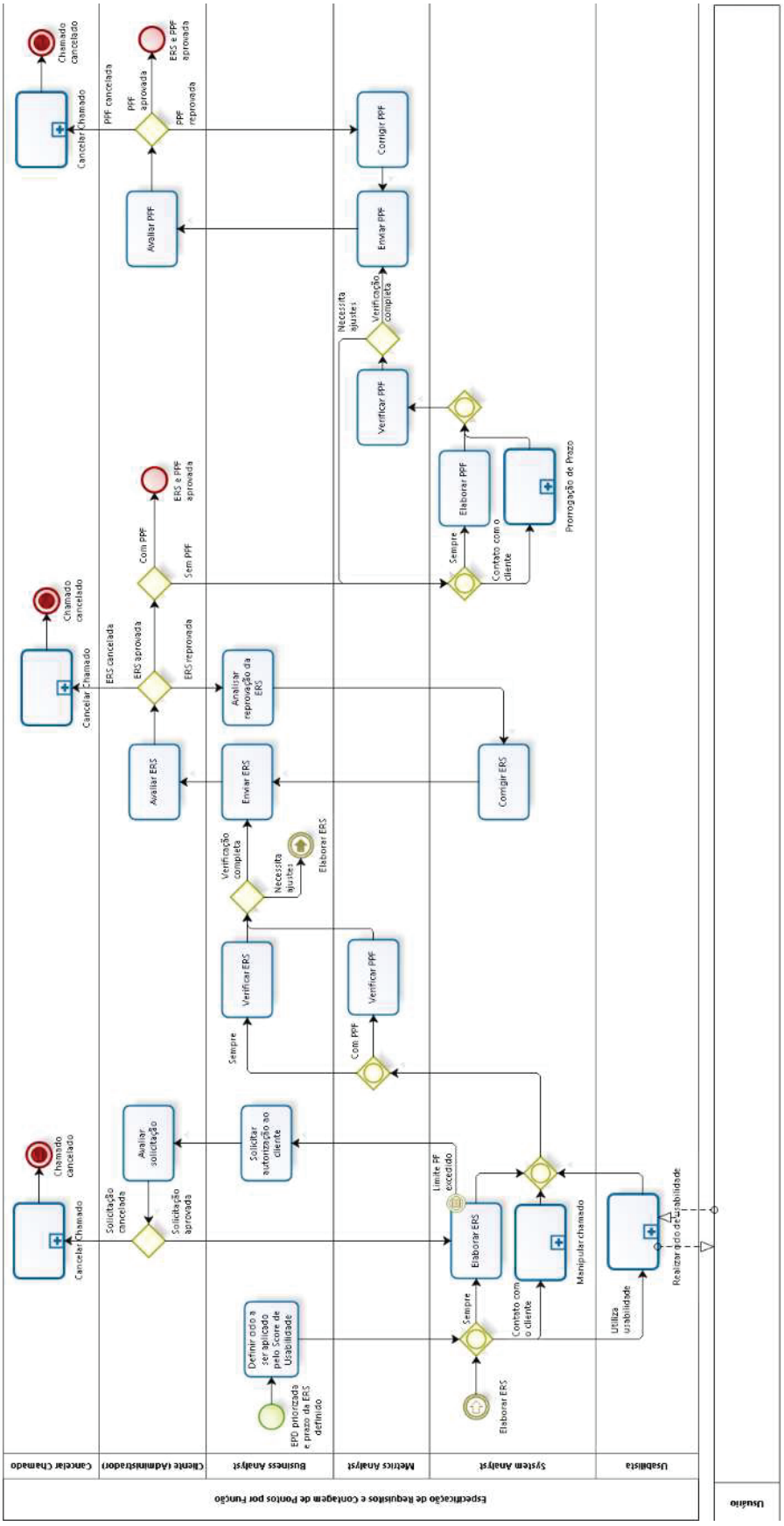
- Ciclo Expresso: esse ciclo é o mais rápido do processo de usabilidade, não requer a participação do usuário, somente a elaboração dos protótipos com base na necessidade do usuário.
- Ciclo Básico: neste ciclo são realizadas entrevistas e testes com os usuários para melhor realização dos protótipos.
- Ciclo Completo: neste ciclo são investidos mais atividades para compreender melhor o trabalho dos usuários e garantir que o protótipo supra as expectativas dos usuários e após a elaboração é realizada a validação do protótipo com os usuários.

O analista de sistemas elabora a ERS, avalia a viabilidade técnica junto ao desenvolvimento e elabora o documento de contagem de pontos por função - PPF. Se a contagem de pontos de função exceder ao que estava previsto no EPD, então o EPD deve ser ajustado pelo analista de negócios. Em seguida o mesmo envia para a aprovação do cliente, se o cliente reprovar o chamado é cancelado, se o cliente aprovar, então devolve para o analista de sistemas finalizar as atividades de requisitos.

Finalizando a ERS e o PPF, a ERS é enviada para a validação do analista de negócios e o PPF para validação do analista de métricas. Caso a ERS ou PPF precisem de algum ajuste, então voltam para o analista de sistemas fazer as devidas alterações. Após os documentos serem aprovados, então o analista de negócios envia a ERS e o PPF (dependendo do contrato) para a aprovação do cliente. Nesse caso a ERS pode ser aprovada, cancelada ou reprovada. Se for reprovada, volta para o analista de negócios, que avalia se são correções ou é uma alteração de escopo, em seguida, envia para o analista de sistemas para as correções da ERS. Se for cancelada, então a ERS é cancelada junto com a solicitação. Se a ERS for

aprovada e no contrato do cliente requer também o documento PPF, então a ERS e a PPF são aprovadas.

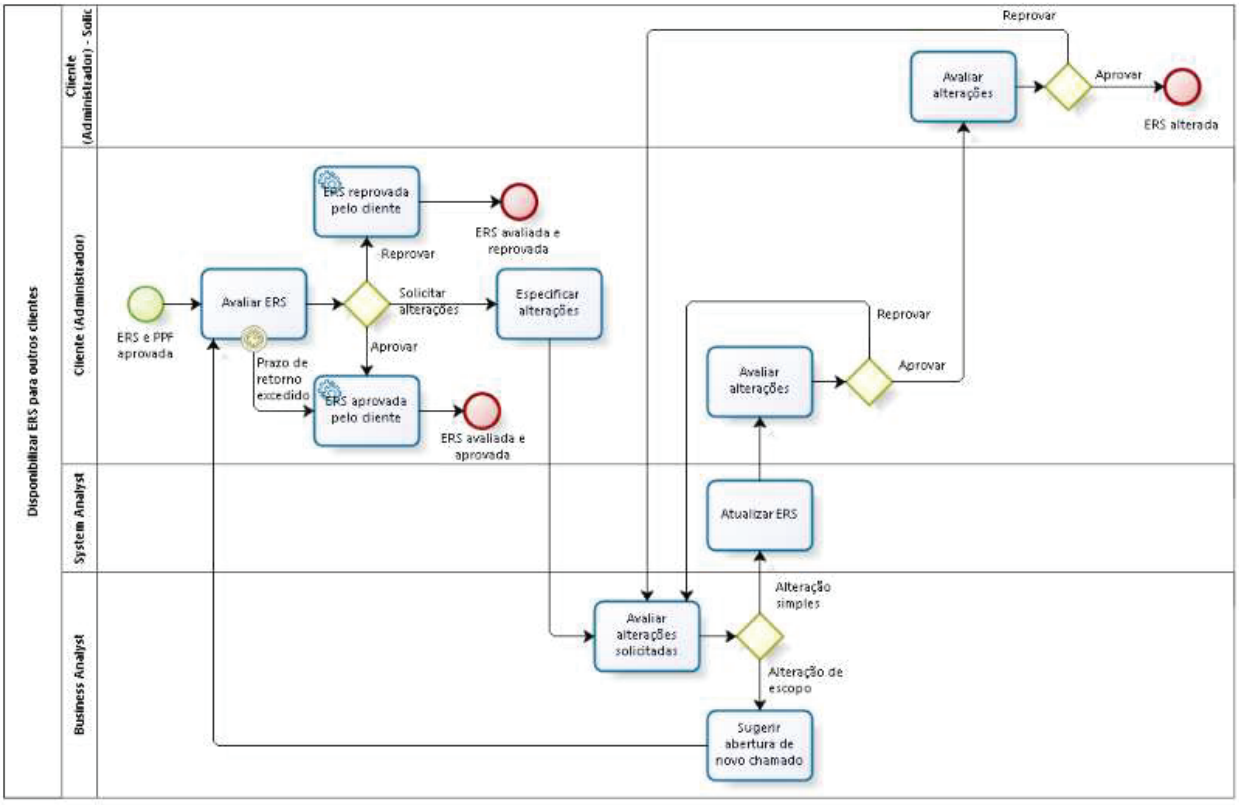
Figura 9. Especificação de requisitos



Fonte: SOFTPLAN, 2015b

Após a ERS e o PPF serem aprovados pelo cliente que abriu a solicitação, a ERS é enviada aos demais clientes para aprovação, representado na figura 10 a seguir. Neste caso, os clientes podem aprovar, reprovar, solicitar ajustes ou perder o prazo para se manifestar. Se o cliente aprovar, então a empresa faz a evolução do sistema para este cliente. Se o cliente reprovar, então o cliente não terá essa evolução no sistema. Se o cliente não retornou e o prazo foi excedido, então se considera aprovado. Se o cliente solicita alterações, então manda essas alterações ao analista de negócios que avalia as alterações que o cliente solicitou e verifica se essa alteração necessita de um novo chamado/solicitação, se sim, envia ao cliente sugerindo abertura do chamado. Caso a alteração for pequena e não necessite de um chamado, então o analista de negócios envia para o analista de sistemas fazer as alterações e depois avalia as alterações feitas pelo analista de sistemas. Após aprovação do cliente, a ERS é alterada e o cliente é informado que haverá a evolução do sistema.

Figura 10. Envio para outros clientes



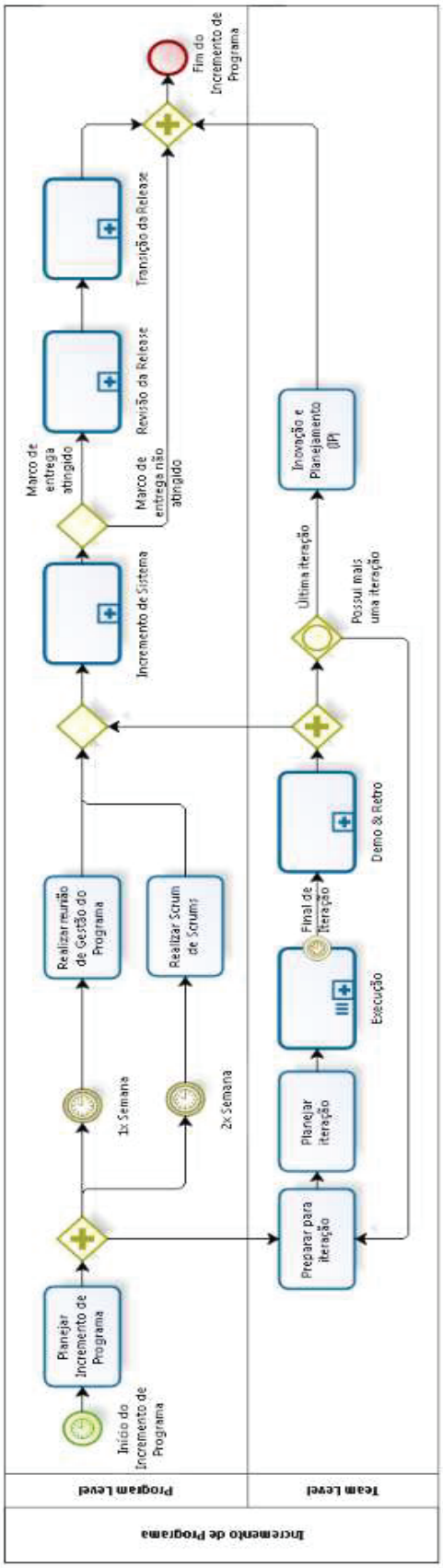
Fonte: SOFTPLAN, 2015b

Feito isso o chamado estará priorizado, estimado e aprovado e em seguida estará no backlog do desenvolvimento. Se a ERS foi aprovada e não foi desenvolvida após transcorrer o prazo de dois ciclos de desenvolvimento, a mesma deve ser avaliada e, se necessário ajustada, enviada novamente para aprovação do cliente. A seguir, é apresentado o processo de desenvolvimento e testes.

4.3.2. Processo de desenvolvimento e testes

O processo descrito a seguir é o processo de desenvolvimento da empresa junto ao processo de testes que está nele incluso, sendo assim, o processo de desenvolvimento está apresentado de forma mais superficial, pois o foco é no processo de testes. Dentro da equipe de desenvolvimento é adotado o modelo ágil Scrum para organizar as atividades, mais detalhes da metodologia ágil pode ser vista na seção 4.2.2.1 deste trabalho. A figura 11, a seguir, representa esses processos.

