UNIVERSIDADE ESTADUAL DO TOCANTINS
CÂMPUS DE PALMAS
CURSO DE SISTEMAS DE INFORMAÇÃO

DEFINIÇÃO DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DO ESC/NIT DA UNITINS

IAGO BATISTA ANTUNES LEOBAS

Palmas - TO

2022

UNIVERSIDADE ESTADUAL DO TOCANTINS
CÂMPUS DE PALMAS
CURSO DE SISTEMAS DE INFORMAÇÃO

DEFINIÇÃO DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DO ESC/NIT DA UNITINS

IAGO BATISTA ANTUNES LEOBAS

Trabalho de Conclusão do Curso apresentado ao Curso de Sistemas de Informação da Universidade Estadual do Tocantins - UNITINS como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação.

Palmas - TO
2022

CURSO DE SISTEMAS DE INFORMAÇÃO

DEFINIÇÃO DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DO ESC/NIT DA UNITINS

IAGO BATISTA ANTUNES LEOBAS

Trabalho de Conclusão do Curso apresentado ao Curso de Sistemas de Informação da Universidade Estadual do Tocantins - UNITINS como parte dos requisitos para a obtenção do grau de Bacharel em Sistemas de Informação.

Me. Fredson Vieira Costa
Orientador

Me. Carlos Henrique Corrêa Tolentino
Examinador

Me. Jeferson Morais da Costa
Examinador

Palmas - TO

2022

Dados Internacionais de Catalogação na Publicação (CIP) Sistema de Bibliotecas da Universidade Estadual do Tocantins

L576d

LEOBAS, Iago Batista Antunes
DEFINIÇÃO DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DO ESC/NIT DA UNITINS. Iago Batista Antunes Leobas. - Palmas, TO, 2022

Monografia Graduação - Universidade Estadual do Tocantins – Câmpus Universitário de Palmas - Curso de Sistemas de Informação, 2022.

Orientador: Fredson Vieira Costa

1. Processos. 2. Desenvolvimento. 3. Software. 4. BPMN.

CDD 610.7

Elaborado pelo sistema de geração automatica de ficha catalográfica da UNITINS com os dados fornecidos pelo(a) autor(a).

ATA DE DEFESA DE TRABALHO DE CONCLUSÃO DE CURSO DO CURSO DE SISTEMAS DE INFORMAÇÃO DA FUNDAÇÃO UNIVERSIDADE ESTADUAL DO TOCANTINS - UNITINS

Aos **14** dias do mês de **Dezembro** de **2022**, reuniu-se via Google Meet, às **16h**, sob a Coordenação do Professor **Fredson Vieira Costa**, a banca examinadora de Trabalho de Conclusão de Curso em Sistemas de Informação, composta pelos examinadores Professor **Fredson Vieira Costa** (Orientador), Professor **Jeferson Morais da Costa** e Professor **Carlos Henrique Correa Tolentino**, para avaliação da defesa do trabalho intitulado **"Definição do Processo de Desenvolvimento de Software do ESC/NIT da Unitins"** do acadêmico **Iago Batista Antunes Leobas** como requisito para aprovação na disciplina Trabalho de Conclusão de Curso (TCC). Após exposição do trabalho realizado pelo acadêmico e arguição pelos Examinadores da banca, em conformidade com o disposto no Regulamento de Trabalho de Conclusão de Curso em Sistemas de Informação, a banca atribuiu a pontuação: 9,5.

Sendo, portanto, o Acadêmico: (X) Aprovado ( ) Reprovado

Assinam esta Ata:

Professor Orientador: Fredson Vieira Costa

Examinador: Carlos Henrique Corrêa Tolentino

Examinador: Jeferson Morais da Costa

Fredson Vieira Costa

Presidente da Banca Examinadora

Coordenação do Curso de Sistemas de Informação

Coordenação do Curso de Sistemas de Informação

Dedico este trabalho àqueles que acreditaram no meu esforço e dedicação ao decorrer dessa longa jornada acadêmica.

AGRADECIMENTOS

Agradeço primeiramente a Deus pelo privilégio da vida e pela oportunidade de me dedicar aos estudos.

Agradeço a minha família, namorada e amigos pelo apoio incondicional durante essa jornada universitária mesmo estando longe de casa.

Agradeço ao meu orientador Fredson Vieira Costa pelo auxílio das atividades desenvolvidas durante o longo período juntos para a finalização deste trabalho.

Agradeço a todos meus professores do curso de sistemas de informação da UNITINS pelo conhecimento imensurável passado ao longo de toda a graduação.

"A nossa maior glória não reside no fato de nunca cairmos, mas sim em levantarmo-nos sempre depois de cada queda." (Oliver Goldsmith)

RESUMO

Ao decorrer dos últimos anos o ESC, Escritório de Soluções Criativas, da Universidade Estadual do Tocantins, esse que tem como propósito ingressar novos acadêmicos à realidade prática do mercado de trabalho, observou a necessidade de uma definição de seus processos de desenvolvimento de software a fim de obter maior qualidade na produção de softwares e controle sobre o fluxo das atividades realizadas. Diante disso, o presente trabalho tem como objetivo a produção de um documento de modelagem BPMN dos processos de desenvolvimento de software do ESC. Esse documento foi desenvolvido analisando as necessidades atuais do ESC e alinhando-as às boas práticas encontradas no mercado de trabalho, dessa forma o documento desenvolvido poderá ser utilizado por discentes e docentes auxiliando-os em projetos atuais e futuros a fim de contribuir para o desenvolvimento de produtos de software com cada vez mais qualidade e confiabilidade, além de auxiliar os acadêmicos integrantes do ESC a vivências e práticas encontradas dentro do mercado de trabalho.

Palavras-chave: Processos; Desenvolvimento; Software; BPMN.

ABSTRACT

Over the last few years, the ESC, Office of Creative Solutions, of the State University of Tocantins, whose purpose is to introduce new academics to the practical reality of the labor market, has observed the need for a definition of its software development processes in order to obtain greater quality in the production of software and control over the flow of activities carried out. In view of this, the present work aims to produce a BPMN modeling document of ESC software development processes. This document was developed by analyzing ESC's current needs and aligning them with the good practices found in the labor market, in this way the developed document can be used by students and professors, assisting them in current and future projects in order to contribute to the development of software products with increasing quality and reliability, in addition to helping academics who are members of the ESC to experience and practices found within the labor market.

Keywords: Processes; Development; Software; BPMN.

LISTA DE TABELAS E QUADROS

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

BPMN	Business Process Model and Notation
ESC	Escritório de Soluções Criativas
NIT	Núcleo de Inovação Tecnológica
UNITINS	Universidade Estadual do Tocantins

SUMÁRIO

1 **INTRODUÇÃO**

O Escritório de Soluções Criativas foi instituído no ano de 2018 na Universidade Estadual do Tocantins, seu objetivo compreende o desenvolvimento de projetos por intermédio da pesquisa aplicada ao setor produtivo, dessa forma, através de demandas estruturadas tanto pelo setor público quanto o privado há aplicação mediante a orientação e coordenação de professores da UNITINS com a visão de ingressar novos acadêmicos de maneira prática às situações baseadas na realidade do dia a dia do setor de produção, no caso específico deste trabalho ao setor de software, oferecendo a possibilidade de experimentação da prática profissional exigida no mercado de trabalho.

Levando em consideração a visão do ESC e a aplicação de seus ideais ao curso de Sistemas de Informação da UNITINS, é possível verificar atualmente a necessidade de uma definição clara dos processos de construção adotados pelas melhores práticas da engenharia de software no desenvolvimento das atividades realizadas por este grupo de trabalho.

Para reforçar essa realidade, de acordo com a visão de Sommerville (2019) a engenharia de software inclui técnicas que apoiam as especificações de um projeto e a evolução dos programas, sendo seu principal objetivo apoiar o desenvolvimento profissional de software, mais do que a programação individual. Dessa forma, para aproximar os envolvidos na construção de softwares propostos pelo ESC às vivências do mercado atual, faz-se imprescindível a utilização de práticas que garantem a qualidade e robustez de um sistema, seja esse computacional ou não.

Dado o cenário de desenvolvimento, é também, de fundamental importância vislumbrar o devido cenário de desenvolvimento, onde a equipe de programadores, testadores e/ou engenheiros de software serão de praxe acadêmicos e por consequência disso necessitam de uma metodologia personalizada e exequível, mas que distinga o menos possível das habituais práticas adotadas em empresas de desenvolvimento presentes no mercado. Logo, para essa definição de processos a serem seguidos, a ferramenta de representação e modelagem dos processos utilizada é a BPMN através da plataforma Bizagi Modeler, notação que auxiliará a visualização não apenas das atividades que devem ser realizadas, mas por quem as realiza e os artefatos de software que devem ser gerados ao decorrer do fluxo de desenvolvimento.

1.1 OBJETIVOS

Para a progressão deste trabalho foram estabelecidos os seguintes objetivos:

geral e específicos.

1.1.1 **OBJETIVO GERAL**

Definir uma metodologia auxiliadora de desenvolvimento de software através de uma modelagem BPMN, para que os envolvidos no processo de construção possam se basear na sequência de processos apresentados e desenvolver um produto de qualidade.

1.1.2 **OBJETIVOS ESPECÍFICOS**

- Analisar distintos procedimentos de desenvolvimento de software de forma técnica e explicativa.
- Distinguir processos de desenvolvimento de software e metodologias que podem ser aplicadas dentro do cenário de desenvolvimento acadêmico proposto, e agrupá-los de forma eficiente.
- Construir uma modelagem sequencial de processos para ser adotado como padrão no desenvolvimento de software em um cenário acadêmico.
- Analisar e apresentar resultados obtidos.

1.2 JUSTIFICATIVA

A adoção de processos e metodologias eficientes no ciclo de desenvolvimento de software tem se mostrado fundamental e imprescindível para a entrega de um produto de qualidade e longevidade no contexto atual, esse cenário proporciona a necessidade e relevância para a elaboração da pesquisa proposta sobre o tema.

Vislumbrando o cenário de desenvolvimento da Unitins e sua necessidade de preparar os acadêmicos associados do ESC ao mercado de trabalho e entregar uma solução de qualidade aos projetos desenvolvidos, a presente pesquisa atua em estabelecer uma definição de processos e metodologias personalizadas ao contexto acadêmico de desenvolvimento de software.

Almeja-se criar um padrão auxiliador para futuros projetos e aplicações dentro do contexto universitário. Dessa forma, trazendo maior inclusão dos acadêmicos de forma gradual e simplificada às práticas utilizadas no mercado de trabalho, além de obter resultados progressivamente mais competentes e satisfatórios.

1.3 RESULTADOS ESPERADOS

Contribuir com o processo de criação de software do ESC/NIT da Universidade Estadual do Tocantins definindo um padrão de criação que atuará desde a análise e definição dos requisitos de sistema até a entrega do produto ao cliente final.

Dessa maneira, evidenciando uma maior produtividade por conta da definição de atividades a serem realizadas e entregues, impactando na maior qualidade do produto final, otimizando o tempo, facilitando correções, atualizações, expansão de escopo e manutenções necessárias.

Com isso, auxiliar os acadêmicos capacitando-os com processos empregados no mercado de trabalho, além de gerar uma maior credibilidade aos softwares desenvolvidos pela Universidade.

Por fim, utilizar esse trabalho como forma de introdução pessoal ao mercado da engenharia de software, estudando e mapeando as estruturas e ferramentas mais utilizadas no contexto profissional atual, e aplicando em uma situação real dentro do cenário de desenvolvimento.

2 **REVISÃO DE LITERATURA**

Neste capítulo será apresentado o aprofundamento das idéias que nortearão o desenvolvimento desta pesquisa como revisão de literatura.

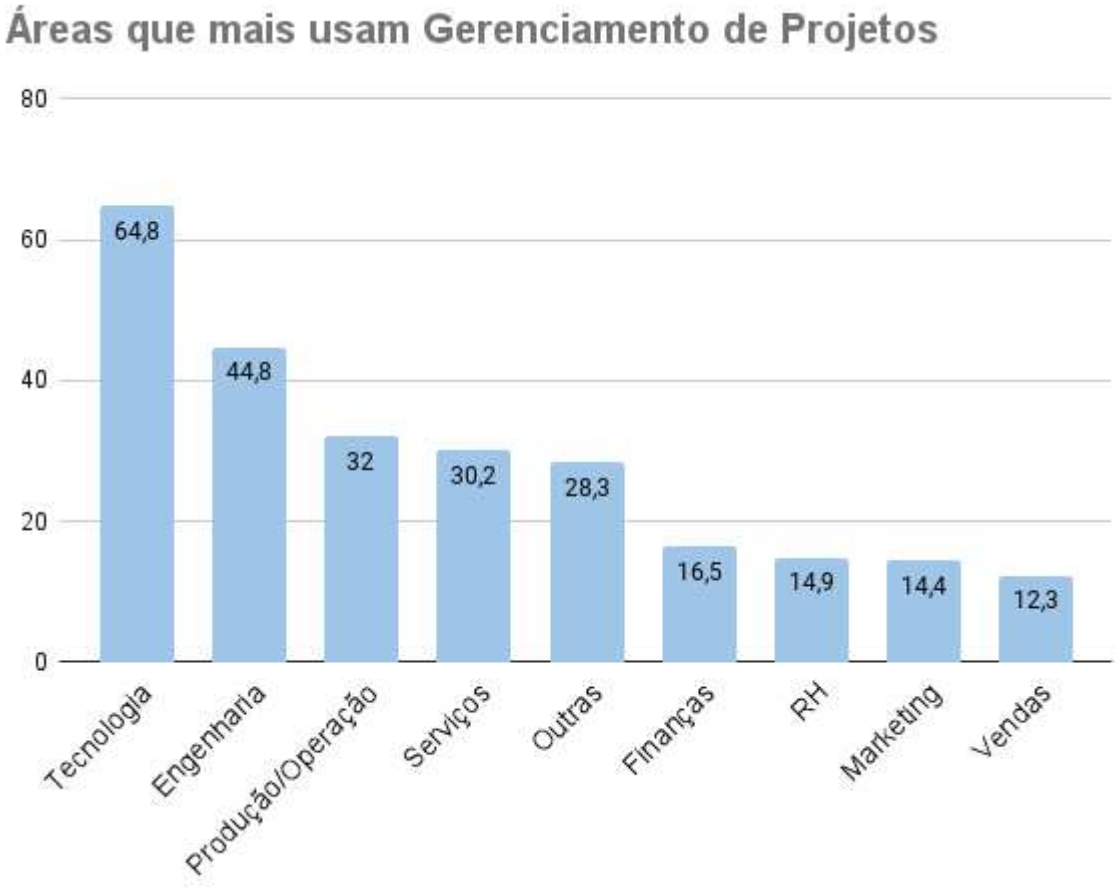
2.1 QUALIDADE DE SOFTWARE

Um dos conceitos imprescindíveis para entender o cenário atual da construção e consolidação de sistemas computacionais, é a qualidade de software. Esse conceito está correlacionado de forma adjacente aos processos de software, esses que serão discutidos ao decorrer do trabalho.

De acordo com Pressman (2016), para classificar a qualidade de um projeto de software é necessário seguir algumas diretrizes, inicialmente o projeto deve implementar claramente todos os requisitos descritos no modelo de requisitos, além de acomodar por completo os requisitos implícitos desejados pelas partes. O projeto também deve se tornar um guia de fácil compreensão para os desenvolvedores e a equipe de testes. Por fim, o projeto de software deve promover uma visão completa do que será desenvolvido.

É nítido que a qualidade de um sistema final está intrinsecamente ligada a uma boa construção e planejamento do software, assim como em outras áreas de construção de qualquer produto, o gerenciamento, a organização e a delineação do escopo são bases que sustentarão a propriedade e a particularidade de um resultado satisfatório. De acordo com o Estudo de Benchmarking em Gerenciamento de Projetos de 2010, feita pelo PMI com 460 instituições, dentre as áreas que mais utilizam uma metodologia de gerenciamento de projeto já pré-definida, a que lidera o topo da lista com um percentual de 64,8%, é a área da tecnologia da informação, reforçando a grande demanda por ciclos de desenvolvimento progressivamente mais planejados e bem executados.

Figura 1 — Realidade do Gerenciamento de Projeto no Mercado



Fonte: Estudo de Benchmarking em Gerenciamento de Projetos Brasil (2010).

Segundo Bartié (2002), apesar da grande evolução ocorrida dentro do ambiente de desenvolvimento de software na virada do século, algumas empresas estão presas a antigos padrões e paradigmas que impedem o amadurecimento dos procedimentos de construção de software. A grande questão que deve ser entendida e visada dentro das empresas e equipes de desenvolvimento é que implantar um processo de garantia de qualidade dentro de um sistema não é opcional no contexto pós século XX, entretanto, uma garantia da sobrevivência dentro da competitividade e altos padrões de exigência encontrados no mercado.

A tabela abaixo apresenta como o nível de complexidade de software dentro das organizações tem apresentado uma nítida evolução desde os primórdios do desenvolvimento.

Tabela 1 — Evolução do Processo de Qualidade e Testes de Software

Características	1960	1980	2000
Tamanho do Software	Pequeno	Médio	Muito Grande
Complexidade do Software	Baixa	Média	Alta
Tamanho da Equipe Desenvolvedora	Pequeno	Médio	Grande
Padrões e Metodologias de Desenvolvimento	Interno	Moderado	Sofisticado
Padrões e Metodologias de Qualidade e Testes	Interno	Emergente	Sofisticado
Organizações de Qualidade e Testes	Bem Poucas	Algumas	Muitas
Reconhecimento da Importância da Qualidade	Pequeno	Algum	Significante
Tamanho da Equipe de Qualidade e Testes	Pequeno	Pequeno	Grande

Fonte: Bartié (2002)

Incorporado ao conceito da iniciação da construção de um sistema e posteriormente no gerenciamento de suas atividades, o conceito de requisitos é fundamental para o andamento e perfazimento de um software, a seguir será discorrido sobre o conceito de requisitos de software e a importância de entender com clareza não apenas seu significado, mas sua relevância no cenário de desenvolvimento.

2.2 REQUISITOS DE SOFTWARE

(MACHADO, 2016, p.10-11) define o que são requisitos da seguinte forma:

> [...] expressam as características e restrições do produto de software do ponto de vista de satisfação das necessidades do usuário e, em geral, independem da tecnologia empregada na construção da solução, sendo a parte mais crítica e propensa a erros no desenvolvimento de software.
> Requisitos são objetivos ou restrições estabelecidas por clientes e usuários do sistema que definem as diversas propriedades do sistema.
> [...] um conjunto de requisitos pode ser definido como uma condição ou capacidade necessária que o software deve possuir para que o usuário possa resolver um problema ou atingir um objetivo ou para atender as necessidades ou restrições da organização ou dos outros componentes do sistema.

Para o esclarecimento da definição de requisito, Pressman (2016) afirma que

a ampla visão de tarefas e técnicas que leva ao entendimento dos requisitos é de fato a engenharia de requisitos. Entendendo o conceito onde é aplicada a engenharia de requisitos, o entendimento da significação do termo é de mais fácil absorção. Para o autor, a engenharia de requisitos atua como uma ponte que liga o projeto à construção, fazendo a indagação de onde seria efetivamente o início dessa ponte. Como resposta é apontado que independe do ponto de partida da construção da ponte o objetivo da engenharia de requisitos é permitir a examinação do contexto do trabalho de software a ser realizado, as especificações nas necessidades que o projeto se propõe a atender, as prioridades que indicam a ordem que o trabalho deve ser concluído, as informações, funções e comportamentos que impactarão no resultado do projeto de software.

Apresentada as definições de requisitos, entende-se que estas são características necessárias que um software deve possuir para satisfazer as necessidades do cliente na entrega de um produto finalizado. A grande questão que trás fundamento para entender a indispensabilidade do conhecimento sobre sua importância é que a partir do momento que estes requisitos não são cumpridos ou são executados de maneira diferente do que o solicitado pelo cliente final, o software já apresenta uma contrariedade ao que foi requerido.

2.2.1 **Elicitação de Requisitos**

Outro conceito fundamental para ser esquadrinhado dentro do universo da qualidade de software é a elicitação dos requisitos. A elicitação é a técnica de derivação dos requisitos do sistema que se origina por intermédio da análise dos sistemas que já existem, envolve procedimentos como conversas com potenciais usuários e compradores, estudo de tarefas, e outras atividades. Essa etapa do processo pode englobar o desenrolar de um ou vários modelos de sistemas e protótipos, que nos auxiliam a compreender o sistema que será criado (SOMMERVILLE, 2019).

A fase da elicitação de requisitos tem uma importância crucial no ciclo de desenvolvimento de software, pois, é o período do planejamento do sistema onde são coletadas as informações que irão nortear o andamento do projeto. Para Pressman (2016), o também chamado levantamento de requisitos abarca a combinação de componentes de resolução de problemas, elaboração, negociação e especificação. Com o intuito de incentivar abordagem colaborativa e voltada a equipes à coleta de requisitos, os colaboradores atuam coletivamente para reconhecer as problemáticas e sugerir novas soluções, além de negociar diferentes abordagens e caracterizar um grupo preliminar de requisitos da solução.

Dessa forma uma boa elicitação de requisitos é benéfica tanto para

diminuição de possíveis retrabalhos futuros, como pela extração das principais funcionalidades almejadas pelo cliente final. Essa fase auxilia aqueles que estão envolvidos no sistema a buscar a compreensão do que será feito focando em algumas dimensões da esfera da problemática trabalhada.

Segundo Kotonya e Sommerville (1998), a fase da elicitação de requisitos se divide em quatro dimensões:

> Entendimento do Domínio: compreender o domínio da aplicação é entender de forma geral onde o sistema será aplicado.
> Entendimento do Problema: os detalhes da especificação do problema do usuário onde o sistema será aplicado devem ser entendidos. Esse entendimento auxiliará na elaboração de uma correta solução.
> Entendimento do Negócio: os sistemas geralmente são criados para contribuir no desenvolvimento de um negócio ou organização. Sendo assim, entender o negócio no qual o sistema será inserido, como as diferentes partes do negócio interagem e afetam uma a outra, é essencial para a sua elaboração.
> Entendimento das Necessidades dos Stakeholders: é preciso entender o processo de trabalho, como o dia a dia das pessoas será afetado pelo sistema. (apud BRITO, 2010, P. 27).

Segundo Rezende (2005), existem alguns fatores que levam à insatisfação dos clientes em relação ao software solicitado, um dos problemas primordiais é o não atendimento ao que foi especificado pelo cliente, ressaltando a necessidade de uma elicitação de requisitos de qualidade, tal ponto é afirmado apresentando uma realidade dentro do cenário de desenvolvimento, que é a precária ou inexistente comunicação entre time de programadores e clientes, ou seja, a partir do momento em que os programadores não tem o contato direto com o usuário solicitante para sanar eventuais dúvidas sobre o funcionamento do sistema, resta seguir com fidelidade as instruções transmitidas através do documento de requisitos.

Em um caso onde há uma elicitação de requisitos feita de forma que não reflete fielmente os interesses dos stakeholders, o projeto está sujeito a mudanças que irão ocorrer durante outras etapas do ciclo de desenvolvimento de software, o que terá impacto em fatores como tempo, dinheiro e presumivelmente na qualidade do produto final.