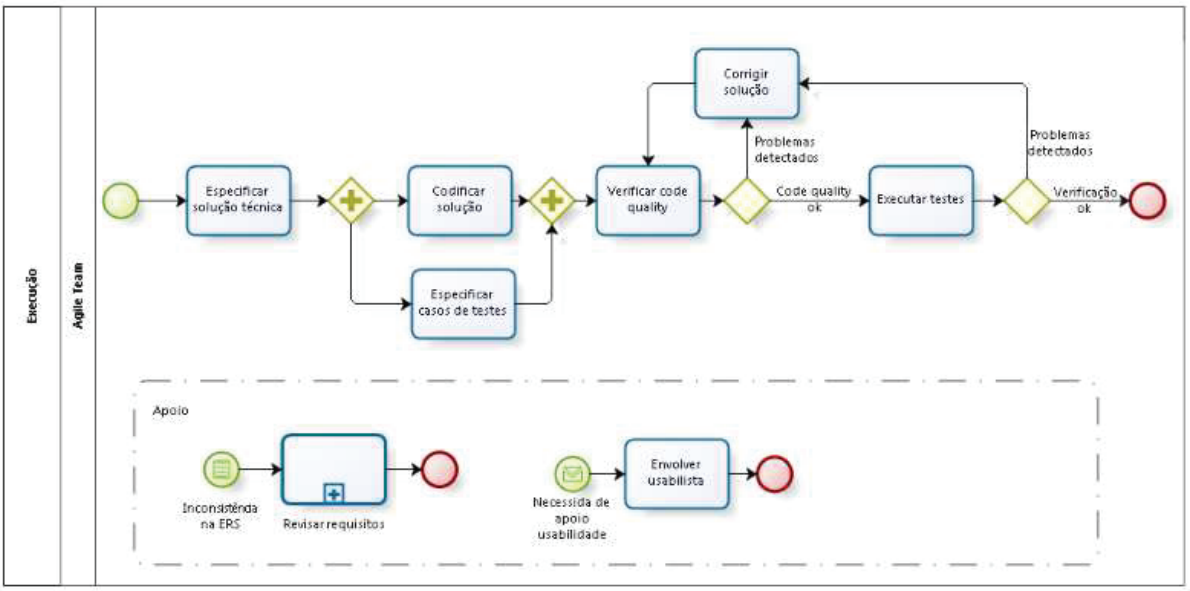
Figura 11. Processo de desenvolvimento



Fonte: SOFTPLAN, 2015b

As ERSs aprovadas e priorizadas pelos clientes são encaminhadas para a equipe de desenvolvimento. Então a equipe se reúne com o PO (Product Owner), para planejar o desenvolvimento, organizar a equipe e definir os atividades e objetivos. O PO é o responsável por sanar dúvidas referente as especificações de requisitos, ou seja, é o profissional que substitui os analistas de sistemas nas sprints, neste processo descrito, os analistas de negócios fazem o papel dos POs. Em paralelo ao desenvolvimento da solução a equipe realiza uma reunião diária para que o time fique sincronizado e uma reunião duas vezes na semana somente dos scrums para alinhar atividades, avaliar impedimentos e informações relevantes da sprint de cada time. Após o alinhamento das tarefas e demandas a serem desenvolvidas, entra o fluxo de execução das atividades. Baseado no documento de requisitos o desenvolvedor faz a especificação técnica, em seguida inicia a codificação, nesse momento o desenvolvedor pode contar com o apoio do usabilista. Em paralelo com o desenvolvimento o analista de teste inicia a execução da elaboração do caso de testes baseado na especificação de requisitos e também a preparação do ambiente de testes que será utilizado para fazer as execuções dos casos de teste. Atualmente não é utilizado um documento de testes, os testes são especificados na Rational Quality Manager (RQM), para registro de erros encontrados é utilizado a ferramenta Rational Team Concert (RTC), ambas ferramentas desenvolvidas pela IBM. Após codificado, o código é verificado pelo scrum master juntamente com o responsável pelo desenvolvimento da solução. Se houver problemas de codificação, o desenvolvedor faz as devidas correções. Quando a codificação estiver correta, inicia-se a execução dos testes. Os analistas de testes/testadores finalizam a preparação do ambiente, executam os casos testes criados e registram os defeitos encontrados. Observa-se na figura 12 a seguir, que a equipe possui o apoio do analista de sistemas, caso haja incoerência na especificação de requisitos, dessa forma, o analista pode fazer os ajustes, se necessário.

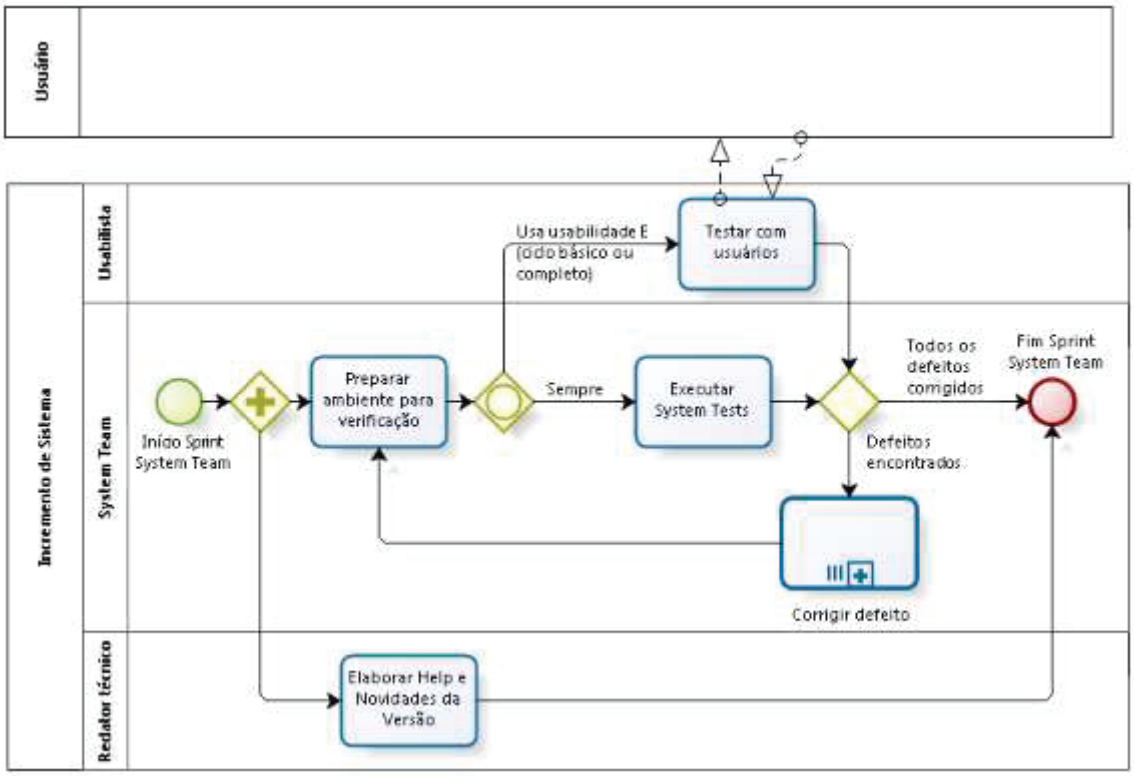
Figura 12. Execução do desenvolvimento da solução



Fonte: SOFTPLAN, 2015b

Após a codificação ser aprovada nos testes, o desenvolvimento faz uma demonstração do que foi desenvolvido para o resto da equipe. Logo depois da demonstração da solução para a equipe, entra a atividade de incremento, é preparado um ambiente para verificação, no qual os testadores realizam as atividades de teste regressivo e o usabilista aplica um teste diretamente com o usuário, envolvendo o analista de sistemas e o cliente. Em paralelo a essa atividade o redator técnico faz a elaboração de um Help (Manual de auxílio) que fica dentro do sistema quando a funcionalidade é acessada pela primeira vez e elabora também um documento de novidades da versão para o cliente e demais interessados. Essas atividades podem ser vistas no processo descrito na figura 13.

Figura 13. Incremento do programa



Fonte: SOFTPLAN, 2015b

Após essas atividades, a equipe de desenvolvimento realiza a demonstração final ao PO (Product Owner), analista de sistemas e demais interessados. Após aprovação, atendendo aos critérios de aceitação, o sistema é liberado na versão para homologação.

A seguir, é apresentada a análise do questionário aplicado para alguns funcionários da empresa, buscando reunir informações e opiniões de pessoas experientes na área e que fazem parte do processo descrito.

4.4 QUESTIONÁRIOS APLICADOS E RESULTADOS OBTIDOS

O questionário contém 10 perguntas e foi baseado no contexto necessário para sugerir, melhorias no processo de requisitos e na documentação desse processo dentro da empresa. As perguntas realizadas nos questionários foram as seguintes:

1. Qual seu cargo na Softplan?
2. Há quanto tempo trabalha nesta empresa?
3. Você acha que o processo de software da empresa está bem definido e ele é completo?
4. Você acha que o processo de teste da empresa precisa de alguma melhoria? Justifique sua resposta.
5. Você acha que o processo de análise de requisitos da empresa precisa de alguma melhoria? Justifique sua resposta.
6. Você acredita que tem algum modo de melhorarmos o levantamento de requisitos e o seu registro, dessa forma facilitando a elaboração de casos de teste? Poderia nos dar exemplos e sua visão sobre o assunto?
7. Em sua opinião quais são os principais fatores que influenciam na testabilidade de um requisito?
8. Você acha que os requisitos devem ter um alto nível de granularidade?
9. Você acha que a documentação elaborada no decorrer do processo de desenvolvimento de software é suficiente?
10. Você tem mais alguma observação a fazer referente ao tema citado?

O questionário foi enviado via e-mail para algumas pessoas selecionadas para responder as perguntas, de forma a ajudar com a construção da melhoria que será proposta para a empresa. O questionário foi aplicado para um total de 24 pessoas, sendo que 22 pessoas responderam. O questionário foi criado no formulário do Google.

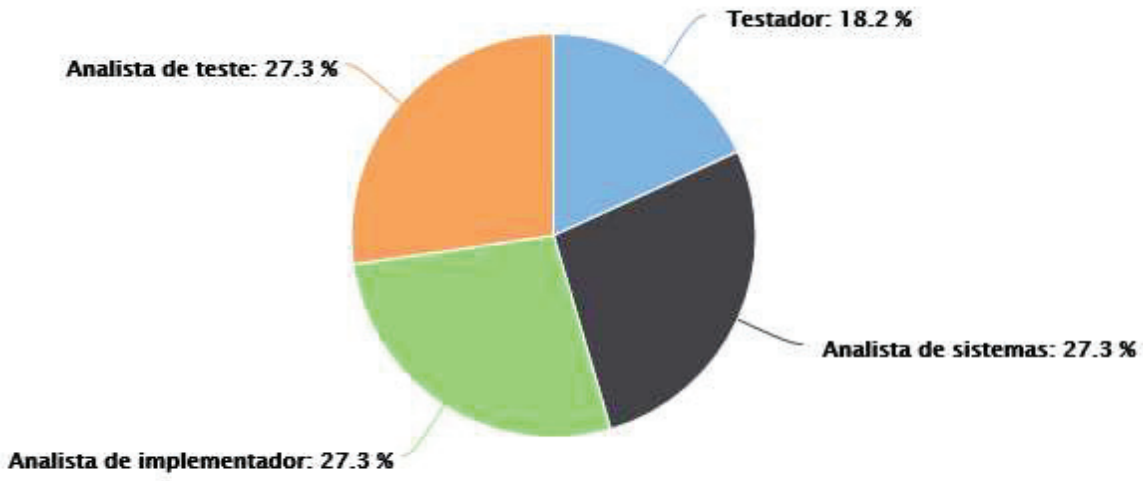
4.4.1 Análise dos resultados obtidos

Nesta seção está apresentada uma análise com base nas respostas obtidas pelas pessoas que responderam ao questionário. No Apêndice C deste trabalho pode-se visualizar as respostas de forma mais detalhada.

Pergunta 1: Qual seu cargo na Softplan?

No gráfico a seguir pode-se observar que o cargo das pessoas envolvidas no questionário foi bem diverso, dessa forma obtiveram-se opiniões de pessoas de diferentes cargos e experiências.

Gráfico 1. Pergunta 1

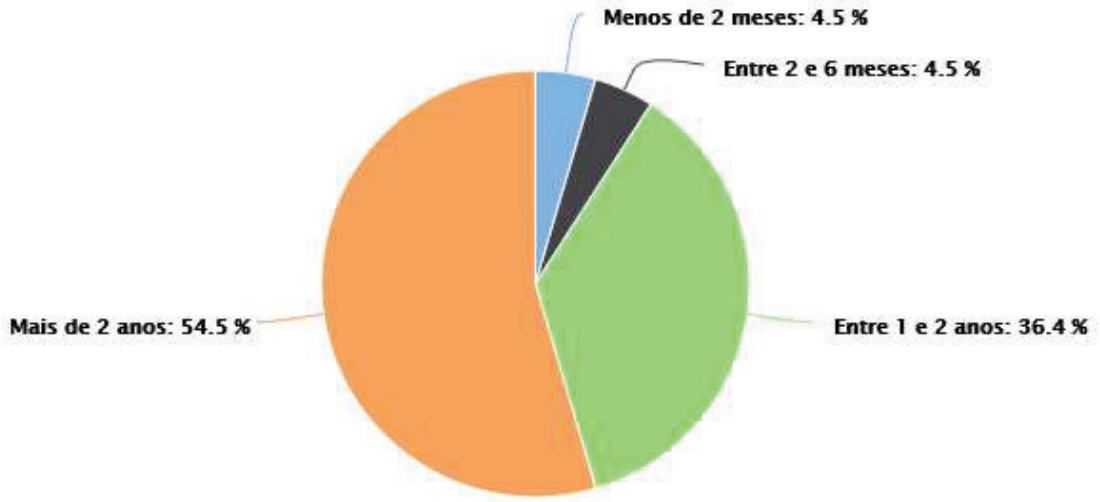


Fonte: das autoras, 2015.

Pergunta 2: Há quanto tempo trabalha nesta empresa?

Com base nos resultados obtidos, constatou que a maioria das pessoas que responderam ao questionário possuem experiência e conhecimento no processo que funciona atualmente na empresa. Dessa forma, pode-se dar boa credibilidade às respostas das próximas questões.

Gráfico 2. Pergunta 2



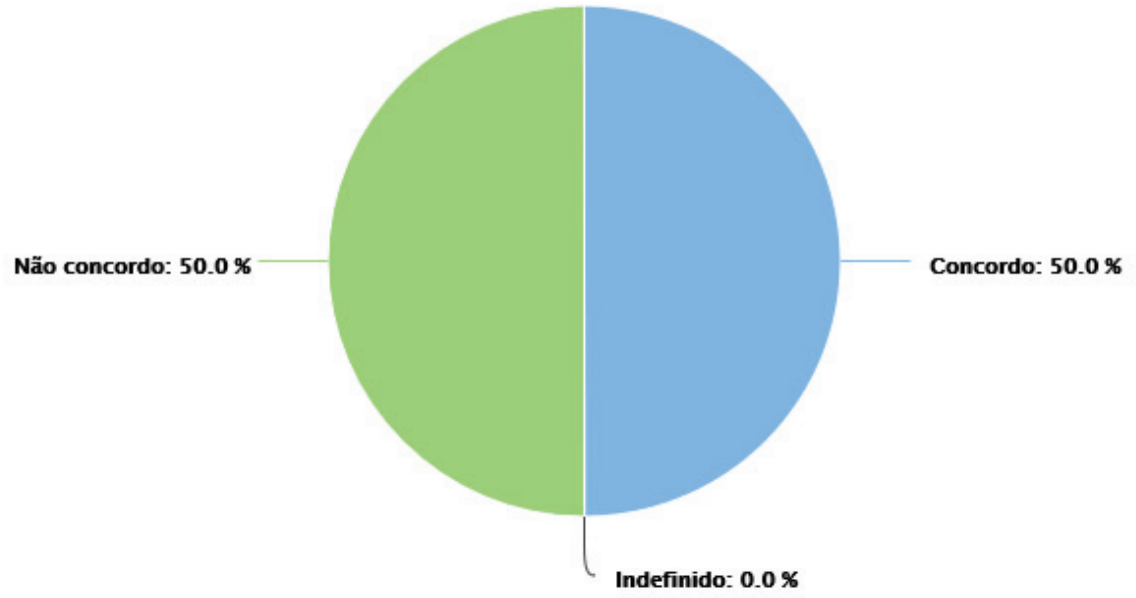
Fonte: Fonte: das autoras, 2015.

Pergunta 3: Você acha que o processo de software da empresa está bem definido e ele é completo?

A figura a seguir tem o propósito de retratar de forma quantitativa as respostas obtidas. Cabe ainda destacar que as respostas “Concordo totalmente” e “Concordo parcialmente” foram agrupadas, assim como as respostas “Não concordo totalmente” e “Não concordo parcialmente”.