2.3 MODELAGEM DE SOFTWARE

Para Vazquez e Simões (2016), os requisitos isolados tendem a não ser complexos de analisar, o que dificulta o entendimento destes são quando colocados em interdependência e relacionados a outros requisitos, a partir desse momento o entendimento de como estes funcionarão fica mais difícil de visualizar, a modelagem

trás apoio a partir de uma representação visual e descritiva para auxiliar o entendimento e comunicação.

Ainda segundo os autores, uma coleção de requisitos pode possuir diferentes perspectivas ao qual podem ser analisadas e priorizadas dependendo da necessidade vigente, essas coleções que tem o papel de fornecer um contexto a fim de passar um informação de distintas visões dos requisitos, por isso se justifica a necessidade de diferentes modelos para uma melhor compreensão.

Esses modelos em formato de diagramas que auxiliam o entendimento dos requisitos são amplamente utilizados dentro do ciclo de vida do desenvolvimento de software, alguns dos popularmente mais conhecidos são:

- Diagrama de Classes
- Diagrama de Caso de uso
- Diagrama de Atividades
- Diagrama de Estados
- Diagrama de Domínio
- Diagrama de Banco de dados
- Diagrama de Sequência

Segundo Sommerville (2019), a modelagem de software é comumente utilizada para melhor performance do segmento da engenharia de requisitos, são fundamentais para esclarecer o que o atual sistema é capaz de fazer e melhorar o repasse dos requisitos propostos aos stakeholders.

2.4 IMPLEMENTAÇÃO DE SOFTWARE

De forma simples e em conformidade com o pensamento de Pressman (2016), a implementação de software é a etapa do ciclo de vida de software onde os modelos são traduzidos para código fonte. Ainda segundo o autor, o processo de implementação ou construção de software é aquele que constrói o que foi anteriormente projetado, essa atividade combina a criação de código e já engloba os testes necessários para descobrir erros durante o desenvolvimento do mesmo. Por questão de organização, a ênfase na parte de testes de software será abordada em um subcapítulo posterior.

A visão de Wazlawick (2019) entra em correspondência com a visão de Pressman (2016) a partir do momento que este define a fase de implementação como aquela que o software é construído segundo uma especificação detalhada, apoiando a princípios como a garantia de que o sistema atenda às necessidades dos usuários, o desenvolvimento de versões úteis do sistema, o desenvolvimento de

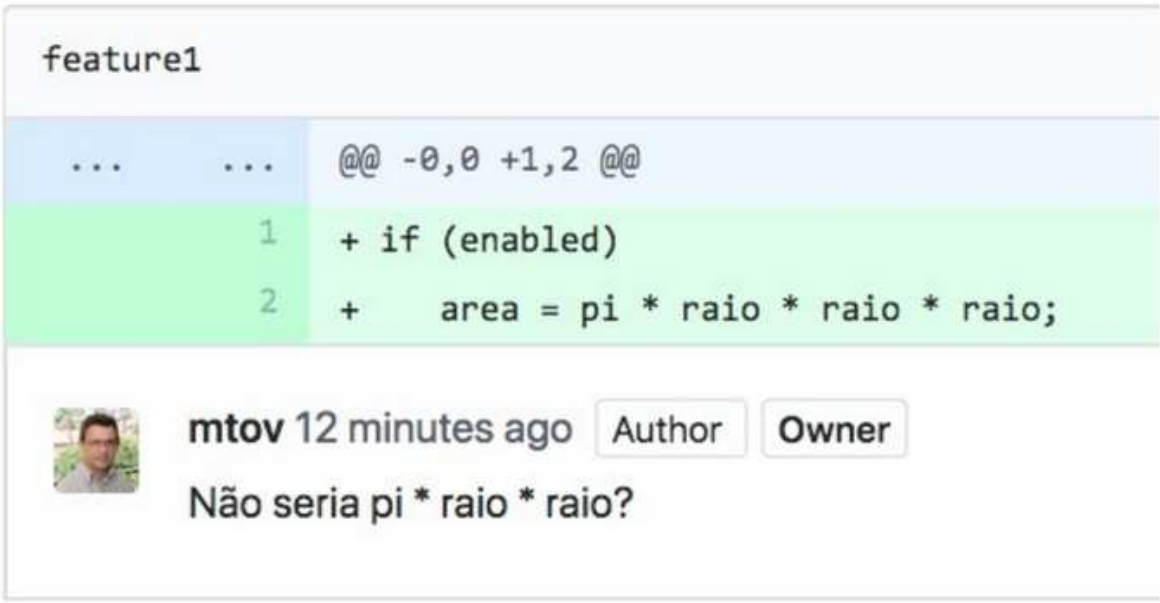
componentes de software e seus devidos testes e o foco na qualidade do produto durante o processo de implementação.

2.4.1 **Code Review**

A revisão de código, ou code review, é a prática de revisar partes de determinado código. Segundo Valente (2020), a revisão de código é uma prática moderna e amplamente utilizada para a garantia da qualidade do código fonte durante a fase de implementação ou testes. A ideia é que o código produzido somente entre em produção após uma inspeção e revisão por outro desenvolvedor do time. O princípio se baseia em que ao colocar um ator diferente ao que fez o código, esse possa ser analisado a fim de detectar bugs, más práticas, legibilidade, manutenibilidade, modularidade e etc. Além de transmitir boas práticas da engenharia de software aos membros da equipe de desenvolvimento.

A figura a seguir representa um exemplo simples da revisão de código na prática utilizando o versionador de código GitHub para a atividade. O código foi submetido ao revisor de código e este detectou um possível bug e informou ao autor do código a problemática.

Figura 2 — Exemplo Code Review

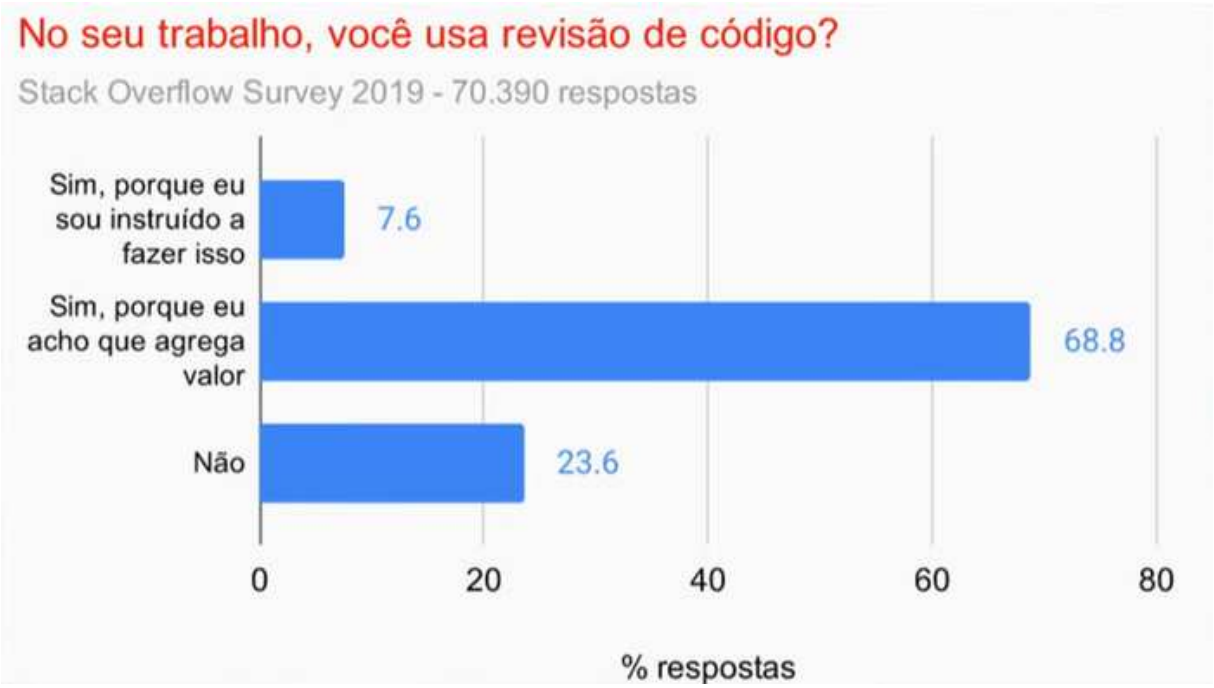


Fonte: Valente (2020).

Dando sequência, a próxima figura apresenta uma pesquisa realizada dentro da plataforma Stack Overflow em 2019, onde o objetivo foi levantar informações sobre o uso da prática de code review dentro do ambiente de trabalho.

Com mais de 70.000 respostas coletadas, a pesquisa apresentou que aproximadamente 70% das pessoas que apresentaram uma resposta usam a prática e acreditam que essa agrega valor ao cotidiano da programação. Por fim, ainda há um tipo de resposta que também é relevante na pesquisa, o uso da prática de revisão de código, porém, com a justificativa que usada por instrução da organização a fim de realizar a atividade, esse agrupamento de resposta chega a quase 8% das respostas totais.

Figura 3 — Uso de Revisão de Código no Trabalho

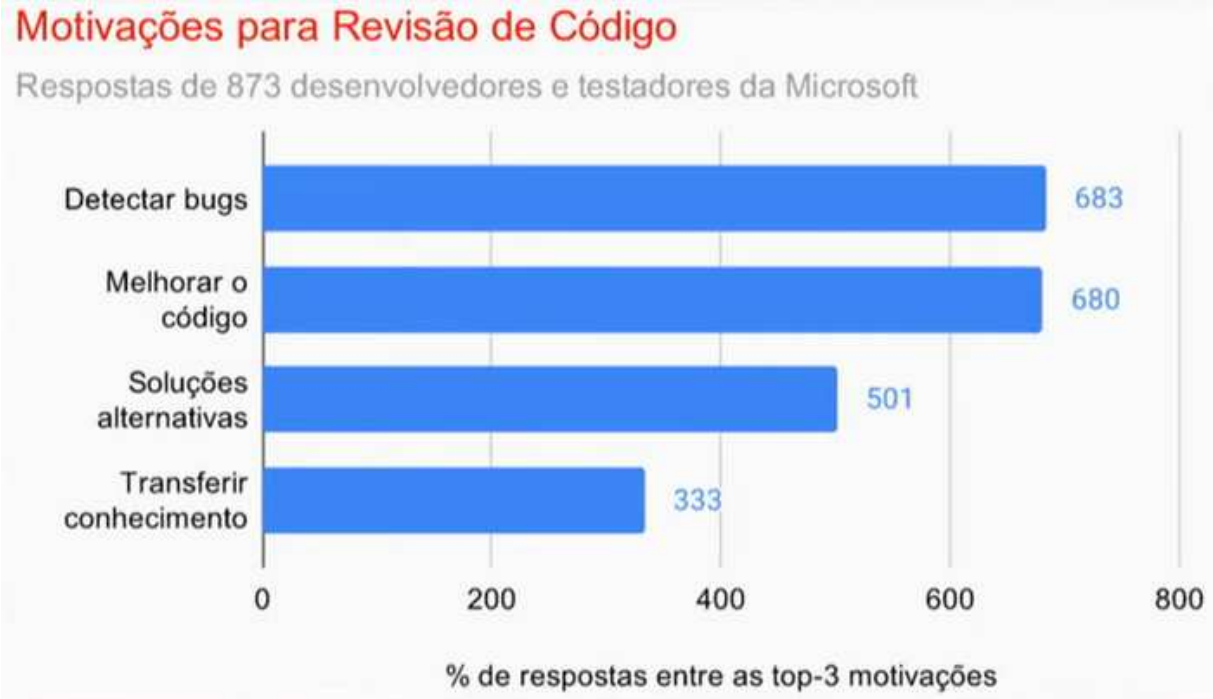


Fonte: Pesquisa Stack Overflow (2019).

Por fim, um outro estudo feito com 873 desenvolvedores e testadores da Microsoft, expõe as motivações que levam os mesmos a utilizar a prática de code review.

É interessante apresentar que as principais respostas coletadas manifestam princípios altamente benéficos a um time de desenvolvimento, entre eles estão: detectar bugs, melhorar o código, encontrar soluções alternativas e transferir conhecimento.

Figura 4 — Motivos para Revisão de Código



Fonte: Estudo de Bacchelli e Bird (2013).

2.5 TESTE DE SOFTWARE

De acordo com Delamaro, Maldonado e Jino (2016), o intuito das atividades relacionadas ao teste de software é executar um programa ou modelo utilizando determinadas entradas e verificar se o comportamento condiz com o esperado. Se a saída não seguiu de acordo com o previsto, pode se dizer que um defeito ou erro foi encontrado no software.

Ainda segundo os autores as atividades relacionadas a testes de software são de grande importância para uma organização pois permitem a detecção de erros, falhas e bugs desde as fases iniciais do ciclo de desenvolvimento. Essas atividades são responsáveis por agregar valor à organização por gerar uma economia significativa de recursos, tempo de manutenção e aumento da confiabilidade e qualidade do produto final.

Segundo Valente (2020), testes de software resumem-se na execução de determinado programa com um conjunto finito de casos e com a finalidade de verificar se o comportamento do sistema ocorreu como esperado.

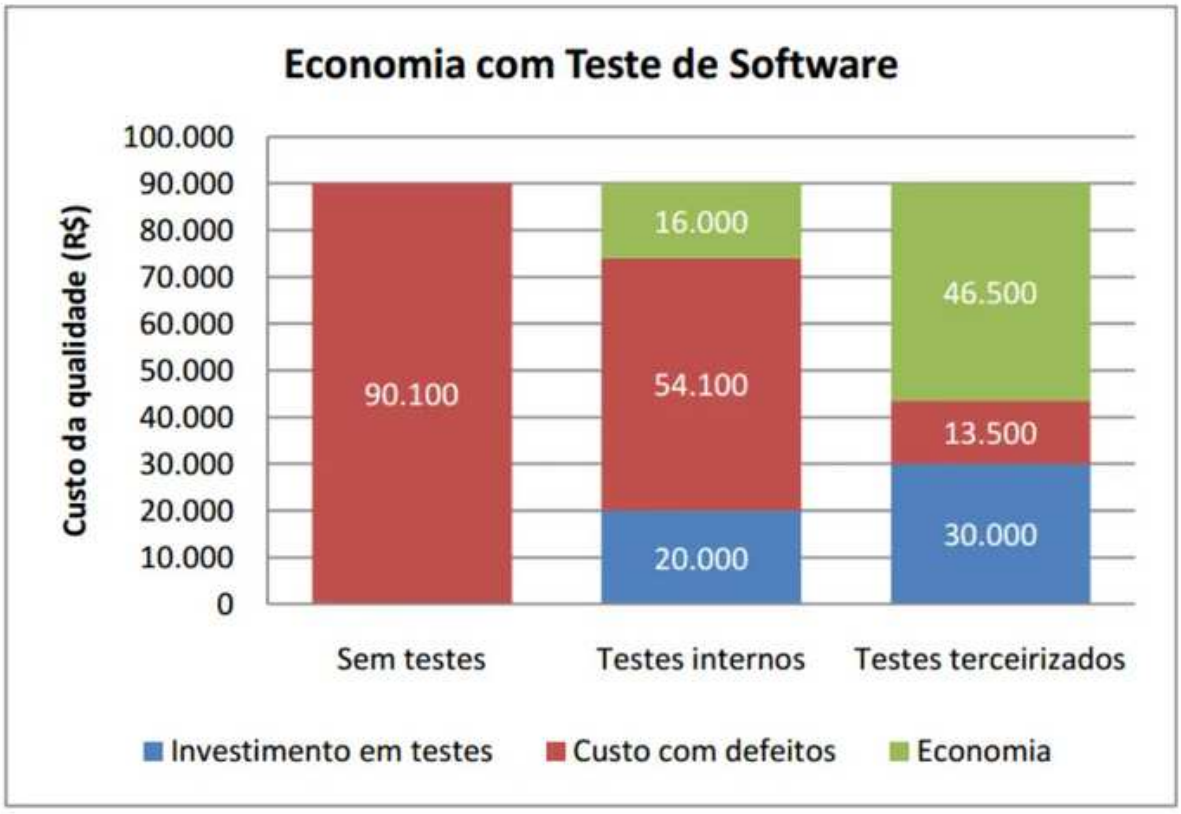
De acordo com o autor, existem diversos tipos de testes, e alguns dos mais utilizados no contexto moderno são:

- Teste Unitário ou de Unidade: Responsável por testar uma pequena unidade do código e verificando sua execução.

- Teste de Integração: Responsável por testar uma unidade de maior granularidade, a fim de favorecer a integração das partes.
- Teste de Performance: Responsável por verificar o desempenho ao ser submetido a uma carga de processamento.
- Teste de Usabilidade: Responsável por verificar a usabilidade da interface do sistema.

A figura a seguir apresenta um estudo do blog One Day Testing, apresentando os benefícios em relação aos recursos economizados ao aplicar um maior investimento na etapa de testes no contexto de uma equipe de software.

Figura 5 — Economia de Recursos com Teste de Software



Fonte: One Day Testing Blog (2015).

Dessa forma, é notório que o investimento na área de teste não traz benefícios apenas ao resultado final do produto em termos de qualidade, entretanto é possível perceber a redução relevante nos custos de desenvolvimento, pois a partir do momento que há um alto foco na atividade de testes, o tempo de retrabalho e manutenção de código é reduzido drasticamente.

2.6 ENTREGA DE SOFTWARE

Wazlawick (2019) diz que o marco mais importante dentro do ciclo de desenvolvimento de um software é a entrega, entretanto se os usuários não conseguirem utilizar o sistema ou se seus dados não forem corretamente importados, pouco essa entrega terá valor. O autor afirma que um projeto de software costuma envolver muito mais pontos fundamentais do que apenas entregar o software ao cliente, existe a necessidade da realização de testes, não apenas os de sistema, mas por exemplo, os de aceitação e validação por parte do cliente. Ainda para o autor, há também a necessidade da importação de dados, treinamento dos usuários, definição de procedimentos operacionais, entre outras práticas e artefatos, por isso, a entrega de software é uma atividade fundamental e imprescindível para o sucesso de um sistema.

De acordo com (HUMBLE e HARLEY, 2004 apud VALENTE, 2020, p. 351), existem alguns princípios importantes a serem seguidos ao realizar a entrega de um software, são eles:

- Crie um processo repetível e confiável para a entrega de software: Sendo considerado o mais importante entre os princípios, seu ideal se baseia que a entrega de software não pode ser vista como uma atividade traumática, entretanto um processo simples e realizado de forma planejada.
- Automatize tudo que possível: A ideia é que todos os passos para a entrega do software sejam o mais automatizados o possível, incluindo atividades como os testes, builds, configuração e ativação dos servidores e rede, carga no banco de dados e etc.
- Mantenha tudo em um sistema de controle de versões: Esse princípio foca no controle do código fonte, documentos, diagramas, arquivo de dados e etc. A fim de que tudo fique salvo caso haja necessidade de pesquisa em um futuro próximo.
- Se um passo causa dor, execute-o com mais frequência e o quanto antes: A ideia é antecipar os problemas antes que esses acumulem-se e se tenham resoluções complexas. Um exemplo é a integração, visto que pode haver casos onde um desenvolvedor passa muito tempo programando isoladamente e deixando um código não revisado e sem integração, isso pode causar muitos prejuízos ao deixar tal atividade para ser realizada perto da entrega do software.
- Concluído significa pronto para entrega: Esse princípio defende que ao desenvolvedor finalizar uma atividade, essa deve estar 100% pronta e

revisada para entrar em produção, isso pois acontecem casos de entregas de desenvolvedores onde um código "pronto" ainda possui pendências simples deixadas pelo desenvolvedor, como por exemplo, a falta de testes.

- Todos são responsáveis pela entrega do software: A equipe deve ser unida como um todo e não trocar informações importantes sobre o projeto apenas na véspera da implantação.

2.7 PROCESSOS DE SOFTWARE X MODELO DE SOFTWARE

Para o seguimento lógico do desenvolvimento da pesquisa, é importante separar a ideia e definição de processos de software e modelos de software, também conhecido como modelos de processos de software. Para isso nessa seção será apresentada a definição de cada um dos termos e como eles são apresentados dentro do ambiente da engenharia de software.

2.7.1 **Processos de Software**

Segundo (SWEBOK, 2004 apud NETO, 2016, p. 115), a qualidade de um software está intrinsecamente ligada aos processos adotados, sendo a qualidade tratada como um processo a ser seguido, esse processo que é empregado em todos os panoramas do projeto. Ele que consegue determinar os responsáveis pelas atividades e os requisitos, suas métricas e saídas, além do feedback. Esse planejamento de garantia da qualidade do software através de processos bem detalhados e explicados envolvem duas características principais:

- Uma definição do produto final ao que diz respeito de seus atributos de qualidade
- Organização e estruturação do processo para a obtenção do produto almejado.

De acordo com KENT (2002), e citado por FONSECA (2021, p. 1) há uma grande pluralidade de propostas de notações, modelos, linguagens e outros tipos existentes para a modelagem do fluxos de processos, cada uma com sua particularidade e distinção sobre sua proposta e visão em cima do que será modelado. Dentro dessas propostas de modelagem de processos, algumas das mais populares existentes no mercado são as notações de processos.

2.7.1.1 Notação de Processos de Software

Notação de processos é uma coleção de símbolos que possuem regras para significação dos mesmos ao serem utilizados em conjunto (ABPMP, 2013, p. 77 apud CGU, 2020, p. 13).

Ainda de acordo com ABPMP (2013), existem muitos padrões de notação de modelagem de processos, entre eles, são usadas normas e convenções bem conhecidas que oferecem algumas vantagens para o entendimento dos fluxos, entre essas vantagens estão:

- Agrupamento de símbolos, linguagens e métodos populares que possam estabelecer um entendimento e comunicação entre as pessoas envolvidas.
- Constância na forma e significação dos modelos e processos.
- Possibilidade de transição entre ferramentas e continuidade do significado.
- Criação de sistemas por intermédio de modelos de processos.

Dessa forma, é notório que as notações de processos de software são utilizadas para uma definição gráfica detalhada do fluxo de atividades que devem ser seguidas, desse modo, o entendimento dos membros da equipe sobre as tarefas que devem ser feitas e por quem devem ser realizadas se torna evidente, evitando assim dúvidas e contrariedades.

Alguns exemplos de notação de processos bem comuns encontrados no mercado atualmente são:

- BPMN
- Fluxogramas tradicionais
- Mapofluxograma

2.7.2 **Modelo de Software**

De acordo com Sommerville (2019) pode-se dizer que o modelo de processo de software é uma visão simplificada dos processos de software, sendo que cada modelo existente fornece uma idéia base de maneira parcial sobre a sequência de fluxos a serem seguidos, ou seja, o modelo apresenta as atividades a serem desenvolvidas em um fluxo com uma sequência lógica, entretanto não entrando no mérito da especificação de como serão desenvolvidos internamente, nem apresentando eventuais processos secundários existentes em cada atividade principal.

Em concordância com o pensamento de Sommerville (2019), Pressman

(2016) evidência os modelos de software como fluxos de tarefas que serão realizadas e identifica sua serventia benéfica dentro do ambiente de trabalho, pois proporciona grande estabilidade, controle e organização para as atividades que se não controladas adequadamente podem tornar-se confusas e desordenadas.