Pode-se verificar que 50% concordam com uma boa definição do processo enquanto a outra metade dos questionários acha que o processo não está bem definido. No Apêndice D desta monografia se encontra o gráfico mais detalhado com as respostas desta pergunta.

Gráfico 3. Pergunta 3



Fonte: das autoras, 2015.

Algumas pessoas responderam que o processo a ser seguido é muitas vezes oneroso e que por esse motivo muitos colaboradores deixam de segui-lo. Alguma das respostas destacava que uma forma de que as pessoas realizem todas as atividades definidas no processo é que exista alguma etapa para forçar o cumprimento de cada tarefa, isto é, não pode passar na próxima atividade do processo até não ter “assinalado” a finalização da atividade anterior, assim facilitando a execução de todas as etapas do processo.

Outras pessoas acreditam que o processo não atende todas as necessidades, e que muitas coisas precisam ser definidas e cobradas. O amadurecimento dos processos e a atualização do mesmo se vê necessário por boa parte dos colaboradores. Alguns acreditam que registrar as lições aprendidas em alguma parte do processo ajudaria na agilidade e eficiência do mesmo.

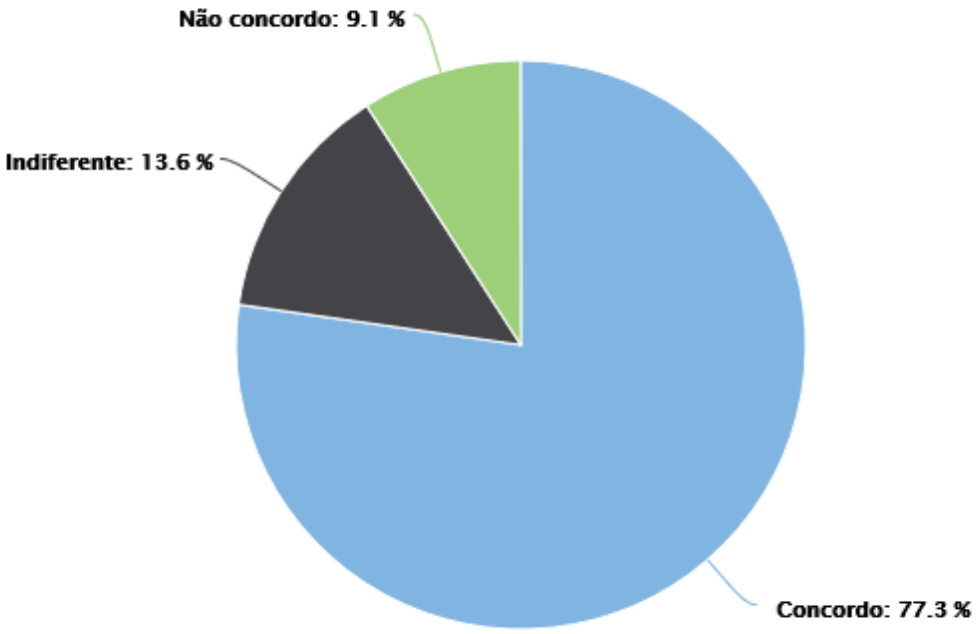
Um grande problema que muitos citaram é que a solicitação para burlar o processo vem muitas vezes de cargo de chefia o que atrapalha no desenvolvimento de um bom processo a ser seguido.

Pergunta 4: Você acha que o processo de teste da empresa precisa de alguma melhoria? Justifique sua resposta.

A figura a seguir tem o propósito de retratar de forma quantitativa as respostas obtidas. Cabe ainda destacar que as respostas “Concordo totalmente” e “Concordo parcialmente” foram agrupadas, assim como as respostas “Não concordo totalmente” e “Não concordo parcialmente”.

Com o gráfico a seguir podemos ver que mais de 70% dos entrevistados acreditam que o processo de testes da empresa precisa de melhorias. No Apêndice D desta monografia o gráfico mais detalhado com as respostas desta pergunta.

Gráfico 4. Pergunta 4



Fonte: das autoras, 2015.

Com base nos resultados obtidos verificamos que muitas pessoas acreditam na falta de documentação e que esse é o ponto chave para as melhorias. Retratam ainda, que se as informações documentadas forem bem definidas ainda na fase de análise de requisitos, será possível proporcionar a melhoria no processo de teste.

Foi mencionada também, a falta de sincronização de informação com todos os envolvidos no processo, porém, se houver a documentação necessária, todos os

envolvidos terão acesso à informação. O Processo definido deve ser cumprido e isso na opinião de muitos não ocorre.

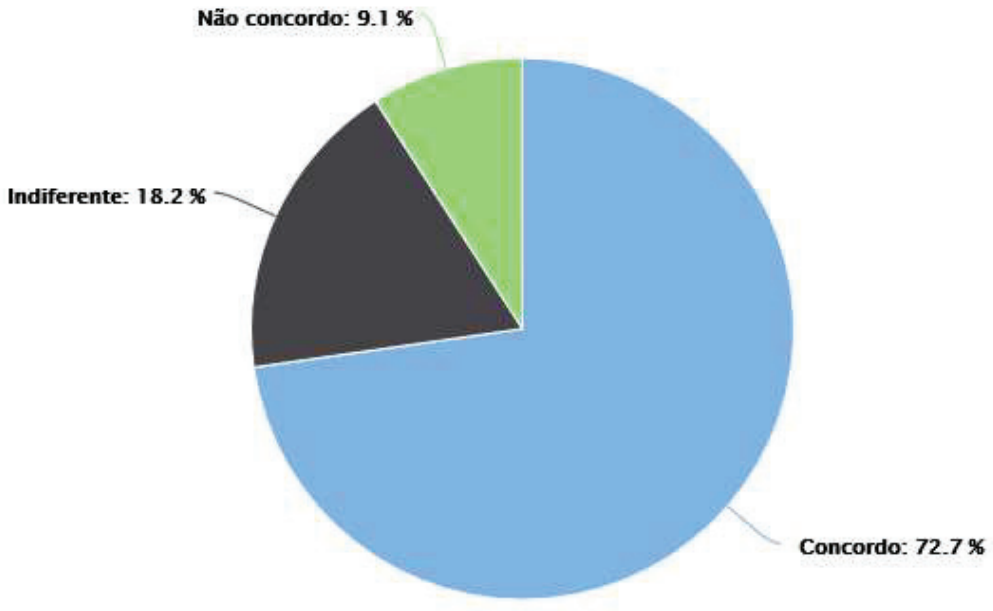
Acredita-se que deveria haver mais roteiros de testes, para a maioria dos módulos do sistema, ou pelo menos para as partes que sofrem mais alterações no sistema, mas sabemos que isso foge da nossa realidade e que na maioria das vezes não é feito prejudicando todo o processo e etapas envolvidas.

Pergunta 5: Você acha que o processo de análise de requisitos da empresa precisa de alguma melhoria? Justifique sua resposta.

A figura a seguir tem o propósito de retratar de forma quantitativa as respostas obtidas. Cabe ainda destacar que as respostas “Concordo totalmente” e “Concordo parcialmente” foram agrupadas, assim como as respostas “Não concordo totalmente” e “Não concordo parcialmente”.

É possível verificar que 72% dos entrevistados concordam que o processo de análise de requisitos da empresa precisa de melhorias. No Apêndice D desta monografia existe o gráfico mais detalhado com todas as respostas desta pergunta.

Gráfico 5. Pergunta 5



Fonte: das autoras, 2015.

Conforme visto no Apêndice C deste trabalho, um dos entrevistados aponta que os Analistas de sistemas não possuem um entendimento da necessidade do cliente e que essa informação nem é buscada pelo mesmo. Observa-se que é apenas transcrita a solicitação do cliente, mas não são verificados os motivos que o levaram a solicitar essa demanda, dessa forma fica escasso o entendimento e o que e como que o cliente espera receber essa alteração.

É apontado que no processo deveria haver etapas para prever a especificação de requisitos do sistema legado, caso a alteração envolva a parte legado do sistema.

Um dos nossos entrevistados aponta que um problema na análise de requisitos é a forma que cada pessoa pensa, pois cada analista tem seu modo de escrever e pensar, tornando um tanto difícil a interpretação do que o analista quer dizer ou representar.

Outro questionado aponta que seria interessante se uma análise de impacto fosse prevista no processo, ela poderia ser aplicada em todo o sistema para prevenir alterações que impactariam em uma parte muito maior que o previsto em primeiro lugar.

Pergunta 6: Você acredita que tem algum modo de melhorarmos o levantamento de requisitos e o seu registro, dessa forma facilitando a elaboração de casos de teste? Poderia nos dar exemplos e sua visão sobre o assunto?

Alguns dos entrevistados acreditam que a especificação de requisitos ficaria mais clara se fossem citados exemplos de fluxo do cliente, assim, facilitando a elaboração dos casos de teste diretamente ligados com a realidade do cliente. Gerar padrão para a elaboração de requisitos também seria uma boa melhoria para o levantamento de requisitos.

Alguns apontaram o uso de ferramentas mais consolidadas no mercado e incluir a obrigatoriedade do levantamento de requisitos no processo. A documentação estar bem elaborada e interligada com todas as documentações que fazem parte de uma demanda ajudaria também na agilidade do processo e em sua melhoria.

Um dos respondentes diz que uma forma de melhorar o levantamento de requisitos é que mais pessoas participem desse processo de validação, como

testadores e implementadores. Uma das coisas que ajudaria também no processo e o tornaria mais confiável é a rastreabilidade do requisito, pois é importante para identificar quem, quando e porque o requisito foi alterado, além dos impactos causados na alteração desse requisito.

Pergunta 7: Na sua opinião quais são os principais fatores que influenciam na testabilidade de um requisito?

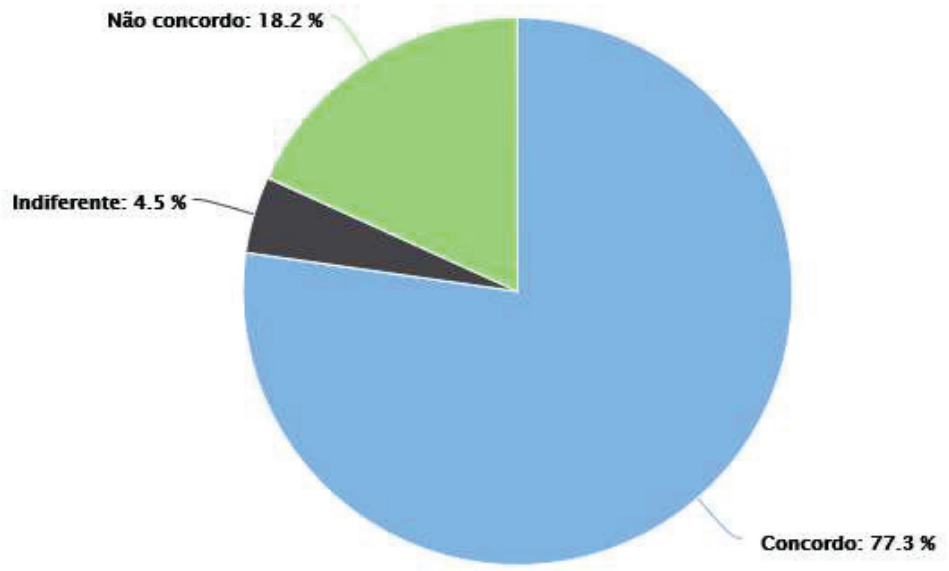
Essa questão foi respondida por apenas 4 pessoas das 22 entrevistadas gerando poucos dados. No entanto, as pessoas que responderam relatam que a clareza no levantamento e na descrição de requisitos é essencial para evitar que exista omissão de informações que são importantes ao usuário. Uma boa especificação de requisitos e a experiência dos analistas envolvidos são fatores que influenciam positivamente na testabilidade do requisito.

Pergunta 8: Você acha que os requisitos devem ter um alto nível de granularidade?

A figura a seguir tem o propósito de retratar de forma quantitativa as respostas obtidas. Cabe ainda destacar que as respostas “Concordo totalmente” e “Concordo parcialmente” foram agrupadas, assim como as respostas “Não concordo totalmente” e “Não concordo parcialmente”.

Percebe-se no gráfico que quase 80% das pessoas entrevistadas acreditam que o requisito deve ter um alto nível de granularidade. No Apêndice D desta monografia se encontra o gráfico mais detalhado com as respostas desta pergunta.

Gráfico 6. Pergunta 8



Fonte: das autoras, 2015.

Conforme as respostas obtidas no Apêndice C, um dos respondentes afirma que se o requisito for muito granular pode gerar muitos artefatos separados que podem confundir durante a modelagem do sistema, é preciso ter um bom senso de saber em que momento um artefato deve ser quebrado em dois ou se basta melhorar o texto e continuar com apenas um.

Outros acreditam que com alto nível de granularidade cada caso de uso a ser desenvolvido tem seu escopo reduzido e isso facilita a visão geral de cada item, dessa forma é possível ter um maior entendimento e clareza por todos os envolvidos, deixando totalmente claro o que e o porquê deve ser feito tal item.

Alguns de nossos entrevistados acreditam que a teoria do dividir para conquistar é muito interessante aplicado a granularidade de um requisito, pois quanto mais quebrarmos os requisitos em níveis de granularidade ficará mais fácil para confeccionar casos de teste e desenvolver.

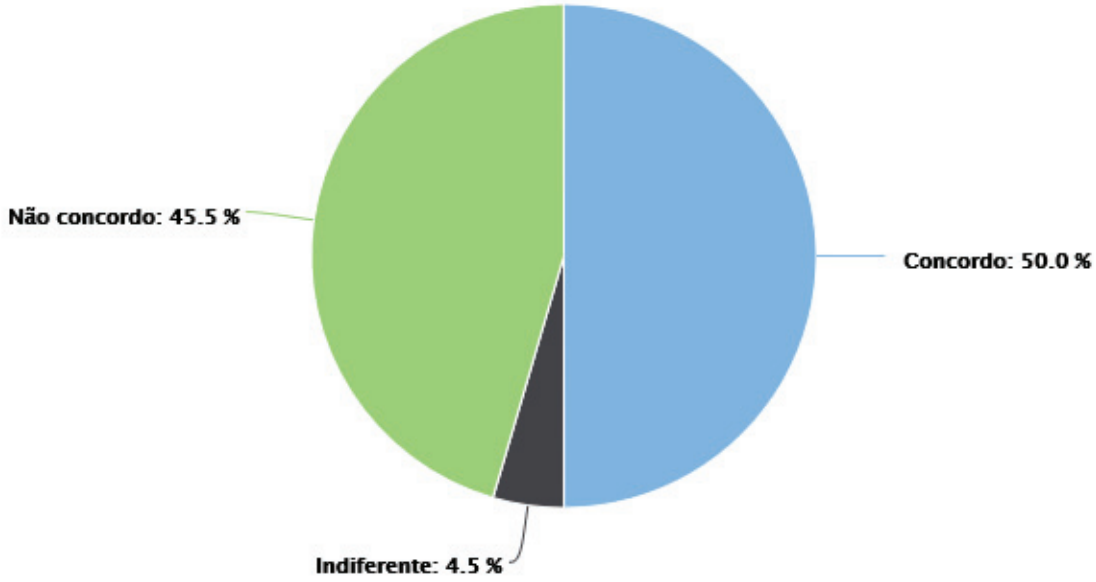
Outro questionado afirmou que um requisito lido isoladamente deve ter sentido completo, sendo capaz de indicar o comportamento esperado. Caso isso não ocorra, pode ser um bom indício que a granularidade está muito alta.