Alguns dos exemplos de modelos de software abordados pelos autores citados:

- Modelo Cascata
- Modelo Incremental
- Engenharia de software orientada a reúso
- Modelo V
- Modelo Evolucionário
- Modelo Espiral
- Modelo Concorrentes

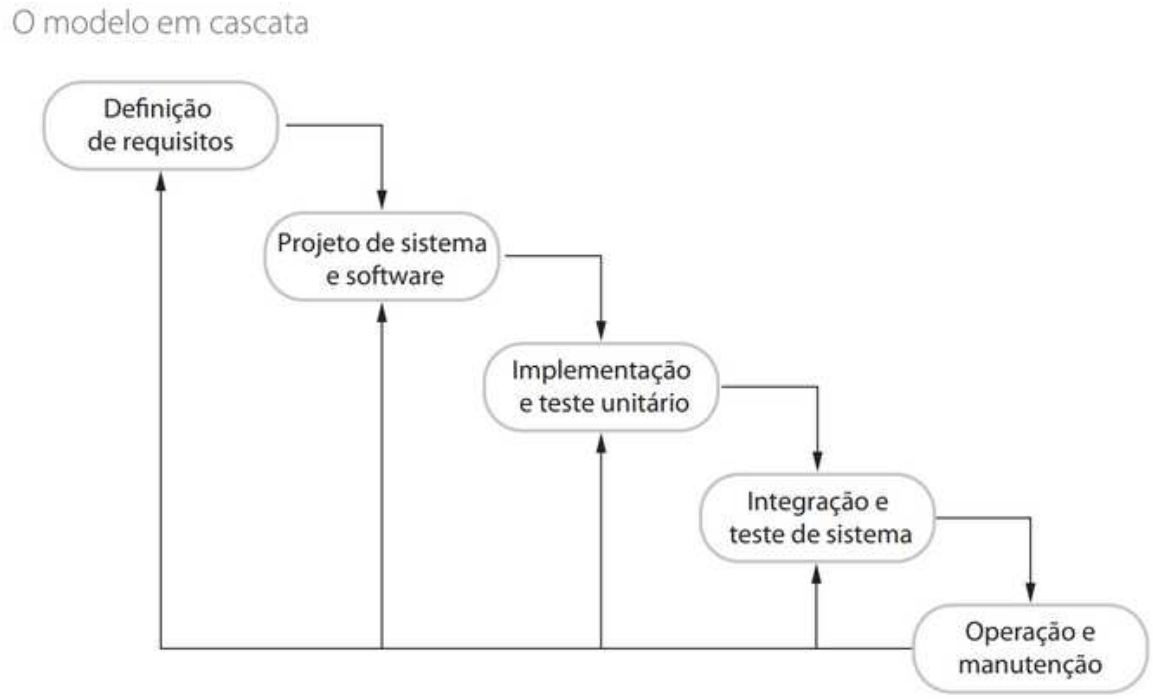
2.8 MODELO EM CASCATA

A seguir será apresentado o modelo cascata, um modelo conhecido por ter oferecido base aos modelos criados posteriormente. Em seguida será apresentado de forma resumida a diferença entre alguns dos modelos mais populares do mercado e da literatura.

Conhecido como o pioneiro dentre os modelos clássicos e também conhecido como ciclo de vida clássico, é uma metodologia de desenvolvimento de software marcada pelas suas etapas de progresso bem estabelecidas, esse modelo cascata foi fundamentado em processos mais comuns da engenharia de sistemas (ROYCE, 1970 apud STANKIEWICZ, 2017, p. 10).

Em concordância com o pensamento de Wazlawick (2019), o modelo cascata apresenta a idealização de um desenvolvimento de software que ocorre em fases bem definidas e não tem como resultado apenas estruturas de código, mas também documentos e registros que irão auxiliar a visualizar o sistema de maneira mais abstrata do que apenas linhas de código.

Figura 6 — Fluxograma do Modelo Cascata



Fonte: Sommerville (2011)

De acordo com Sommerville (2019), os estágios fundamentais do modelo cascata têm impacto direto nas atividades de desenvolvimento.

Tabela 2 — Principais estágios do Modelo Cascata (continua)

1º Etapa	Análise e definição de requisitos.	Estabelecimento das restrições, serviços e objetivos do sistema coletados através da consulta com os usuários. Posteriormente, são determinados os detalhes e especificidades do sistema.
2º Etapa	Projeto de sistema e software	Alocação dos requisitos de sistemas por meio de uma arquitetura geral do sistema. O plano de software compreende a identificação e descrição das abstrações principais do sistema e seus relacionamentos.
3º Etapa	Implementação e teste unitário	Desenvolvimento do software como um conjunto de programas ou unidades de programa. O teste unitário abrange a garantia de que cada uma unidade siga sua particularização como já predefinida.
4º Etapa	Integração e teste de sistema	Realização da integração e testes do(s) programa(s) para o acoplamento no sistema completo, essa etapa tem a finalidade de certificar que os requisitos de software

Tabela 2 — Principais estágios do Modelo Cascata (conclusão)

		foram atendidos. Só após a etapa de teste o sistema é entregue ao cliente.
5º Etapa	Operação e manutenção	Após a entrega ao cliente, o sistema é instalado e posto em uso. Essa etapa serve para manutenção de erros e eventuais melhorias que podem ser descobertas através de novos requisitos. Essa é geralmente, mas não sempre, a fase mais longa do ciclo de vida de um software.

Fonte: Sommerville (2019)

Ainda em conformidade com a linha de raciocínio de Sommerville (2019), o modelo cascata por apresentar sua forte linearidade tem alguns problemas agravantes dentro do cenário de desenvolvimento. Por conta dos estágios serem altamente conectados, um estágio não pode ser iniciado sem a finalização do anterior, dessa forma, criando uma dependência de etapas a serem concluídas. Ao analisar de forma prática dentro do cotidiano de desenvolvimento os estágios se sobrepõem o tempo todo, assim nutrindo uns aos outros com informações e possíveis mudanças, pois durante o projeto são identificados erros e problemas com requisitos já elicitados. A dinâmica do processo de software não é um procedimento linear, entretanto, envolve uma política de feedback de um ciclo para outro, dessa forma é imprescindível ter o conhecimento que os documentos de software e até mesmo seus requisitos podem ser modificados futuramente.

Para Wazlawick (2019), o modelo cascata em sua forma mais simples é impraticável por sua inflexibilidade, para resolver isso foram criadas variações desse modelo ao decorrer dos anos que possibilitaram menor rigidez nos fluxos e maiores respostas aos feedbacks. Alguns desses modelos serão apresentados na tabela a seguir.

Tabela 3 — Modelos de Software (continua)

Modelos	Descrição
Modelo Espiral	O modelo espiral tem foco em suas versões evolucionárias, de forma a gerar versões progressivamente mais completas do sistema. Esse modelo é composto por uma série de atividades metodológicas a serem realizadas até finalizar determinada versão e essa ser entregue aos stakeholders. (Pressman 2016)
Engenharia de Software Orientada a Reúso	Essa abordagem é utilizada quando há um número significativo de componentes reutilizáveis, no qual o processo de desenvolvimento do sistema tem foco em integrar os diferentes componentes já desenvolvidos

Tabela 3 — Modelos de Software (conclusão)

Modelos	Descrição
	ao criar um sistema do absoluto zero. (Sommerville 2019)
Modelo V	Variação do modelo cascata onde há uma garantia da qualidade associada a ações de comunicação, modelagem e construções iniciais. A partir do momento em que a equipe desce em direção aos processos do lado esquerdo do V, há então uma refinação dos requisitos básicos da solução em representações progressivamente mais detalhadas e técnicas do projeto. Ao ser gerado o código, a equipe avança para o lado direito do V, onde é realizado os testes que garantam a qualidade e validam cada um dos modelos gerados, conforme a equipe desce em direção aos processos do lado esquerdo do V. (Pressman 2016)
Modelo Incremental	O diferencial do modelo incremental é o feedback constante ao decorrer do desenvolvimento. Nesse modelo há diferentes versões, no caso a inicial, intermediária e final, onde dentro de cada uma delas há a especificação, o desenvolvimento e a validação, essas atividades ocorrem de maneira intercalada. Com isso o cliente se encontra próximo a equipe de desenvolvimento, e em cada versão que é apresentada ao cliente, é então feito o feedback dessa versão e então a equipe já pode começar a desenvolver a próxima atentando as observações do cliente até alcançar o sistema almejado. (Sommerville 2019)

Fonte: Autor (2022).

3 **METODOLOGIA**

METODOLOGIA

Neste capítulo será apresentado o roteiro metodológico que foi utilizado para o desenvolvimento desta pesquisa e sua aplicação.

Desta maneira, a estrutura deste capítulo pode ser dividida em duas partes principais a serem apresentadas, inicialmente será delimitada a metodologia de desenvolvimento da pesquisa e posteriormente o desenvolvimento do fluxo de processos.

3.1 CARACTERIZAÇÃO DA METODOLOGIA

Esta seção tem por objetivo exibir a caracterização da metodologia adotada para o andamento do projeto diante aos variados tipos e modelos que estão presentes dentro do universo da pesquisa científica.

3.1.1 **Classificação da Pesquisa**

Compreendendo os objetivos de pesquisa existentes, este trabalho se enquadra dentro da pesquisa científica aplicada, visto que tem por objetivo geral definir os processos de desenvolvimento de software do ESC/NIT da UNITINS. A natureza científica deste trabalho se delimita ao tipo aplicado, pois esse modelo de acordo com Gil (2008, p. 32) é utilizado para “Pesquisas voltadas à aquisição de conhecimentos com vistas à aplicação numa situação específica”.

Mediante aos objetivos existentes, esta pesquisa se classifica como exploratória, uma vez que tem por finalidade obter maior compreensão do cotidiano da equipe universitária de desenvolvimento de software e estudar quais ferramentas, metodologias e processos se qualificam para o melhor aperfeiçoamento das atividades prestadas diariamente, que em concordância com o próprio Gil (2008), a pesquisa exploratória tem por objetivo promover maior familiaridade com a problemática para assim torná-la mais clara e construir hipóteses.

A abordagem empregada para a progressão da pesquisa é a qualitativa, pois segundo MINAYO (2008), e citado por GUERRA (2014, p.12) o grande objetivo da pesquisa qualitativa é a objetivação, onde ao decorrer da investigação científica é preciso entender a complexidade do objeto estudado, rever criticamente as teorias que cercam o tema, estabelecer conceitos e teorias relevantes, usar técnicas de coleta de dados adequadas e, por fim, analisar todo o material de maneira focada, específica e contextualizada.

Por fim, as metodologias referente aos procedimentos adotados são a pesquisa bibliográfica, sendo essa de fundamental importância para o levantamento de informações sobre a temática, e a pesquisa participante, visto que o objetivo do estudo não se refere a uma análise da realidade de desenvolvimento de software de uma empresa, organização ou contexto acadêmico externo, entretanto ao escritório de soluções inteligentes do curso de sistemas de informação da Unitins. A adoção dessa modalidade de pesquisa se justifica por conta do procedimento de pesquisa participante ter a finalidade de ajudar a população envolvida a identificar seus próprios problemas, a fim de realizar a análise crítica destes e a buscar as soluções adequadas, segundo o pensamento de (LE BOTERF, 1984).

3.2 DESENVOLVIMENTO DO FLUXO DE PROCESSOS

Esta seção busca apresentar o desenvolvimento do produto almejado e as etapas trabalhadas até a entrega do produto final.

3.2.1 **Delimitação do trabalho**

Nesta subseção, serão apresentadas algumas delimitações para o melhor entendimento e da atuação da pesquisa.

Inicialmente é imprescindível apresentar que o NIT (Núcleo de Inovação Tecnológica), esse que atua diretamente gerindo e direcionando as atividades realizadas no ESC, tem atuação em diversas áreas e projetos relacionados a universidade, não concentrando-se apenas no universo da codificação de software. Sabendo disso, para delimitar o espaço de atuação desse trabalho, é importante ressaltar que para o desenvolvimento deste, o foco principal é direcionado também apenas às atividades de software do ESC contempladas dentro do ciclo de vida da engenharia de software e suas práticas. Atividades relacionadas à gestão de projeto e métodos ágeis não são o foco do trabalho e não serão abordados durante o desenvolvimento das atividades deste trabalho.