Pergunta 9: Você acha que a documentação elaborada no decorrer do processo de desenvolvimento de software é suficiente?

A figura a seguir tem o propósito de retratar de forma quantitativa as respostas obtidas. Cabe ainda destacar que as respostas “Concordo totalmente” e “Concordo parcialmente” foram agrupadas, assim como as respostas “Não concordo totalmente” e “Não concordo parcialmente”.

No gráfico a seguir pode nota-se que 50% das pessoas concordam que a documentação elaborada é suficiente e 45% acredita que a documentação é insuficiente. No Apêndice D desta monografia você encontra as respostas mais detalhadas desta pergunta.

Gráfico 7. Pergunta 9



Fonte: das autoras, 2015.

De acordo com nossos entrevistados a documentação gerada pode ser suficiente desde que seja bem elaborada, que contenham todas as informações de forma clara e objetiva. É sugerido que exista Documentação inicial de negócio, documentação de requisitos e documentos de acompanhamentos de teste desde as

fases iniciais de levantamento e ainda se vê a necessidade de um documento ao levantar requisitos com o cliente.

Uma documentação que contemple o que o sistema tem de legado é muito importante, pois a documentação do legado atual é muito desatualizada, incompleta ou ainda é inexistente.

As documentações devem ser fáceis de pesquisar e navegar, para que seja possível recuperar o que se deseja de forma rápida.

A mesma documentação enviada para o cliente é a documentação que é repassada para a equipe de desenvolvimento. A documentação é totalmente incompleta e não suficiente para o desenvolvimento de novas funcionalidades. Muitos detalhes, que o cliente não precisa saber, mas que são fundamentais para o desenvolvimento não são colocados neste documento. Deveria haver dois tipos de documentação.

Pergunta 10: Você tem mais alguma observação a fazer referente ao tema citado?

Algumas pessoas ainda completaram o questionário respondendo que acreditam que uma alternativa para melhorar o processo é torná-lo mais ágil para realizar as atividades, dessa forma todos os envolvidos deveriam participar do processo e suas definições desde o começo, com isso gerando mais agilidade e comunicação entre as partes envolvidas.

Uma outra alternativa de melhoria do processo é a necessidade de os profissionais da empresa saberem de como a especificação foi desenvolvida e como fazer sua configuração no cliente, pois na hora de instalar e homologar no cliente “sempre faltam parâmetros ou fazer certo comando e a falta de informação prejudica nesse momento”.

4.5. SUGESTÃO DE MELHORIAS

Nesta seção estão descritas as melhorias sugeridas no processo, bem como novos artefatos e sugestão de melhorias nos artefatos já utilizados. As sugestões descritas a seguir são baseadas na abordagem teórica deste trabalho e também nos relatos recebidos através dos questionários aplicados aos funcionários da empresa.

4.5.1. Comunicação com o cliente

No dia a dia verifica-se o quanto é importante à comunicação com o cliente, pois quanto mais se questiona o cliente e procura-se entender sua necessidade, melhor será definido e entregue o produto. Dessa forma, uma das melhorias sugeridas no processo é aplicar uma entrevista ou questionário no cliente para coletar dados mais concretos na hora de analisar os requisitos e a necessidade dos mesmos.

É a partir das entrevistas que os analistas levantam os requisitos de software. As perguntas podem ser fechadas ou abertas, sendo que no questionário, perguntas fechadas, o stakeholder responde a uma determinada sequência de perguntas e na entrevista o stakeholder responde a uma série de dúvidas e questionamento da equipe de engenharia de requisitos, desenvolvendo, assim, uma melhor compreensão das necessidades. (SOMMERVILLE, 2011)

Com essa melhoria visa-se evitar alguns problemas que os entrevistados apontaram ao responder ao questionário. Um dos respondentes relatou que algumas vezes o analista de sistemas não tem um entendimento da necessidade do cliente e que essa informação não é buscada pelo mesmo. Menciona também, que é apenas transcrita a solicitação do cliente, mas não são verificados os motivos que o levaram a solicitar essa demanda, dessa forma entende-se que a inclusão da entrevista no processo ajudará a entender o que o cliente deseja e como ele imagina a solução.

A comunicação do analista de sistemas com o cliente vai melhorar o entendimento do analista, de forma que este possa realizar especificações de requisitos mais claras e com a real necessidade do cliente. Essa melhoria impacta diretamente na elaboração dos casos de testes, porque assim como o cliente, o profissional de teste tem a visão e o papel de um cliente com a intenção de reproduzir o fluxo do usuário e entender também sua necessidade. Sendo assim, percebe-se que ao aplicar essa melhoria podemos garantir um maior entendimento da necessidade do cliente por parte dos analistas e isso vai impactar diretamente nos analistas de testes que vão montar seus casos de teste voltados para essas necessidades.

4.5.2. Duas especificações de requisitos

Na questão 9 do questionário aplicado aos funcionários da empresa houve várias opiniões sobre as documentações elaboradas durante o processo, uma das pessoas relatou que a mesma documentação enviada para o cliente é a documentação que é repassada para a equipe de desenvolvimento. A documentação geralmente é incompleta e não suficiente para a equipe de desenvolvimento. Muitos detalhes, que o cliente não precisa saber, mas que são fundamentais para o desenvolvimento, não são colocados neste documento. Assim, inferimos que haja dois tipos de documentação.

De acordo com Ian Sommerville (2011), o processo de engenharia de requisitos tem como objetivo a produção de um documento de requisitos que especifica um sistema em detalhes, satisfazendo o que o cliente deseja. Esses requisitos geralmente são apresentados de duas formas, com um documento em alto nível para o cliente e um documento de forma mais técnica e mais detalhada para os desenvolvedores.

De acordo com a sugestão dos entrevistados e com base na teoria, a segunda melhoria a ser sugerida é a elaboração de dois documentos, sendo que no primeiro o sistema será especificado, em alto nível, de forma a fazer que o cliente entenda todos os requisitos do sistema. Essa especificação será realizada de forma mais

superficial, com as principais regras e protótipos de tela para facilitar o entendimento do cliente sobre o que será entregue. O outro documento será mais técnico, escrito diretamente para os desenvolvedores e testadores. Será um documento mais detalhado e terá informações que vão ajudar diretamente no desenvolvimento da solução.

O segundo documento criado ajudará na especificação de teste, pois ele será mais detalhado, mais técnico e com exemplos reais para facilitar a elaboração da especificação, além de ajudar também no desenvolvimento, com o item de **análise de impacto** e o item **alterações técnicas**. Este último apontará ao desenvolvedor alterações de parâmetros, necessidade de criação de scripts e outras questões mais técnicas.

Os modelos desses dois documentos se encontram nos Apêndices F e G desta monografia.

4.5.3. Relação de documentos envolvidos na especificação de requisitos

Sabe-se que a falta de documentação nos processos de software pode ser algo comum nas empresas, mas para as empresas que possuem uma documentação mais completa, o problema pode ser na hora de encontrar esses documentos.

Um dos entrevistados relatou no questionário que as documentações devem ser fáceis de pesquisar e navegar, para que seja possível recuperar o que se deseja de forma rápida. Ian Sommerville (2011), destaca que um dos principais aspectos que fazem parte da gerência de requisitos é gerenciar as dependências entre o documento de requisitos e outros documentos produzidos durante o desenvolvimento de software.

Desta forma, a outra sugestão é colocar no modelo do documento de requisitos, que vai para o desenvolvimento, um tópico para documentos relacionados durante o processo daquela determinada solução. Essa medida tem o intuito de simplificar a busca por documentos relacionados, facilitando as atividades

da especificação do teste, pois assim qualquer profissional poderá, com mais rapidez avaliar os outros documentos que fazem parte da solução especificada.

No Apêndice G desta monografia, o documento de Especificação de Requisitos de Software Técnico apresenta o tópico criado para os documentos relacionados.

4.5.4. Descrição da revisão na especificação de requisitos

Um dos principais erros na documentação de requisitos é a falta de atualização desse documento. Às vezes, mesmo atualizado, não se sabe o porquê da alteração, se foi solicitação de algum ajuste pelo analista de negócios, alteração de escopo pelo cliente, entre outros motivos pelos quais tantas mudanças ocorrem, no decorrer de um desenvolvimento de software.

A próxima melhoria sugerida é a inclusão de um resumo no quadro de versões de o porquê essa ERS está sofrendo alterações. Ian Sommerville (2011), destaca uma estrutura de um documento de requisitos e nele, um prefácio descrevendo o histórico de versão do documento com resumo destas mudanças.

Mais uma vez pode-se notar a vantagem de uma melhoria no documento de requisitos, possibilitando a pessoa que for usar o documento ter conhecimento do motivo pelo qual aquela modificação foi realizada, dentre estas pessoas, a equipe de testes, que tanto usa o documento de requisitos para elaborar os inúmeros cenários nas especificações de requisitos.

4.5.5. Validação e verificação da especificação de requisitos

No processo atual da empresa, é possível perceber que a validação e verificação do documento de requisitos são feitas pelo analista de negócios com envolvimento do analista de sistemas, quem atuou no levantamento de requisitos, e

elaboração do documento, porém, nenhum documento de validação é utilizado nessa etapa do processo. Pressman diz que é na validação de requisitos que é verificado se os requisitos estão bem claros, sem ambiguidade e possíveis de serem testados e que um documento de requisitos pode ser validado após algumas perguntas que possam garantir a qualidade desse documento. Desta forma, sugere-se um documento de avaliação dos requisitos, no qual o analista de negócios responde a um check list de perguntas referentes aos requisitos apresentados no documento. O documento criado sugerido para utilizar na validação dos requisitos pode ser visualizado através do Apêndice E deste trabalho.

Como se pode observar na figura 4, essa validação no processo atual é feita somente pelo analista de negócios. Nota-se que muitas vezes as falhas no documento de requisitos são encontradas no momento de testar ou de especificar uma determinada solução. Desta forma, sugere-se também o envolvimento do analista de testes na validação dos documentos de requisitos.

Essa melhoria também foi sugerida por um dos respondentes que aponta que uma forma de melhorar o levantamento de requisitos é que mais pessoas participem desse processo de validação, como testadores e implementadores.

4.5.6. Entrega da solução

O processo atual da empresa possui uma atividade na qual o redator técnico elabora o Help do sistema e um documento de novidades da versão a ser entregue. Pode se garantir, que não basta finalizar o desenvolvimento de um software e entregar o produto ao cliente, o cliente precisa saber como utilizar o sistema e a nova solução. Com base no sistema das procuradorias jurídicas, a solução é apresentada sempre aos procuradores que estão à frente da negociação, os demais procuradores só tomarão conhecimento da solução quando ela já estiver funcionando dentro da procuradoria, deste modo, toda documentação para ajudar os demais procuradores é necessária.

A próxima sugestão é que o próprio analista de sistemas elabore um manual para o cliente (documento não sugerido), de alto nível, sobre como fazer

determinada tarefa dentro daquela solução desenvolvida, pois é o analista de sistemas que faz a prototipação e toda a documentação de como o sistema deve se comportar, desta forma, o próprio analista é um profissional melhor adequado para elaborar esse documento.

Sommerville (2011), explica que a engenharia de software não se trata somente da programação, mas também, de toda a documentação durante o processo de desenvolvimento de software, incluindo, inclusive, manual do usuário.

Pode-se notar também, que o help elaborado pelo redator técnico acontece depois da especificação de testes e depois da execução dos testes. Sendo assim, essa melhoria impacta diretamente no teste, facilitando o entendimento e a produção da documentação necessária destes colaboradores.

Esse documento pode não ser usado só para o cliente e a equipe de testes, mas para todos os envolvidos dentro da empresa, até mesmo, para a equipe de homologação, que muitas vezes são os consultores que estão mais em contato com os clientes do que com o próprio time de desenvolvimento.

4.5.7. Rastreabilidade de Requisitos

No processo utilizado na empresa atualmente não possui nenhum artifício para fazer que um requisito seja rastreável. Um dos respondentes dos questionários aponta que uma das coisas que ajudaria também no processo, e o tornaria mais confiável, é a rastreabilidade de requisitos, pois é importante para identificar quem, quando e por que o requisito foi alterado, além dos impactos causados na alteração desse requisito.

Segundo Hélio Engholm Jr. (2010), a rastreabilidade serve para auxiliar na manutenção do sistema, dos testes a serem executados e os impactos que essas mudanças podem causar. A partir do momento que o processo de engenharia de requisitos inicia, deve-se começar também a rastreabilidade, para atualizar os atributos e avaliar os impactos.

Diferente da melhoria “Descrição da revisão na especificação de requisitos” que aponta que o documento de ERS terá uma tabela para apontar quando e o

porquê aquele documento foi alterado, esta melhoria sugere que, de preferência, uma ferramenta faça a gestão de rastreabilidade dos requisitos. No processo atual da empresa é utilizado a ferramenta EA (Enterprise Arquitect) para especificações de requisitos além dos documentos já citados anteriormente. A sugestão a ser aplicada é o uso da Ferramenta Doors da IBM.

Segundo o site da empresa IBM:

> Rational DOORS é um aplicativo de gerenciamento de requisitos para otimizar a comunicação, colaboração e verificação de requisitos em toda sua organização e em sua cadeia de fornecimento. Essa solução escalável pode ajudá-lo a gerenciar o escopo e custo do projeto e atender às metas de negócios. O Rational DOORS permite capturar, rastrear, analisar e gerenciar as mudanças nas informações e demonstrar a conformidade com os regulamentos e padrões. (IBM, 2015)

A ferramenta DOORS faz a integração com um módulo de versionamento de requisitos que torna muito mais fácil a gestão desses requisitos através do versionamento de cada um.

4.6. SUGESTÃO DE UM NOVO PROCESSO

Nesta seção está descrito o novo processo de levantamento de requisitos idealizado com base na fundamentação teórica deste trabalho, no processo já existente na empresa e com as melhorias sugeridas na seção 4.5. Desta forma, propõe-se um processo mais definido, mais completo e menos suscetível a erros.

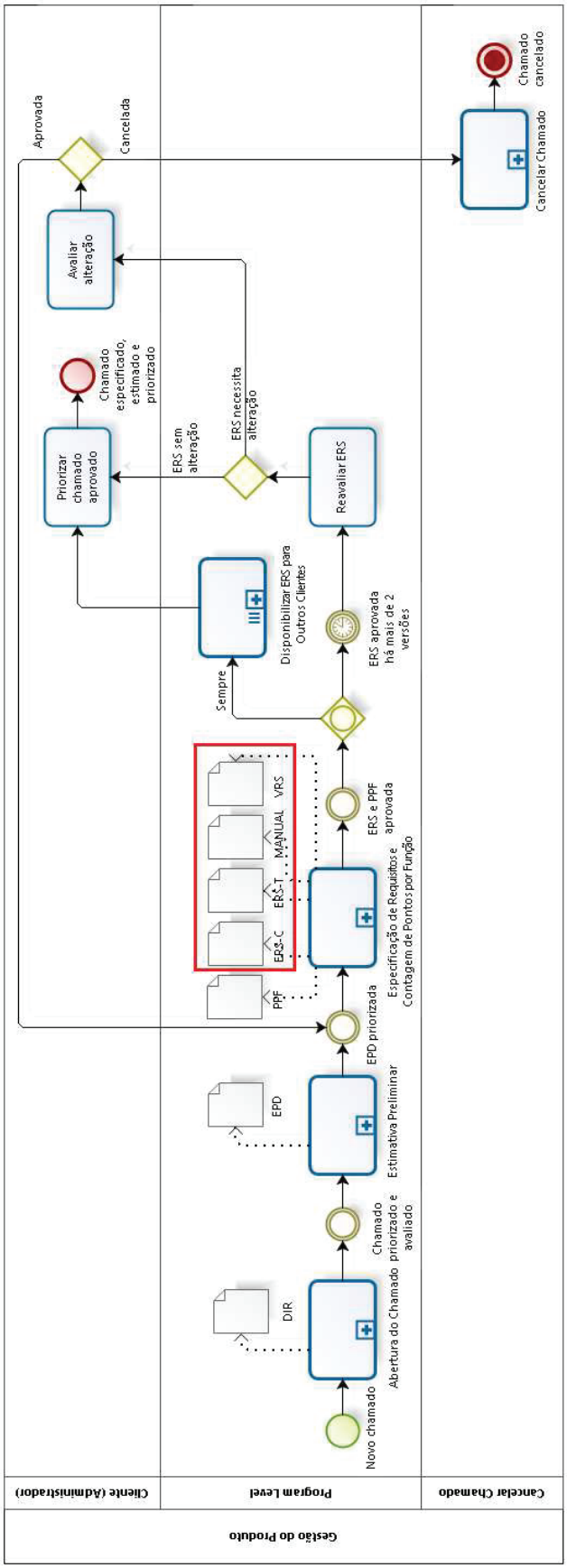
Conforme Sommerville (2011), mesmo que não exista um processo ideal de software, muitas empresas possuem lacunas nos processos de softwares, que podem ser melhorados. Os processos podem ser melhorados sendo padronizados, possibilitando melhor comunicação e reduzindo treinamentos, tornando-se mais econômico o apoio ao processo automatizado.

Os fluxos dos processos apresentados a seguir, são somente dos que sofreram alguma modificação, dessa forma, os fluxos de elaboração da DIR e do EPD não são apresentados.

4.6.1. Processo Sugerido

A figura 14 a seguir apresenta o processo de gestão do produto na qual foram incluídos os documentos sugeridos que são utilizados nesse processo.

Figura 14. Processo de manutenção evolutiva – Gestão de produto Novo



Fonte: das autoras, 2015

O processo continuará a inicializar com a abertura de uma solicitação (chamado) de melhoria no sistema. É preenchido um registro chamado de DIR (Documento Inicial de Requisitos) pelo analista de negócios ou pelo próprio cliente. Após o aceite do cliente, a solicitação é priorizada e realizado EPD (Estimativa Preliminar de Demanda), também pelo analista de negócios, conforme o processo atual da empresa detalhado na seção anterior.

Após o EPD ser aprovado e o prazo da ERS definido, o analista de negócios avalia qual ciclo de usabilidade será aplicado nessa especificação, em seguida, encaminha para o analista de sistemas fazer a elaboração da ERS (Especificação de Requisitos de Software – Cliente), o modelo deste documento encontra-se no apêndice F deste trabalho (melhoria apontada na seção 4.5.2). Ao se iniciar o processo requisitos, o analista de sistema avalia o tamanho da alteração e faz o levantamento de requisitos. No levantamento de requisitos será realizada uma atividade de rastreabilidade dos requisitos, onde o analista de sistemas verifica na ferramenta adotada se o sistema já possui os requisitos dessa alteração, se é necessário alterá-los ou se o sistema ainda não possui esses requisitos, nesse caso, devem ser criados (melhoria apontada na seção 4.5.7). Ainda no levantamento de requisitos, o analista pode realizar uma entrevista, diretamente com o cliente, aplicando um questionário para o melhor entendimento da solicitação, se necessário (melhoria apontada na secão 4.5.1).

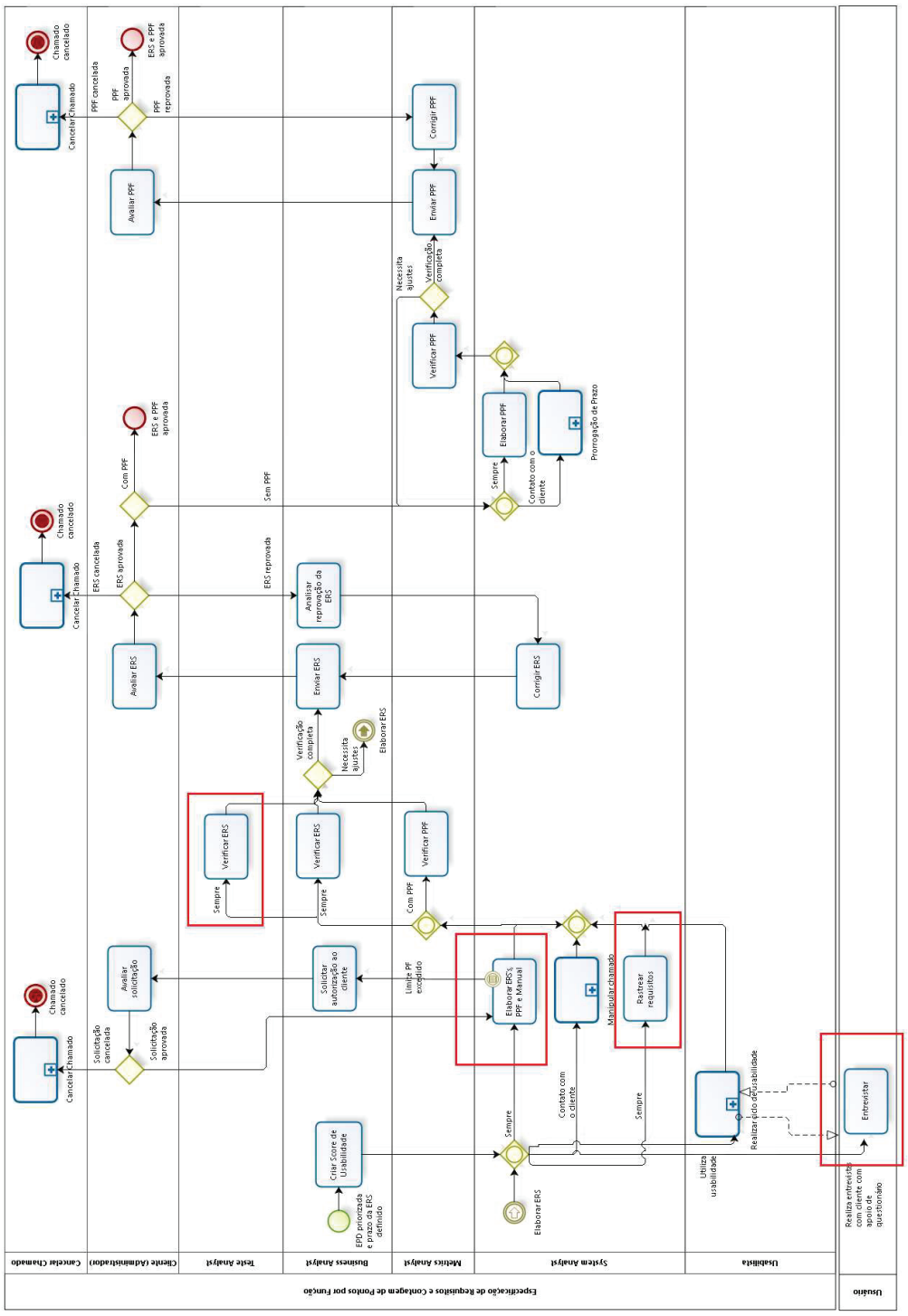
Em paralelo a elaboração da especificação de requisitos, o analista de sistemas realiza o ciclo de usabilidade definido pelo analista de negócios com o apoio do usabilista. O analista de negócio poderá utilizar um dos três ciclos de usabilidade definido: expresso, básico e o completo, os mesmos foram explicados na descrição do processo da empresa na seção anterior.

Após a elaboração da “ERS – Cliente”, ele deve elaborar a “ERS – Técnico” (documento sugerido nas melhorias, o mesmo encontra-se no Apêndice G), o documento de PPF (ponto por função) e o manual da solução (melhoria sugerida na seção 4.5.6), avaliando a viabilidade técnica da solução proposta junto ao desenvolvimento.

Finalizando as ERS's, é feito a validação das ERS’s pelo analista de negócios com a participação do analista de testes (melhoria sugerida na seção 4.5.5). Para a validação dos requisitos de software é utilizado o documento VRS (melhoria sugerida na seção 4.5.5), o mesmo pode ser encontrado no Apêndice E desta

monografia. Este documento funciona em forma de check list, garantindo a qualidade dos requisitos. O documento de PPF é enviado ao analista de métricas para validação. Caso as ERS's ou PPF precisem de ajustes, então voltam para o analista de sistemas fazer as devidas alterações. Se os documentos elaborados estão corretos, então o analista de negócios envia o documento de especificação de requisitos e o documento de PPF para a aprovação do cliente (dependendo do contrato). Caso reprovada às mesmas medidas são tomadas conforme descrito no processo da empresa na seção anterior. Na figura 15 pode-se observar o fluxo de requisitos alterado.

Figura 15. Processo de Levantamento de Requisitos - Novo



Fonte: das autoras, 2015

Após a ERS - Cliente e o documento PPF serem aprovados pelo cliente, a ERS - Cliente é enviada aos demais clientes para aprovação. Neste caso, os

clientes podem aprovar, reprovar, solicitar ajustes ou perder o prazo para se manifestar, caindo em algum destes casos as mesmas medidas do processo da empresa já descritos são tomadas.

Feito isso o chamado estará priorizado, estimado e aprovado e em seguida estará no backlog do desenvolvimento. Se a ERS foi aprovada e não foi desenvolvida após transcorrer o prazo de dois ciclos de desenvolvimento, a mesma deve ser avaliada e, se necessário ajustada, enviada novamente para aprovação do cliente.

5. CONCLUSÕES E TRABALHOS FUTUROS

Nesta seção será apresentado as conclusões referente ao estudo de caso que abrange está monografia, em seguida será apresentado sugestões e ideias para trabalhos futuros.

5.1. CONCLUSÕES

Durante a monografia foram apresentadas as justificativas, os objetivos e a fundamentação teórica para se atingir a meta do trabalho, que é a melhoria no processo de levantamento de requisitos para qualificar as especificações de testes.

A revisão bibliográfica apresentada nesta monografia foi de fundamental importância para dar embasamento nas sugestões de melhorias apontadas no processo de levantamento de requisitos e nos artefatos. A análise realizada na empresa sobre o processo que já existe também foi de grande relevância para constatar em que pontos o processo possuí fragilidades e o que pode ser melhorado.

Desde que a ideia desta monografia surgiu, foi visto que era necessário a aplicação de um questionário para as pessoas envolvidas no processo da empresa, para que elas pudessem dar suas opiniões baseadas em suas experiências, indicando fragilidades do processo e em que pontos poderia melhorar. Assim sendo, foram coletadas diversas opiniões de diferentes áreas, porém, que estavam dentro do mesmo processo, o processo descrito neste trabalho. Desta forma, aproximadamente 22 profissionais, a maior parte com mais de dois anos de empresa, expressaram as suas opiniões sobre as fragilidades do processo e descreveram algumas melhorias, baseadas em suas experiências e conhecimentos teórico-práticos. Foram encontradas dificuldades na análise dos questionários, pois, as 22 pessoas que responderam, algumas trabalham com clientes que possuem um contrato diferenciado que não se encaixa no processo utilizado na empresa, sendo

assim, essas pessoas não acompanham o processo como as outras. Mesmo assim, foi visto que era necessário o relato de todos os envolvidos, alguns com um pouco mais de experiência no processo utilizado, outros nem tanto.

A sugestão de melhorias no processo de especificação de requisitos foi levantada com base na revisão bibliográfica, nos questionários aplicados, e também com base na opinião das autoras desta monografia, pois ambas estão por dentro do processo da empresa e vivenciam cenários todos os dias de sua aplicação. Essas sugestões de melhorias fazem com que o processo se torne menos vulnerável a erros e atenda muitas das necessidades dos profissionais que estão dentro do processo da empresa, necessidades que foram relatadas nos questionários. As melhorias foram sugeridas não só para melhorar o processo de levantamento de requisitos, mas principalmente para facilitar o processo e o trabalho dos analistas de testes no momento de especificar um caso de teste.

Com este trabalho espera-se contribuir para melhorar a qualidade do processo de levantamento de requisitos, não só da empresa Softplan, mas de todas as empresas que possuem um processo frágil, e assim melhorar a qualidade do produto. Embora não necessariamente essas recomendações sejam aplicadas na empresa, este trabalho, com certeza, contribuiu para o aprendizado das autoras, ao forçar uma reflexão sobre atividades realizadas no dia a dia, observando, descrevendo e comparando essas práticas com a teoria da área.

5.2. TRABALHOS FUTUROS

Ao alcançar os objetivos impostos nesta monografia foram surgindo novas ideais para continuar este estudo de caso.

Uma das ideias como trabalho futuro é aplicar na empresa o novo processo proposto, apresentado nesta monografia. Após um tempo de aplicação seria possível constatar as vantagens e desvantagens que o novo processo trouxe, através da aplicação de novos questionários com os funcionários envolvidos.

Outra ideia para trabalhos futuros é sugerir melhorias em outras partes do processo como no desenvolvimento da solução ou na especificação de casos de teste.

Durante o desenvolvimento desta monografia também foi pensado em ferramentas que poderiam auxiliar em todas as partes do sistemas. Desta forma, como trabalho futuro é estudar a fundo ferramentas para auxiliar no controle de versões, especificações de requisitos, especificação de casos de teste e a gestão de defeitos.

REFERÊNCIAS

ALVES, Marcel F.,FERREIRA, Paulo C. G., **Sistema de automação de controle e serviços para setor comercial**. Belém – PA, 2006. 115 p.