3.2.2 **Escolha do Modelo de Software**

Dentre as etapas de desenvolvimento do trabalho, uma das que exigiu maior estudo e embasamento bibliográfico foi a etapa de definição de um modelo de software para ser utilizado como padrão dentro da UNITINS.

Wazlawick (2019) apresenta que não é simples conceituar e aplicar a engenharia de software em uma equipe, porém, é necessário. Para o autor, os

processos da engenharia de software são diferentes em cada caso e depende do tipo de software que será desenvolvido. É preciso da mesma forma, ter ciência que dependendo do nível de conhecimento dos requisitos ou estabilidade destes, deve-se optar por outros ciclos de vida e desenvolvimento, o qual será o mais apropriado ao tipo de cenário de construção que será encontrado.

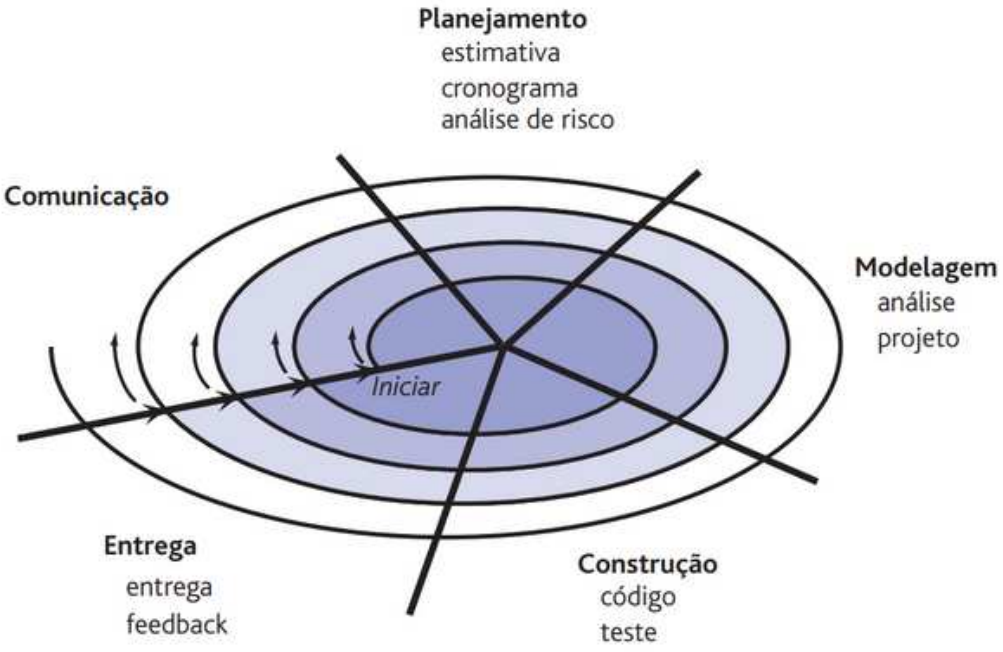
Logo, não é possível dizer que existe o melhor modelo de software definitivo para o desenvolvimento de todos os projetos dentro do ESC. Entretanto, em alinhamento com a orientação do professor orientador deste trabalho e também coordenador dos projetos desenvolvidos no ESC, existe um direcionamento para qual modelo supre a demanda do mesmo atualmente em relação a mesclagem de uma abordagem preditiva com a adaptativa.

Foi verificada a demanda de um modelo híbrido pela necessidade da combinação de aspectos da abordagem preditiva, essa que tem grande foco em processos bem estabelecidos e forte controle sobre o projeto, com a associação à adaptativa, metodologia voltada ao ágil e com abertura a mudanças, pois dessa forma o ESC não estaria preso a modelos datados e ultrapassados como o cascata, e poderia ser adaptar a demandas do mercado como abordagens iterativas e modernas.

Vislumbrando essa necessidade, o modelo Espiral foi o escolhido como aquele que melhor se encaixa nesse panorama por conta de ser um modelo moderno, de abordagem híbrida, e em concordância com Pressman (2016) com fases que podem ser definidas e personalizadas pelo engenheiro de software.

A seguir será apresentado um exemplo de representação do modelo espiral.

Figura 7 — Modelo Espiral



Fonte: Pressman (2016).

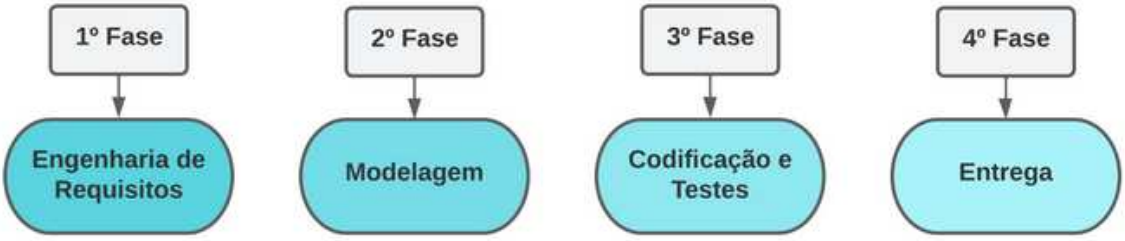
O proveitoso desse modelo é a flexibilidade dada aos projetos, em caso de projetos projetos menores, pode haver a necessidade de poucos ciclos de entrega do que foi feito até a entrega do produto final, já em casos de projetos maiores, é facilmente possível realizar ciclos de desenvolvimento com abordagens cada vez mais ágeis, como uso do SCRUM por exemplo e backlogs bem definidos.

Por conta do escopo do presente projeto, não será realizado uma definição padrão de todas as fases que devem ser realizadas dentro do modelo espiral do ESC, pois o modelo espiral pode conter aspectos presentes na gestão de projeto, esses que como citado na seção 3.2.1 não serão embarcados nesse projeto, ficando então a cargo do engenheiro de software responsável envolver as etapas que achar pertinente, como por exemplo análise de riscos, cronogramas e etc. Entretanto na seção a seguir, serão apresentadas as fases da engenharia de software que serão utilizadas dentro do modelo.

3.2.3 **Fases de Software**

Sob uma ótica macro, foram divididas as etapas fundamentais das fases de software que foram modeladas em quatro partes, são elas: Engenharia de Requisitos, Modelagem, Codificação e Testes, e Entrega.

Figura 8 — Fases de Software



Fonte: Autor (2022).

A divisão foi realizada dessa forma vislumbrando a síntese e melhor visualização dos processos, alinhado às melhores práticas do mercado e a literatura renomada, como por exemplo as obras de Pressman (2016), Sommerville (2019), Wazlawick (2019), Valente (2020), entre outros.

Um ponto de grande importância que foi mesclado para melhor rotina de trabalho dentro do ESC, foram as etapas de codificação e testes. Essa fase de número três tem alta relevância dentro do planejamento de processos do desenvolvimento de software dentro do ESC por conta de práticas que irão contribuir para maior qualidade no desenvolvimento de softwares, são elas os testes unitários antes da disponibilização do código, prática de code review e demais testes que serão planejados e impostos pelo analista de testes. Essa e as demais fases serão melhor detalhadas ao decorrer do trabalho.

3.2.4 **Escolha da modelagem**

A notação escolhida para a modelagem das fases de software definidas durante o trabalho foi a BPMN (Business Process Model and Notation). A escolha se justifica por dois simples pontos, primeiramente esta é uma notação geralmente utilizada no curso de sistemas de informação da universidade, usualmente nas matérias voltadas à gestão, e em segundo lugar, é uma linguagem rica de ferramentas se comparado a simples fluxogramas, ao mesmo tempo sem muitas dificuldades de compreensão.

3.2.4.1 BPMN

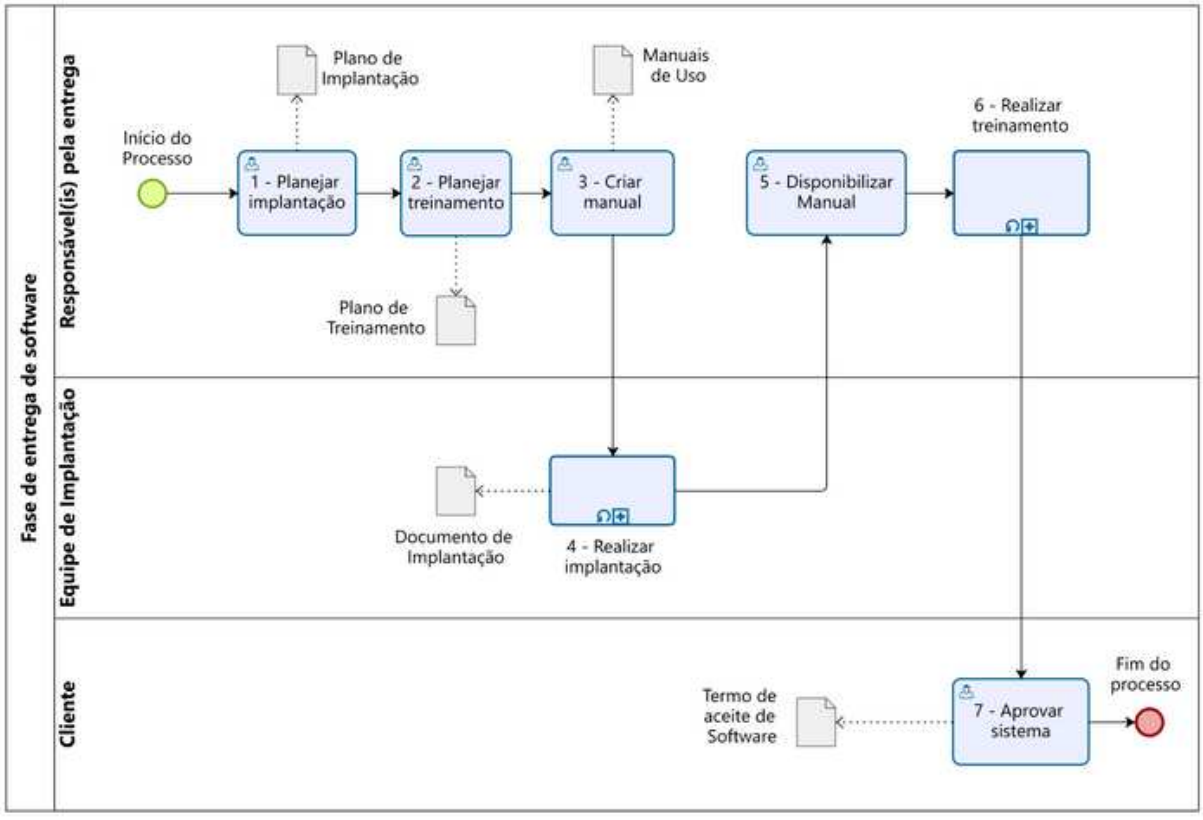
Para CHINOSI e TROMBETTA (2002), ainda citado por FONSECA (2021, p. 1) o BPMN, cuja tradução significa Modelo e Notação de Processos de Negócios, é

uma notação de modelagem para processos com foco na área de negócios da instituição, sendo que essa notação pode ser especializada para os processos de software que serão modelados.

O BPMN não se limita a definir uma notação de processos apenas para a área de desenvolvimento de software, entretanto, à quaisquer processos de negócios e gerenciamento de outras áreas que possam ser agrupados por uma linguagem já preestabelecida e proporcione uma comunicação inteligível à aqueles que irão participar das etapas de atividades. O grande motivo que faz o BPMN ser amplamente utilizado para modelagem dos processos de desenvolvimento de software é que de acordo com FABRIS (2014), essa notação possibilita a estruturação das tarefas para uma linguagem de execução de sistemas, gerando um meio de visualização padrão dos processos de negócio, estabelecendo uma otimizada linguagem de execução.