BARROS, Aidil J. S., LEHFELD, Neide A. S. **FUNDAMENTOS DE METODOLOGIA: UM GUIA PARA INICIAÇÃO CIENTÍFICA.** 2. Ed. São Paulo. 2000.

BASTOS, Cleverson L., VICENTE, Keller. **Aprendendo a aprender: Introdução à Metodologia Científica.** 2. Ed. Rio de Janeiro. 1991.

CAETANO, Cristiano. **Gestão de Testes Ferramentas Open Source e Melhoras Práticas na Gestão de Teste.** 2011. Disponível em <http://www.garcia.pro.br/EngenhariadeSW/artigos%20engsw/teste/teste%20de%20software%20-%20artigo%202%20-%20rev3%20-%20gestao%20de%20teste%20de%20sw.pdf>. Acesso em 10 de junho de 2015.

DIAS NETO, Arilo Cláudio. **Artigo Engenharia de Software - Introdução a Teste de Software.** 2014. Disponível em <http://www.devmedia.com.br/artigo-engenharia-de-software-introducao-a-teste-de-software/8035>. Acesso em 20 de junho de 2015

ENGHOLM JR., Hélio. **Engenharia de software na prática.** 1. ed. São Paulo.2010.

IBM. **Rational Doors: Características**. Disponível em <http://www-03.ibm.com/software/products/pt/ratidoor>. Acesso em 13 de outubro de 2014

KOSCIANSKI, Andre, SOARES, Michel dos Santos. **Qualidade de Software.** 2 ed. Sao paulo. 2007

LEONEL, Vilson, MOTTA, Alexandre M. **Ciência e Pesquisa.** Livro didático. 2. Ed. Palhoça, 2007.

MACHADO, Felipe N. **Análise e Gestão de Requisitos de Software – onde nascem os sistemas.** 1. ed. São Paulo. 2011.

MACORATTI.NET – **Implementando soluções OOP**. Disponível em < http://www.macoratti.net/11/09/net_ioop.htm>. Acesso em 21 de junho de 2015.

MAFFEO, Bruno. **Engenharia de Software e Especificação de sistemas**. Rio de Janeiro. 1992

Os Ágeis e os Perplexos. Disponível em: < http://pt.slideshare.net/manoelp/os-eis-e-os-perplexos-manoel-pimentel-jorge-diz-presentation> Acesso em 18 de outubro de 2015

OMG. Object Management Group. Business Process Model and Notation. Disponível em <http://www.bpmn.org/ >. Acesso em 28 de outubro de 2015

PAULA FILHO, Wilson de Pádua. **Engenharia de Software.** e. LTC – Livros Técnicos e científicos editora S.A. 2001.

PETERS, James. **Engenharia de Software, Teoria e Prática.** Rio de Janeiro. 2001.

PONTES, Melissa Barbosa. **Introdução a Teste de Software.** 2011. Disponível em <http://www.garcia.pro.br/EngenhariadeSW/artigos%20engsw/teste/teste%20de%20software%20-%20artigo%205%20-%20rev11%20-%20introducao%20a%20teste%20de%20sw%20PESSOA.pdf>. Acesso em 10 de junho de 2015

SCHWABER, Ken, SUTHERLAND, Jeff. **Um guia definitivo para o Scrum: As regras do jogo.** Disponível em: < http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-Portuguese-BR.pdf> Acesso em 29 de novembro de 2015.

SOFTPLAN. **Site interno**. Disponível em: < https://colabore.softplan.com.br> Acesso em 29 de setembro de 2015b

SOMMERVILLE, Ian. **Engenharia de Software**. 9. Ed. São Paulo. 2011.

PRESSMAN, Roger S. **Engenharia de Software**, 6. ed. Porto Alegre. 2010.

PRESSMAN, Roger S. **Engenharia de Software, uma abordagem profissional.** 7. ed. Porto Alegre. 2011.

PRESSMAN, Roger S. **Engenharia de Software.** 3. ed. São Paulo. 1995.

SOFTPLAN. Disponível em: < https://softplan.com.br> Acesso em 8 de setembro de 2015a

TONSING, Sérgio L. **Engenharia de software – Análise e projeto de sistemas.** São Paulo. 2003.

ANEXOS

ANEXO A – DOCUMENTO INICIAL DE REQUISITOS (DIR)

Família de Sistemas: <nome>

Sistema: <sistema>

<mês>/<ano>

ÍNDICE

1. INTRODUÇÃO

<Breve descrição do propósito deste documento e seu conteúdo, sendo recomendado:

- Conter de um (1) a dois (2) parágrafos.>

2. ENVOLVIDOS <OPCIONAL>

<Lista das pessoas envolvidas no levantamento das informações>

Nome	Cargo e Instituição	Participação	E-mail

3. ESCOPO

<Esta seção deve identificar às áreas de negócio afetadas, contendo lista das características que deverão ser atendidas pela solução a ser especificada. Faz parte da descrição do escopo:

- Identificar as necessidades;
- Modelagem no fluxo de negócio a ser atendido (BPMN) (opcional);
- Lista de requisitos de negócio identificados;
- Impactos relevantes.

Não faz parte deste documento a identificação de requisitos funcionais, requisitos não funcionais, casos de usos, regras de sistema, mensagens e outros itens da fase de especificação de requisitos e projeto da solução.

Este seção pode ser subdividida em:

- **Necessidades**: descreve as necessidades identificadas, sendo descritas de forma unitária com seus respectivos identificadores;
- **Fluxo de Negócio** [*opcional*]: descreve o fluxo de negócio atual do cliente, permitindo identificar quais processos serão atendidos por este escopo de levantamento;
- **Características a serem atendidos**: lista de características a ser atendida pela especificação de requisitos;

- **Impactos**: descrever quais usuários, setores e tarefas serão afetados pelo escopo do projeto. Podendo ser destacado quais funcionalidades do sistema existente serão afetadas;
- **Restrições**: descrever a lista de requisitos de projetos que impactam na solução ou na implantação.>

4. FORA DE ESCOPO

<Apresenta listas dos itens levantados que estão fora do escopo atendidos pela solução proposta, devendo ser descrito o motivo.
É recomendada a utilização de tabela contendo a restrição e seu motivo. Exemplo:

Id	Item Levantado	Motivo

5. REVISÕES

Descrição	Versão	Data	Autor
Elaboração do documento	1		<autor>
<Descrição da alteração>	2		<autor>

ANEXO B – ESTIMATIVA PRELIMINAR DE DEMANDA (EPD)

EPD-<Unidade>-<Sistema>.<aaaa>.<nnnn>

EPD - Estimativa Preliminar de Demanda

SOFTPLAN

De:	
Para:	
A/C:	
Assunto:	Orçamento Estimativo
Sistema:	

Com base nas solicitações de mudança <número da(s) SALT(s)>, a seguir são apresentadas as estimativas necessárias para a realização da manutenção evolutiva do sistema <nome do sistema>.

[ ] Até 20 PF – Prazo: até 9 dias úteis para elaboração da ERS;

[ ] De 21 a 30 PF – Prazo: até 14 dias úteis para elaboração da ERS;

[ ] De 31 a 40 PF – Prazo: até 18 dias úteis para elaboração da ERS;

[ ] De 41 a 50 PF – Prazo: até 23 dias úteis para elaboração da ERS;

[ ] De 51 a 60 PF – Prazo: até 27 dias úteis para elaboração da ERS;

[ ] De 61 a 70 PF – Prazo: até 32 dias úteis para elaboração da ERS;

[ ] De 71 a 100 PF – Prazo: até 33 dias úteis para elaboração da ERS;

[ ] Acima de 100 PF – Prazo a ser acordado entre as partes para elaboração da ERS.

Ficamos no aguardo de aprovação deste documento para dar início à elaboração do documento de Especificação de Requisito de Software (ERS).

Salientamos que este orçamento é estimativo e que pode apresentar divergências (a mais ou a menos) do orçamento definitivo, visto que o segundo possui maior nível de detalhamento.

Em anexo, é apresentado o contexto da(s) solicitação(ões) e análise de impacto preliminar.

Colocamo-nos à disposição para quaisquer esclarecimentos adicionais.

Atenciosamente,

_____

< Representante Empresa>

<Cidade>, <dd> de <mm> de <aaaa>.

Anexo 1 – Análise de Escopo Preliminar

A seguir apresentamos o conjunto de SALTs de atendimentos nesta manutenção evolutiva. Para cada SALT é feito um breve descritivo do entendimento da solicitação e escopo preliminar a ser atendimento. As informações apresentadas neste documento não descrevem a solução a ser desenvolvida.

SALT <Número da SALT>:
Escopo Preliminar:
Módulo/Função afetados:

<Orientações para montagem da "Análise de Escopo Preliminar".

O anexo "Análise do Escopo Preliminar" apresenta os principais pontos que serão afetados, possuindo o seguinte formato:

- **SALT <Número da SALT>** - Texto descritivo do entendimento da solicitação encaminhada pela SALT, enfatizando contatos prévios realizados para fechamento do entendimento da demanda.
- **Escopo Preliminar**: Texto descritivo enfatizando os principais pontos dos sistemas que farão parte do escopo da solicitação, enfatizando tela, funções e pontos críticos.
- **Módulo/Função afetados**: Neste tópico deverá ser destacado os Módulos e Funções afetados.

Nota 1: O uso de telas do sistema é opcional neste anexo, caso seja usado, o mesmo deve ter o objetivo de enfatizar o local do sistema, ficando vedada qualquer referência de solução técnica a ser implementado.

Exemplos:

1. **SALT 999999/9** – Tornar obrigatório o campo Classe.

Escopo Preliminar: Para atendimento da solicitação serão ajustados os seguintes pontos do sistema:

- Tornar o campo obrigatório nas telas:
  - Tela de "Cadastro de Processo do 1º Grau";
  - Tela de "Cadastro de Processo do 2º Grau";
- Ajustar o serviço de integração entre MP e TJ para tornar o campo obrigatório na importação.

Módulo: Processo
Funcionalidade: Cadastro de Processo

1. **SALT 999999/1** - O cliente solicita nova implementação no sistema para atender a Seção de Leilões da Vara de Execuções Fiscais de São Paulo.

Escopo Preliminar: Criar nova funcionalidade no SAJ/PG que permita as seguintes funcionalidades:

- Módulo de leilões contendo agendamento, cadastro, redesignação e publicação;
- Ajustar fluxo de trabalho para novas funcionalidades;

- Criar relatórios de leilões;
- Ajustar cadastros de processos para gerenciamento de das informações Armas e Bens para lote de leilões;

Módulo: Processo
Funcionalidade: Cadastro de Processo>

ANEXO C – MODELO DE CONTAGEM DE PONTOS POR FUNÇÃO

FUNÇÕES DE DADOS		
RESUMO DA CONTAGEM		
TIPO	QTDE	
AIE	0	
ALI	0	
TIPO		
TIPO	QTDE	PF
INCLUSÃO	0	0,0
ALTERAÇÃO	0	0,0
EXCLUSÃO	0	0,0
TOTAL PONTOS DE FUNÇÃO		0,0
0		

Grupo de Dados	Tipo de dados	Operação	DER	RLR	Complex.	PF	Descritivo de implementação Realizada

CÁLCULO_PF		
TOTAL DE FUNÇÃO BRUTO		0
Operação	FI	PONTOS FUNÇÃO
INCLUSAO	1,00	-
ALTERAÇAO	0,50	-
EXCLUSÃO	0,40	-
CONVERSÃO	1,00	-
Itens Não Mensuráveis (PF)	1,00	-
TOTAL DE PONTOS DE FUNÇÃO		0,00
FATOR DE AJUSTE		1,00
TOTAL DE PONTOS DE FUNÇÃO AJUSTADO		0,00

ANEXO D – ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE (ERS)

ERS-UNJ-PJ-aaaa.xxxx-

ERS - Especificação de Requisitos de Software

SOFTPLAN

ÍNDICE

1. INTRODUÇÃO

<Breve introdução do que o cliente deseja>

2. ALTERAÇÃO

2.1. ALTERAÇÃO 1

<PARTE 1 DO SISTEMA que está sendo alterado com uma explicação. Em seguida vem os requisitos, regras de negócio, protótipos e regras de tela.>

4. REVISÃO

Descrição Versão	Data	Autor	

<nessa tabela é informado quem produziu, quem validou e todas as alterações feitas no documento>

APÊNDICES

APÊNDICE A – CRONOGRAMA

Atividades	2015			
	Março	Abril	Maio	Junho
f) Elaboração do capítulo 1: Introdução (objetivos, problemática e justificativa) e leitura de conteúdo bibliográfico.	X			
g) Elaboração do capítulo 2: Referência bibliográfica para a elaboração do trabalho e leitura de conteúdo bibliográfico.	X	X	X	X
h) Elaboração do capítulo 3: Método, caracterização da pesquisa, etapas metodologias e proposta da solução.			X	X
i) Entrega do capítulo 1.		X		
j) Entrega do capítulo 2.			X	
k) Entrega do capítulo 3.			X	
l) Entrega da primeira versão do Trabalho de Conclusão de Curso.				X

Atividades	2015				
	Julho	Agosto	Setembro	Outubro	Novembro
m) Elaboração do capítulo 4: estudo de caso.	X	X	X	X	
n) Elaboração e aplicação de questionários para analistas de testes.			X		
o) Análise de resultado dos questionários.			X	X	
p) Reunir dados e elaborar diagramas de acordo com os dados obtidos.				X	
q) Revisão do Trabalho de Conclusão de Curso.				X	X
r) Entrega final do Trabalho de Conclusão de Curso.					X

APÊNDICE B - QUESTIONÁRIO

Melhorias na Especificação de Requisitos para elaboração de Casos de Teste

Caro colega, gostaríamos que você respondesse este formulário para nos ajudar na elaboração do nosso TCC, o tema é o do título deste formulário. Respondê-lo levará apenas entre 3 e 10 minutos do seu tempo. Obrigadas pela sua colaboração. Alcieny Nunes e Renata Duarte

*Obrigatório

1. Qual seu cargo na softplan? *
   - ☐ Testador
   - ☐ Analista de teste
   - ☐ Analista de sistemas
   - ☐ Analista implementador
   - ☐ Outro: ☐

2. A quanto tempo trabalha nesta empresa?
   - ☐ Menos de 2 meses
   - ☐ Entre 2 e 6 meses
   - ☐ Entre 1 a 2 anos
   - ☐ Mais de 2 anos

3. Você acha que o processo de software da empresa está bem definido e ele é completo?
   - ☐ Não concordo totalmente
   - ☐ Não concordo parcialmente
   - ☐ Indiferente
   - ☐ Concordo parcialmente
   - ☐ Concordo totalmente

Justifique sua resposta referente a questão anterior:
R:

4. Você acha que o processo de teste de sua unidade precisa de alguma melhoria?
   - ☐ Não concordo totalmente
   - ☐ Não concordo parcialmente
   - ☐ Indiferente

- ☐ Concordo parcialmente
- ☐ Concordo totalmente

Se você acha que sim, qual melhoria o processo de teste deveria ter?
R:

5. Você acha que o processo de análise de requisitos da sua empresa precisa de alguma melhoria?

- ☐ Não concordo totalmente
- ☐ Não concordo parcialmente
- ☐ Indiferente
- ☐ Concordo parcialmente
- ☐ Concordo totalmente

Se você acha que sim, qual melhoria o processo de análise de requisitos deveria ter?
R:

6. Você acredita que tem algum modo de melhorarmos o levantamento de requisitos e o seu registro, dessa forma facilitando a elaboração de casos de teste?
Poderia nos dar exemplos e sua visão sobre o assunto?
R:

7. Na sua opinião quais são os principais fatores que influenciam na testabilidade de um requisito?
R:

8. Você acha que os requisitos devem ter um alto nível de granularidade?

- ☐ Não concordo totalmente
- ☐ Não concordo parcialmente
- ☐ Indiferente
- ☐ Concordo parcialmente
- ☐ Concordo totalmente

Justifique sua resposta referente a questão anterior.
R:

9. Você acha que a documentação elaborada no decorrer do processo de desenvolvimento de software é suficiente?

- ☐ Não concordo totalmente
- ☐ Não concordo parcialmente
- ☐ Indiferente
- ☐ Concordo parcialmente
- ☐ Concordo totalmente

Se não, indique que tipo de documentação é necessária para melhorar o processo.
R:

10. Você tem mais alguma observação a fazer referente ao tema citado?
R:

APÊNDICE C – ANÁLISE DAS RESPOSTAS DO QUESTIONÁRIO

3. Você acha que o processo de software da empresa está bem definido e ele é completo?

R:

- Existe um processo porém ele é oneroso e muitos colaboradores não seguem. Quando existe algo no processo que force a execução de cada tarefa o processo pode ser seguido com mais facilidade.
- Processo não atende todas as necessidades. Pouca coisa foi definida e muitas vezes burlado o pouco definido
- Processos devem ser atualizados e amadurecidos.
- Processo vem com o crescimento da empresa.
- Pensa-se e gasta-se muito tempo em qual ferramenta usar do que no próprio processo. Ferramenta é apenas um acessório para facilitar o trabalho quando o processo é bem definido e maduro.
- Processo deve ficar independente da capacidade de especialistas, para dessa forma ficar bem alinhado entre todos.
- A solicitação de burlar os processos muitas vezes vem de cargo de chefia
- Falta de comunicação
- Definição de atendimento a SLA
- Registro de lições aprendidas para ajudar na melhoria do processo.

4. Você acha que o processo de teste de sua unidade precisa de alguma melhoria?

R:

- É reflexo de falta de documentação. A partir do momento que as coisas forem bem definidas e documentadas na fase de análise, proporcionará a melhoria que falta no processo de teste.
- Maior sincronização com as atividades desenvolvidas. Muitas definições são feitas entre analista e implementador e, pelo processo atual, não chegam as informações necessárias para o teste ser efetuado de forma adequada.
- Definição de um processo sólido que seja realemnte cumprido a risca.
- Deveria haver uma preocupação maior com a especificação de testes e mapeamento de cobertura de testes nos produtos.
- Deveria haver mais roteiros de testes, com casos de teste para a maioria dos modulos do sistema, ou principalmente os que são mais alterados, seguindo sempre as especificações de requisitos, coisa que hoje na grande maioria das vezes não á feito.

5. Você acha que o processo de análise de requisitos da sua unidade precisa de alguma melhoria?

R:

- Entendimento da necessidade do cliente, em muitos casos é observado que é apenas transcrita a solicitação do cliente, mas não é verificado os motivos

que levaram o cliente a solicitar essa demanda, e a solução pode estar em outra forma de controle, que pode ser diferente do que foi solicitado.
- O processo também deveria prever especificação de requisitos do legado, caso a alteração seja em cima do legado.
- Cada analista tem seu próprio modo de escrever seus requisitos, tornando um pouco difícil a interpretação do que o analista que dizer ou representar.
- Amadurecimento com foco no que realmente é necessário.
- Analistas de negócio efetivamente cumprindo seu papel de elucidar as necessidades e dúvidas dos clientes. Esta interface inicial é realizada quase sempre pelos analistas de sistemas.
- Deve ser realizado uma análise de impacto em todo o sistema pra prevenir alterações que impactariam em uma parte muito maior que o previsto.

6. Você acredita que tem algum modo de melhorarmos o levantamento de requisitos e o seu registro, dessa forma facilitando a elaboração de casos de teste?

R:

- A especificação de requisitos poderia citar exemplos de fluxo do cliente para facilitar a elaboração de casos de teste com a realidade do cliente
- Padronização no levantamento de requisitos.
- Não permitir andamento do projeto sem que as etapas de um processo definido seja seguida.
- Sim, com o uso de ferramentas mais consolidadas no mercado e com a inclusão de alguma forma de obrigatoriedade do levantamento de requisitos no processo de software.
- Acredito que uma boa documentação e a interligação entre os documentos gerados ajudaria.
- Acredito que uma das formas de melhorarmos o levantamento de requisitos é fazer com que não somente essa função fique nas mãos do analista de sistemas para fazer a validação dos mesmos, mas sim testadores, desenvolvedores participem dessa validação.
- Um ponto importante a melhorar seria a rastreabilidade do requisito, pois é importante para identificar quem, quando e porque o requisito mudou. Atualmente vejo que não temos essa rastreabilidade, o que prejudica saber qual requisito o sistema deve atender.
- Mais envolvimento do cliente

7. Na sua opinião quais são os principais fatores que influenciam na testabilidade de um requisito?

R:

- Clareza no que foi descrito, e não omitir informações por entender que já esta subentendido.
- Primeiramente o ambiente, depois o conhecimento e formas de automatizar esse teste.
- Especificação, experiência

8. Você acha que os requisitos devem ter um alto nível de granularidade?
R:

- Se for muito granular pode gerar muitos artefatos separados que podem confundir durante a modelagem do sistema, é preciso ter um bom senso de saber em que momento um artefato deve ser quebrado em 2 ou basta melhorar o texto e continuar com apenas 1.
- Dividir para conquistar.
- Com alta granularidade, cada caso de uso a ser desenvolvido tem seu escopo reduzido, facilitando a visão geral de cada item.
- Auto nível de granularidade possibilita o entendimento claro e objetivo por todos os envolvidos e deixando totalmente claro o que e o porque deve ser feito.
- Usando se a teoria do dividir para conquistar acredito que seria interessante, pois quanto mais quebrarmos os requisitos em níveis de granularidade ficará mais fácil para confeccionar casos de testes e o desenvolvimento possa ocorrer.
- Acredito que o requisito deve ter também, quando lido isoladamente, sentido completo, sendo capaz de indicar o comportamento esperado. Se por acaso isso não ocorrer, pode ser um indício de que a granularidade está muito alta.

9. Você acha que a documentação elaborada no decorrer do processo de desenvolvimento de software é suficiente?
R:

- A documentação gerada pode ser suficiente desde que seja bem elaborada, que contenham todas as informações de forma clara e objetiva.
- Documentação de inicial de negócio. Documentação de requisitos. Acompanhamento dos testes desde as fases iniciais de levantamento.
- Seria necessária uma documentação contemplando as funcionalidades legado do nosso sistema, que, atualmente, ou está muito desatualizada, incompleta ou inexistente.
- É importante que se tenha uma rastreabilidade efetiva que seja integrada em todas as fases do ciclo de desenvolvimento.
- Não há documentação de levantamento de requisitos no cliente.
- Seria muito importante ter ao menos a documentação dos métodos implementados.
- O documento deve ser fácil de pesquisar e navegar, para que seja possível recuperar o que se deseja rapidamente.
- A mesma documentação enviada para o cliente é a documentação que é repassada para a equipe de desenvolvimento. Acho essa documentação totalmente incompleta e não suficiente para o desenvolvimento de novas funcionalidades. Muitos detalhes, que o cliente não precisa saber, mas que são fundamentais para o desenvolvimento não são colocado neste documento. Deveriam haver dois tipos de documentação.

10. Você tem mais alguma observação a fazer referente ao tema citado?
R:

- Acredito que poderia se pensar em uma melhoria de processo, com alternativas mais ágeis de realizar as atividades, onde todos os envolvidos participem do processo desde o início. E também gere mais agilidade e comunicação entre as partes.
- Sempre percebo a necessidade de saber como a especificação foi desenvolvida e como configurar ela, pois na hora de instalar no cliente é um tal, erro de configuração, precisa fazer isso ou aquilo, precisa habilitar tal parâmetro, enfim.

APÊNDICE D – GRÁFICOS COMPLETOS

Gráfico A – Respostas da pergunta número 3.

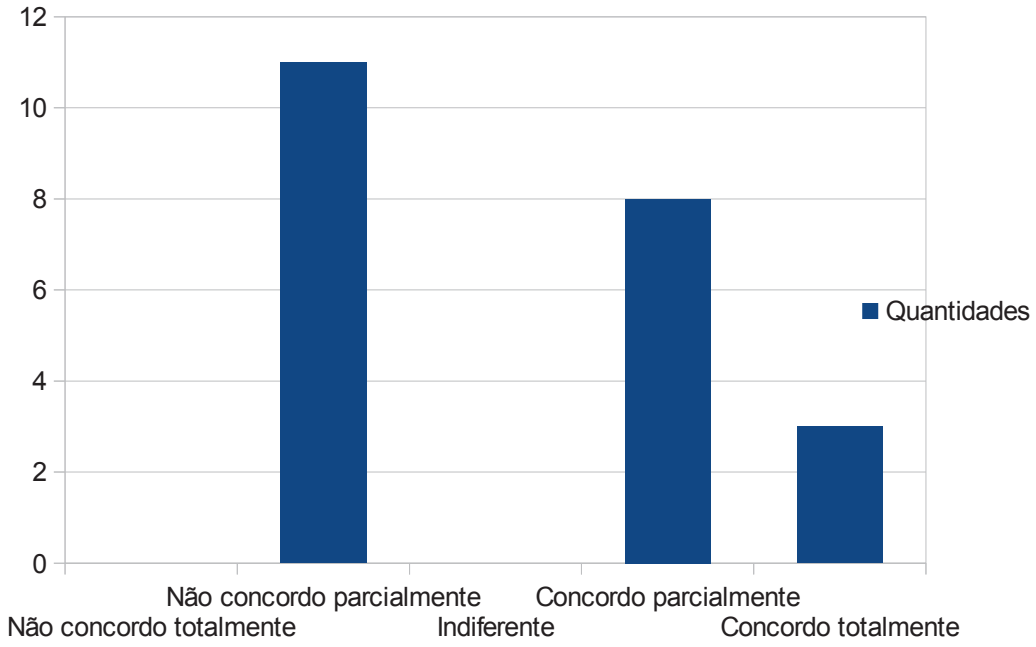


Gráfico B – Respostas da pergunta número 4.

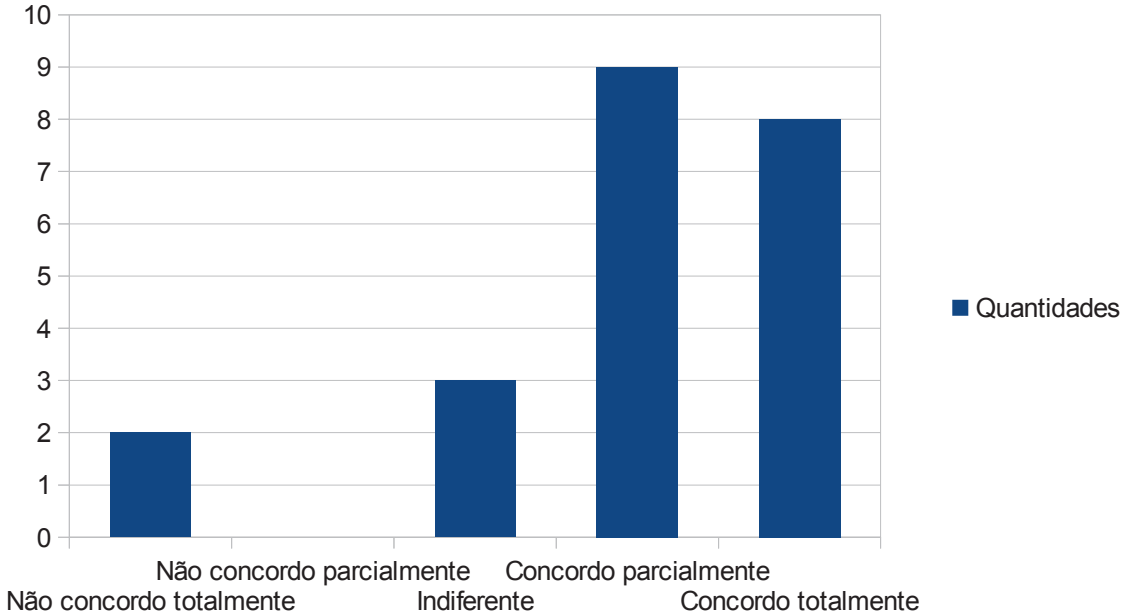


Gráfico C – Respostas da pergunta número 5.

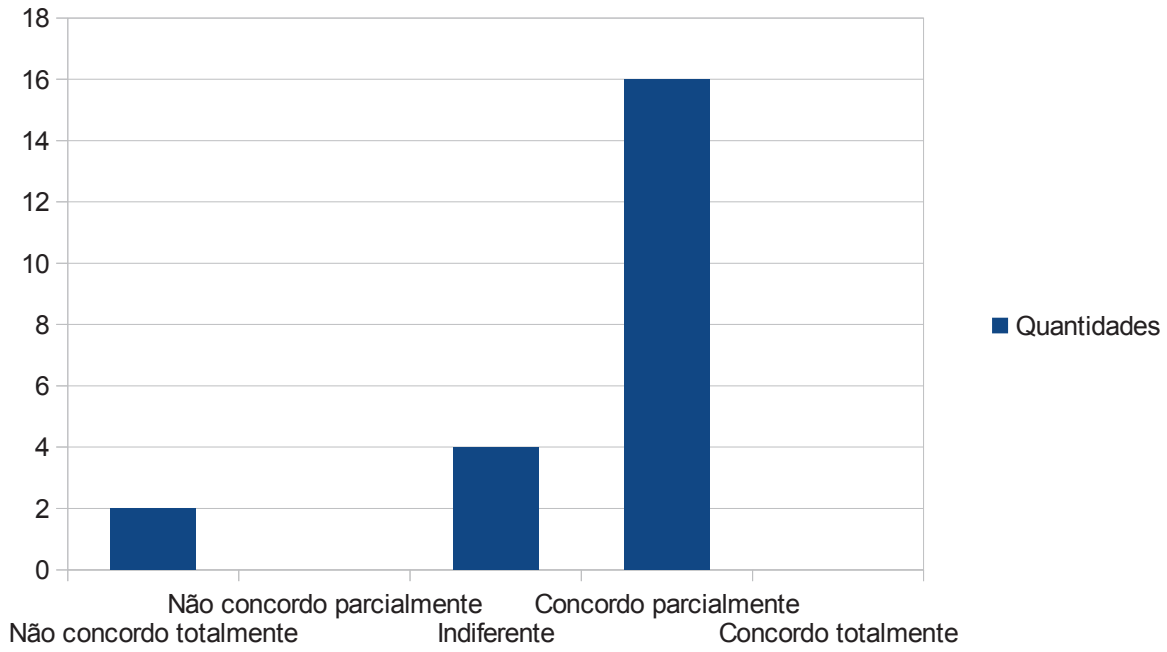


Gráfico D – Respostas da pergunta número 8.