A figura a seguir apresenta um exemplo do funcionamento prático do BPMN na menor fase modelada neste trabalho, a fase de entrega de software. Para contexto, após o sistema já finalizado essa fase tem o objetivo de implantar o sistema no ambiente do cliente e finalizar o ciclo de desenvolvimento do software.

Figura 9 — Exemplo de um Processo Mapeado Utilizando BPMN



Fonte: Autor (2022).

Além do seguimento lógico dos processos até o ponto de finalização da processo principal, que é a aprovação por parte do cliente, há a divisão organizada das atividades em "raias", que tem a função de dividir os atores e suas devidas atividades, no caso apresentado são os Responsável(is) pela entrega, a Equipe de Implantação e o Cliente, e a "piscina", que é o processo geral almejado, nesse caso é a ação de entregar o software ao cliente.

A imagem a seguir apresenta a organização e identificação da Piscina e das Raias.

Figura 10 — BPMN - Organização das Atividades em Categorias Visuais

Fonte: Autor (2022).

FABRIS (2014) divide o conceito de Piscinas e Raias da seguinte forma:

- Piscina: Retrata um participante em um processo, operando como um container gráfico com o intuito de dividir um agrupamento de atividades de outras piscinas.
- Raias: Retrata uma subpartição pertencente, de forma a serem utilizadas para estruturar, organizar e categorizar as tarefas.

Segundo CERQUEIRA (2017), a principal finalidade do BPMN é fornecer uma notação padrão para ser utilizada na modelagem de processos, de forma a superar as deficiências e limitações de outras notações de modelagem encontradas no

mercado. Para o autor, o BPMN continuará sendo a opção de modelagem mais adotada pelo mercado por ter um forte padrão, robustez e consistência, além da crescente adoção desse padrão pelos fornecedores de softwares.

3.2.4.2 Ferramentas utilizadas para modelagem

Para realizar a modelagem das fases utilizando a notação BPMN foram utilizadas duas ferramentas: Bonita Studio e Bizagi Modeler, ambas ferramentas utilizadas para construção de notações BPMN.

Inicialmente foi realizada toda a modelagem das fases com o Bonita Studio, ferramenta escolhida por conta da familiaridade visto que é uma ferramenta geralmente utilizada no curso de sistemas de informação da universidade.

Entretanto, com o percorrer do trabalho foi observado a limitação da ferramenta Bonita Studio por conta da necessidade de modelar um elemento que não possuía dentro das opções da ferramenta, o elemento em questão é o “Artefato”, ou também chamado de objeto de dados dependendo da condução do projeto.

Visto essa limitação, além da pesquisa na documentação oficial da ferramenta e em fóruns online, ambos apontando a inexistência do elemento, foi enviado então no e-mail empresarial da empresa o questionamento sobre a existência do elemento “Artefato”, até o momento da entrega do presente trabalho a empresa não ofereceu resposta.

Por conta dessa limitação, a modelagem foi refeita do zero, dessa vez na ferramenta Bizagi Modeler, essa por sua vez, oferecendo a possibilidade da modelagem do elemento artefato, esse que se mostra essencial para o seguimento lógico da modelagem dos processos. Ao decorrer deste trabalho será apresentado a importância desse elemento e como ele impacta diretamente na garantia da qualidade dos processos desenvolvidos.

Na seção do APÊNDICE B — Guia de Elementos BPMN, será possível visualizar um guia de auxílio para entendimento da notação BPMN.

3.2.5 **Apresentação do Produto**

O planejamento de apresentação do produto, no caso a modelagem dos processos, sofreu diversas mudanças ao decorrer do trabalho, pois com as reuniões de orientação com os professores do ESC, foi visto a necessidade de colocar alguns pontos fundamentais dentro do documento da modelagem que oferecesse uma visão mais detalhada das atividades necessárias de cada fase. Ao final do trabalho, a apresentação do documento inclui:

- Modelagem BPMN.
- Descrição de cada ator envolvido na fase.
- Descrição de cada processo presente na fase.
- Tabela de Artefatos.
- Descrição dos Artefatos.
- Pontos Fundamentais e Diretrizes de cada Artefato.

3.2.5.1 Modelagem BPMN

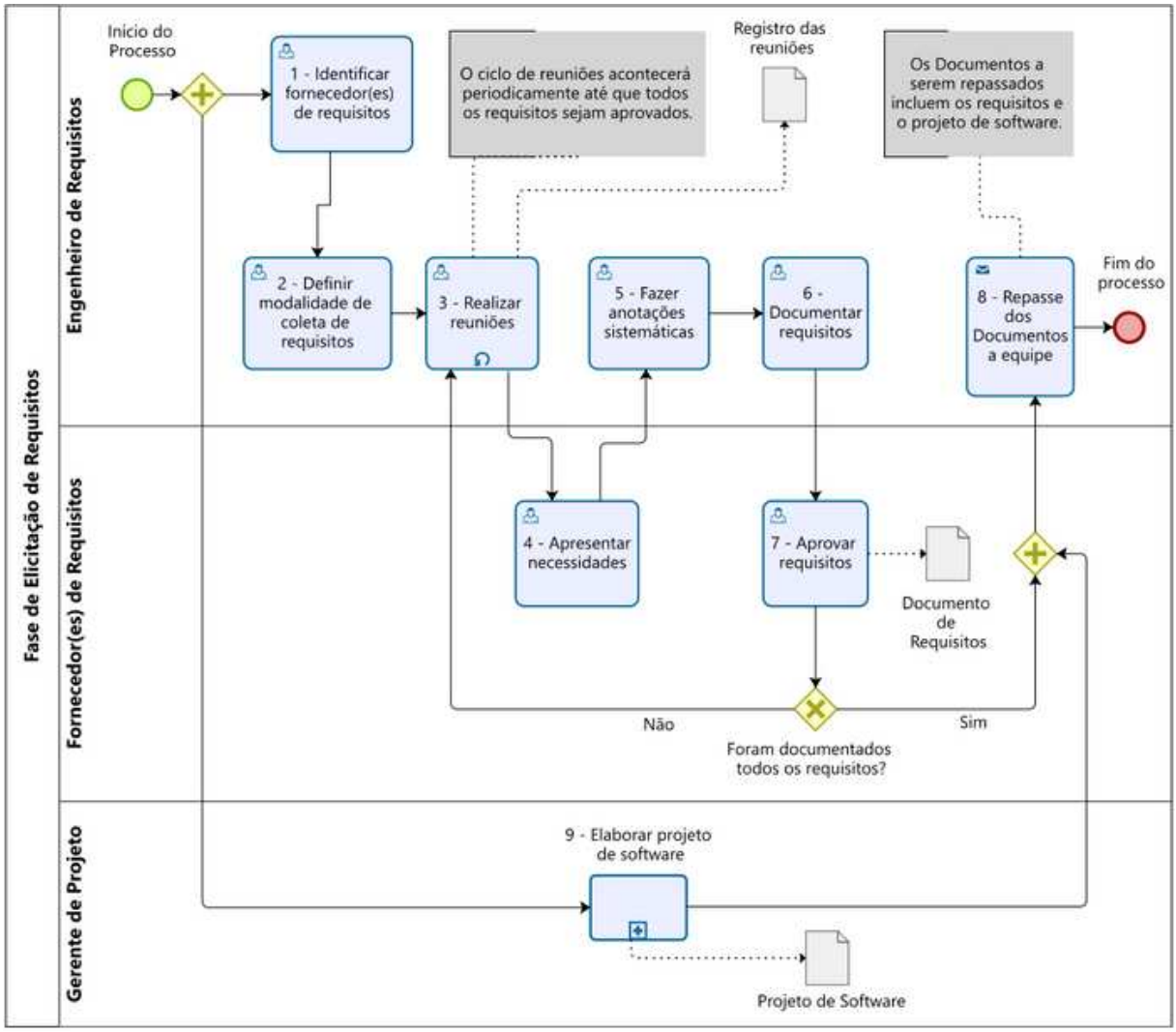
A modelagem foi feita seguindo as quatro fases de software definidas na seção 3.2.3. Todas as fases foram construídas focando na simplicidade e clareza das atividades, focando nos processos primordiais e nos artefatos que necessitam ser gerados durante as atividades.

3.2.5.1.1 *Primeira Fase*

A primeira fase, Elicitação de Requisitos, tem o foco principal nos processos relacionados à coleta de requisitos com os fornecedores de requisitos e a documentação e aprovação dos mesmos.

Essa fase ainda tem outros processos que irão gerar artefatos importantes para o software, como exemplo o registro das reuniões, e o projeto de software.

Figura 11 — Fase de Elicitação de Requisitos

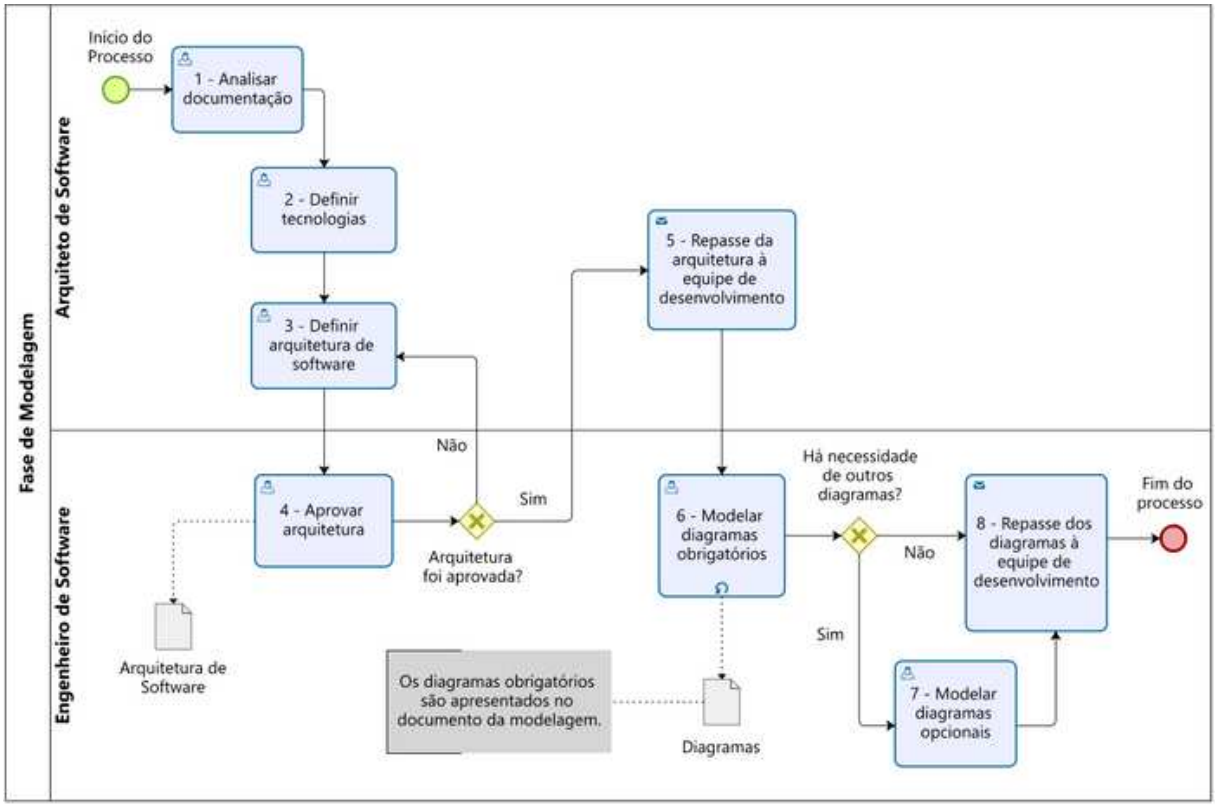


Fonte: Autor (2022).

3.2.5.1.2 *Segunda Fase*

A segunda fase, Modelagem, dá atenção aos processos relacionados à definição da arquitetura e modelagem dos diagramas, além da aprovação destes. Essa fase, porém, não se limita a apenas isso, pois é nela que a definição das tecnologias são feitas e documentos fundamentais sobre o sistema são repassados para a equipe de desenvolvimento.