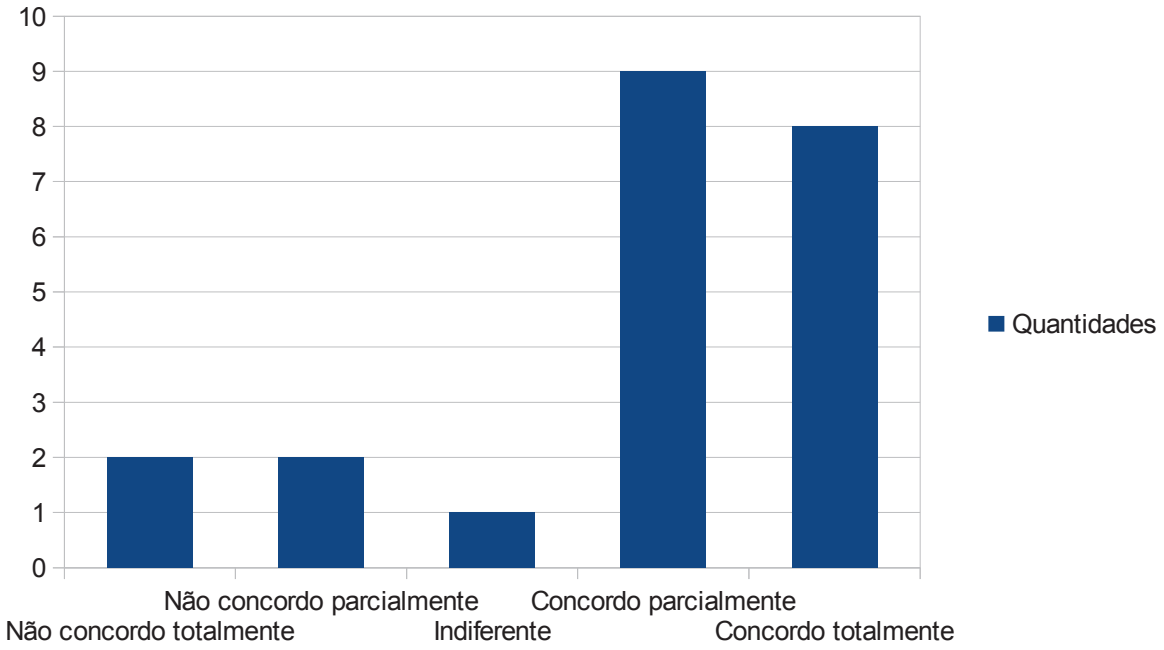
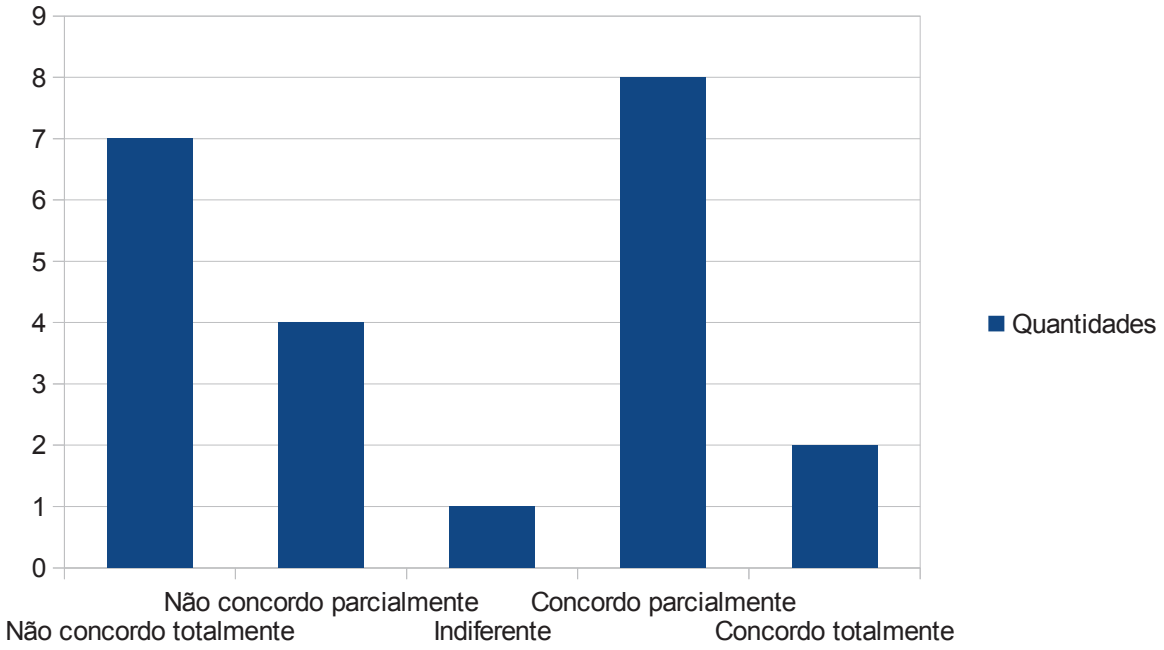


Gráfico E – Respostas da pergunta número 9.



APÊNDICE E – VRS – DOCUMENTO DE VALIDAÇÃO DE REQUISITOS DE SOFTWARE

VRS-<Unidade>-<Sistema>.<aaaa>.<nnnn>

VRS – Documento de Validação de Requisitos de Software

SOFTPLAN

Atendimento:
Analista de sistemas responsável:
Analista de negócios:
Analista de testes:
ERS nº:

1) Cada um dos requisitos é consistente com os objetivos globais para o sistema/produto?
( ) SIM ( ) NÃO

2) Todos os requisitos foram especificados no nível de abstração apropriado? Ou seja, algum dos requisitos fornece um nível de detalhe técnico inapropriado no estágio atual?
( ) SIM ( ) NÃO

3) O requisito é realmente necessário ou representa um recurso adicional que talvez não seja essencial para o objetivo do sistema?
( ) SIM ( ) NÃO

4) Cada um dos requisitos é limitado e sem ambiguidade?
( ) SIM ( ) NÃO

5) Cada um dos requisitos possui atribuição? Ou seja, uma fonte (em geral, um individuo específico) é indicada para cada requisito?
( ) SIM ( ) NÃO

6) Algum dos requisitos conflita com outros requisitos?
( ) SIM ( ) NÃO

7) Cada um dos requisitos é atingível no ambiente técnico que irá abrigar o sistema ou produto?
( ) SIM ( ) NÃO

8) Cada um dos requisitos pode ser testado, ou seja, se encaixa nas medidas de testabilidade?
( ) SIM ( ) NÃO

9) O modelo de requisitos reflete, de forma apropriada, a informação, função e comportamento do sistema a ser construído?
( ) SIM ( ) NÃO

10) O modelo de requisitos foi "dividido" para expor progressivamente informações mais detalhadas sobre o sistema?
( ) SIM ( ) NÃO

11) Os padrões de requisitos foram utilizados para simplificar o modelo de requisitos? Todos os padrões foram validados adequadamente? Todos os padrões são consistentes com os requisitos do cliente?
( ) SIM ( ) NÃO

Observação:

_____

_____

_____

APÊNDICE F – ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE – CLIENTE

ERS-UNJ-PJ-AAAA-XXXX

ERS - Especificação de Requisitos de Software - Cliente

SOFTPLAN poligraph

ÍNDICE

INTRODUÇÃO

<Breve introdução da necessidade do usuário>

2. ALTERAÇÃO

2.1. ALTERAÇÃO 1

<PARTE 1 DO SISTEMA que está sendo alterado com uma explicação. Em seguida vem os requisitos, regras de negócio, protótipos e regras de tela.>

REGRAS DE NEGÓCIOS	DESCRIÇÃO
<ATRIBUIR O NOME DA REGRA DE NEGÓCIO.>	<DESCREVER A REGRA DE NEGÓCIO PARA ATENDER A NECESSIDADE DO CLIENTE.>

REQUISITOS FUNCIONAIS	DESCRIÇÃO
<ATRIBUIR O NOME DO REQUISITO FUNCIONAL>	<DESCREVER O REQUISITOS FUNCIONAL PARA ATENDER A NECESSIDADE DO CLIENTE.>

2.1.1. PROTÓTIPOS:

<COLOCAR O PROTÓTIPO DA ALTERAÇÃO DE DETERMINADA TELA, DESCREVENDO DE QUE TELA PERTENCE ESTE PROTÓTIPO>

3. REVISÃO

Autor	Data	Versão	Descrição
			<Escrever aqui alguma sugestão para melhorar a especificação, ou o motivo pelo qual ele foi alterado>

<Nessa tabela é informado quem produziu, quem validou e todas as alterações feitas no documento>

APÊNDICE G – ESPECIFICAÇÃO DE REQUISITOS DE SOFTWARE - TÉCNICO

ERS-UNJ-PJ-AAAA-XXXX

ERS - Especificação de Requisitos de Software - Técnico

SOFTPLAN poligraph

ÍNDICE

1. INTRODUÇÃO

<Breve introdução da necessidade do usuário>

2. ALTERAÇÃO

2.1. ALTERAÇÃO 1

<PARTE 1 DO SISTEMA que está sendo alterado com uma explicação. Em seguida vem os requisitos, regras de negócio, protótipos e regras de tela.>

REGRAS DE NEGÓCIOS	DESCRIÇÃO
<ATRIBUIR O NOME DA REGRA DE NEGÓCIO.>	<DESCREVER A REGRA DE NEGÓCIO PARA ATENDER A NECESSIDADE DO CLIENTE>
	<DESCREVER AO MENOS UM EXEMPLO PARA VALIDAR A REGRA DE NEGÓCIO>

REQUISITOS FUNCIONAIS	DESCRIÇÃO
<ATRIBUIR O NOME DO REQUISITO FUNCIONAL>	<DESCREVER O REQUISITOS FUNCIONAL PARA ATENDER A NECESSIDADE DO CLIENTE.>
	<DESCREVER AO MENOS UM EXEMPLO PARA VALIDAR A REQUISITO FUNCIONAL>

REQUISITOS NÃO FUNCIONAIS	DESCRIÇÃO
<ATRIBUIR O NOME DO REQUISITO NÃO FUNCIONAL.>	<DESCREVER O REQUISITO NÃO FUNCIONAL PARA ATENDER A NECESSIDADE DO CLIENTE.>
	<DESCREVER AO MENOS UM EXEMPLO PARA VALIDAR A O REQUISITO FUNCIONAL>

REGRAS DE TELA	DESCRIÇÃO
<ATRIBUIR O NOME DA REGRA DE TELA.>	<DESCREVER A REGRA DE TELA PARA ATENDER A NECESSIDADE DO CLIENTE.>
	<DESCREVER AO MENOS UM EXEMPLO PARA VALIDAR A REGRA DE TELA>

MENSAGENS	DESCRIÇÃO
<ATRIBUIR NOME DO ARTEFATO QUE GEROU A MENSAGEM> (REGRA DE TELA, REGRA DE NEGÓCIO.)	<COLOCAR A DESCRIÇÃO DA MENSAGEM E DESCREVER QUAL REGRA QUE VAI GERAR ESTÁ MENSAGEM.>
	<DESCREVER AO MENOS UM EXEMPLO PARA VALIDAR A MENSAGEM QUE SERÁ EXIBIDA NA TELA.>

2.1.1. PROTÓTIPOS:

<COLOCAR O PROTÓTIPO DA ALTERAÇÃO DE DETERMINADA TELA, DESCREVENDO DE QUE TELA PERTENCE ESTE PROTÓTIPO>

2.1.2. IMPACTO

<DESCREVER AQUI O QUE CADA ALTERAÇÃO E IMPLEMENTAÇÃO DE REGRA DE NEGÓCIO, REGRA DE TELA E AFINS PODEM IMPACTAR NO SISTEMA, DE FORMA QUE O IMPLEMENTADOR DESENVOLVA COM CONSCIÊNCIA E QUE ANTES DE LIBERAR DEVE SE CERTIFICAR DE QUE ESSES PONTOS APONTADOS AQUI NÃO SOFRERAM ALTERAÇÃO.>

2.1.3. ALTERAÇÕES TÉCNICAS

<COLOCAR AQUI AS ALTERAÇÕES TÉCNICAS QUE PODEM SER ANALISADAS PELO ANALISTA DE SISTEMA, PARÂMETROS QUE DEVEM SER ALTERADOS, SCRIPTS QUE DEVEM SER CRIADOS, COLUNAS E TABELAS DE BANCOS QUE DEVEM SER CRIADAS OU ALTERADAS.>

2.1.4. DOCUMENTOS RELACIONADOS

<COLOCAR AQUI OS DOCUMENTOS RELACIONADOS COM ESSA ESPECIFICAÇÃO. (EPD, DIR, VDR, PPF).>

3. REVISÃO

Autor	Data	Versão	Descrição
			<Escrever aqui alguma sugestão para melhorar a especificação, ou o motivo pelo qual ele foi alterado>

<Nessa tabela é informado quem produziu, quem validou e todas as alterações feitas no documento>

APÊNDICE H – TERMO DE COMPROMISSO DA EMPRESA

UNIVERSIDADE DO SUL DE SANTA CATARINA
Sistemas de Informação
Av. Pedra Branca, 25 – Cidade Universitária Pedra Branca –
CEP 88132-000 – Palhoça - SC

DECLARAÇÃO DE CIÊNCIA E CONCORDÂNCIA DAS INSTITUIÇÕES ENVOLVIDAS NO TRABALHO DE CONCLUSÃO DE CURSO

Local e data: Florianópolis, 17 de novembro de 2015

O representante da instituição **Softplan Poligraph**, objeto do estudo de caso dos estudantes Alciony Nunes e Renata Aparecida Duarte, executores do **Trabalho de Conclusão do Curso Ciência da Computação**, intitulado Melhorias no processo de levantamento de requisitos para a qualificação dos casos de teste, declara estar cientes e de acordo com o desenvolvimento desse trabalho.

O mesmo também autoriza [X], não autoriza [ ] o uso do nome da instituição no referido Trabalho de Conclusão de Curso.

Ass. do(s) aluno(s) executores do Trabalho de Conclusão de Curso (UNISUL)

Ass. e carimbo do Coordenador do Curso

Ass. e carimbo do responsável da Instituição envolvida no estudo

Adaptado do Comitê de Ética em Pesquisa – UNISUL (CEP UNISUL)