Figura 12 — Fase de Modelagem



Fonte: Autor (2022).

3.2.5.1.3 *Terceira Fase*

A terceira fase, Implementação e Testes, tem foco nas atividades relacionadas à codificação do sistema e os testes de software, essa fase pontua a atenção à elementos que garantam a qualidade da implementação do software. Inicialmente, após a atividade de desenvolvimento de fato, onde é gerado o código fonte, é realizado a primeira etapa de testes, essa que é feita pelo próprio desenvolvedor, gerando os testes unitários, dessa forma problemas simples já podem ser identificados e solucionados antes do código ser passado adiante e ter que voltar em determinado momento, causando retrabalho e comprometendo tempo da equipe.

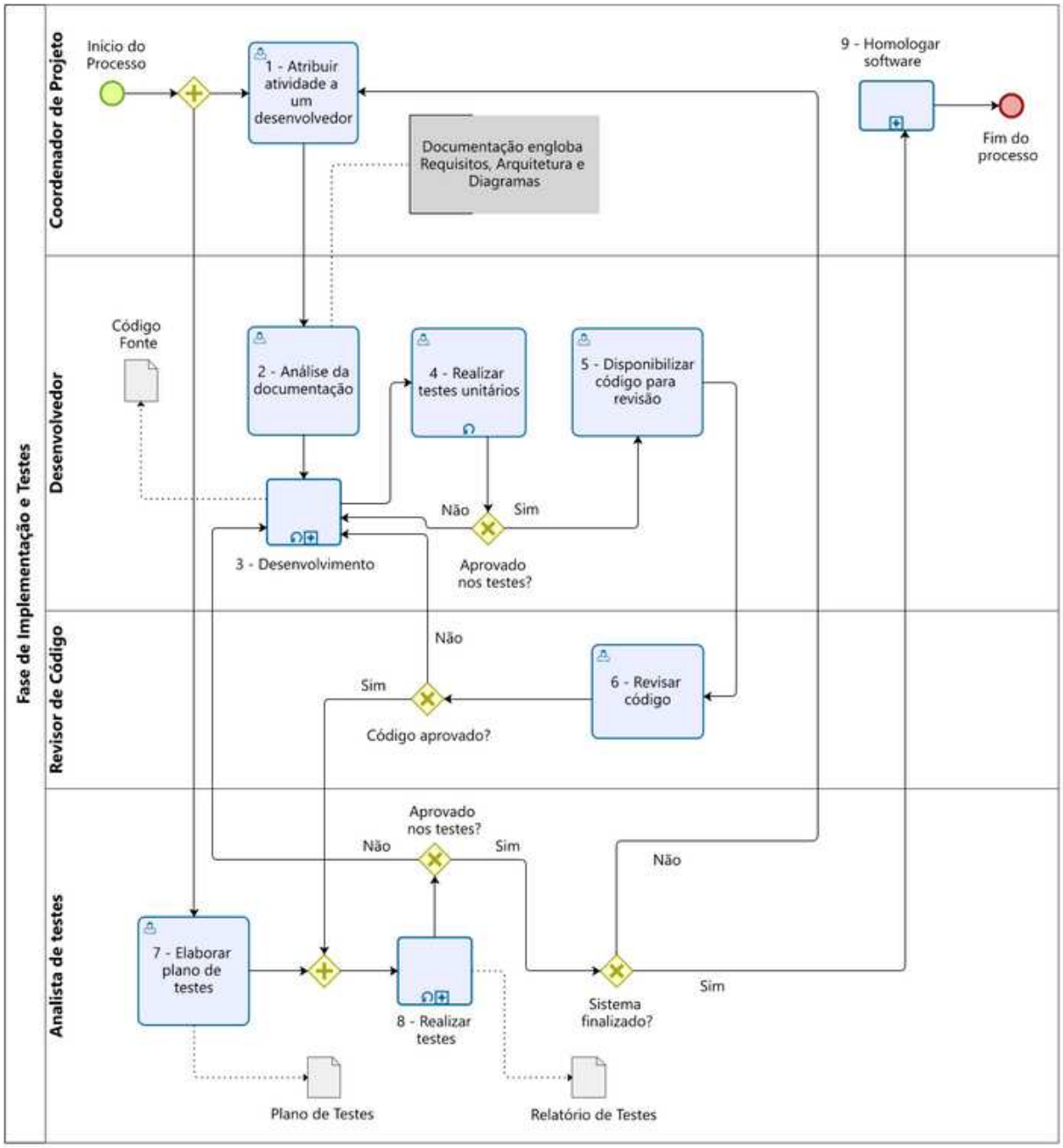
Após a realização dos testes unitários e já enviado o código para revisão, é então iniciada outra etapa de garantia da qualidade do código, o code review. Outro desenvolvedor analisa o código fonte gerado e repassa o checklist de observações em relação ao código analisado, este checklist é feito e disponibilizado pelo analista de testes no plano de testes, esse que será falado com mais detalhes no documento da modelagem. O uso do checklist é fundamental para elencar pontos que necessitam de atenção durante o processo de revisão.

Após o processo de code review, existe mais um processo da garantia da qualidade do software, os testes finais. Esses testes são os definidos pelo analista

de testes no plano de testes, dessa vez menos superficiais que os testes unitários e mais efetivos também, oferecendo mais uma camada de proteção ao produto final.

Além das atividades de codificação e testagem, existem atividades de atribuição de tarefas e homologação de software, essas que são mais detalhadas no documento da modelagem.

Figura 13 — Fase de Implementação e Testes



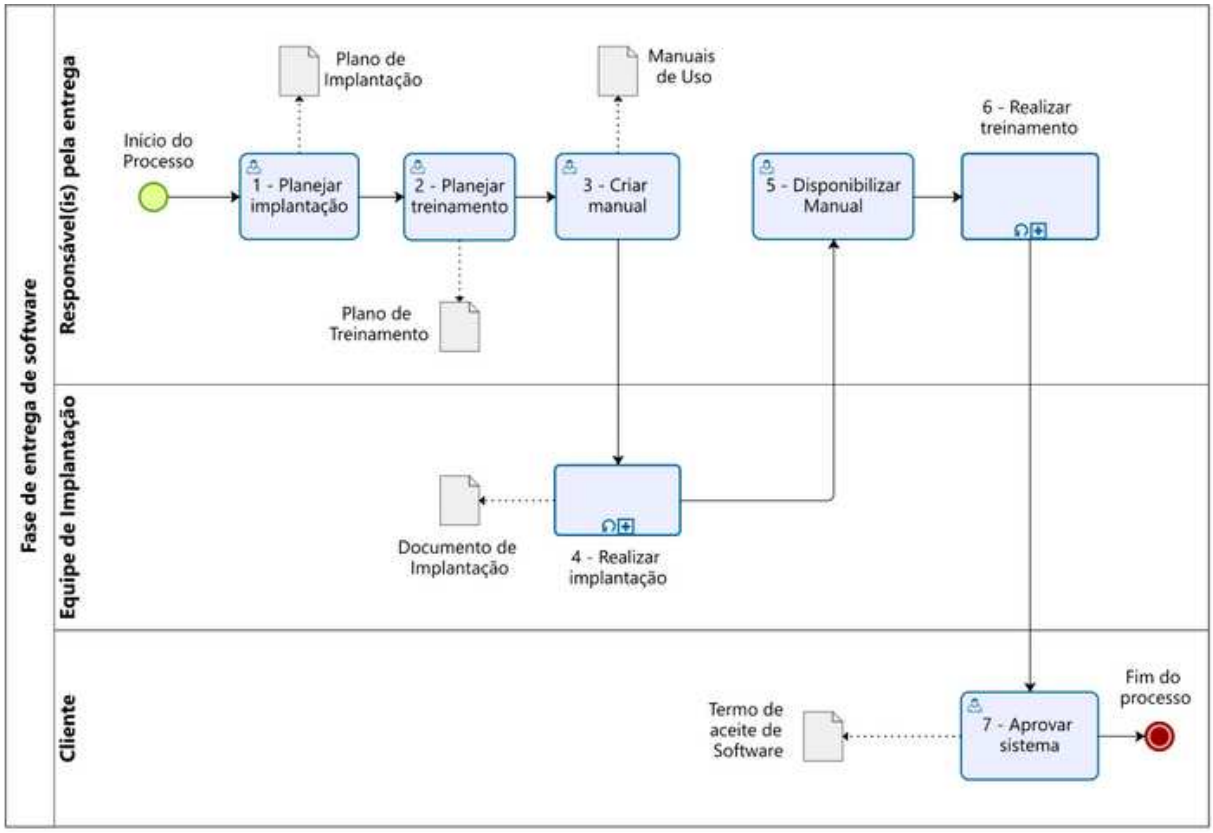
Fonte: Autor (2022).

3.2.5.1.4 *Quarta Fase*

A quarta fase, Entrega, tem foco nas atividades relacionadas a implantação do software no ambiente do cliente, então é visto muitos processos referentes ao planejamento de atividades e com isso a geração de artefatos, como exemplo os planos de implantação e treinamento.

Há também nessa fase o foco nas atividades de treinamento dos usuários, essa atividade é importante para capacitar os usuários do sistema a utilizarem o software. Além do treinamento, há o processo de criação de manual(is) de uso, para garantir que os futuros usuários possam utilizar o sistema sem impasses.

Figura 14 — Fase de Entrega



Fonte: Autor (2022).

3.2.5.2 Descrição de cada Fase

Após a apresentação da modelagem BPMN é apresentado um quadro com os papéis dos atores envolvidos na presente fase, desse modo, além de nomear o papel de cada ator, é mostrado uma definição desse papel, com suas responsabilidades e quem pode exercer o papel.

A seguir será mostrado um exemplo do quadro de papéis. O exemplo será da

fase de Elicitação de Requisitos:

Quadro 1 — Fase de Elicitação de Requisitos - Papéis

Papel	Definição
Engenheiro de Requisitos	Responsável por fazer toda a coleta e documentação dos requisitos com o cliente. Esse tem o papel de ter uma boa comunicação com o cliente. Geralmente será um docente.
Fornecedor(es) de Requisitos	Responsável ou responsáveis por apresentar a demanda de forma clara e inteligível, geralmente esse será o cliente solicitante ou alguém que entenda a fundo a problemática do cliente.

Fonte: Autor (2022).

Ainda na descrição de cada fase, é então, feito um quadro para descrever cada atividade realizada dentro da fase. Nesse quadro é apresentado o nome da atividade, as entradas necessárias para realizar a atividade, os procedimentos para realização da atividade e as saídas que a atividade terá.

A seguir será mostrado um exemplo da descrição de cada atividade. O exemplo será da primeira atividade (Identificar fornecedor(es) de requisitos) da fase de Elicitação de Requisitos:

Quadro 2 — Identificar fornecedor(es) de requisitos

Atividade	1 - Identificar fornecedor(es) de requisitos
Entradas	Informações sobre o cliente.
Procedimentos	O engenheiro de requisitos identifica no ambiente do cliente, aquele ou aqueles que entendem os processos internos do cliente e são aptos a fornecer requisitos e realizar reuniões de alinhamento.
Saídas	Definição dos responsáveis por fornecer os requisitos ao engenheiro de requisitos.

Fonte: Autor (2022).

3.2.5.3 Tabela de Artefatos

Ao decorrer do desenvolvimento da modelagem, foi visto a necessidade da criação de uma tabela para listar todos os artefatos que devem ser gerados durante o ciclo de vida de um software.

Os artefatos de software são subprodutos imprescindíveis dentro do ciclo de vida de um software, não só para auxiliar a equipe de desenvolvimento durante as atividades, mas também para haver um registro daquilo que é gerado durante a produção do software, isso gera uma garantia para o cliente e para os envolvidos no

desenvolvimento.

Dessa forma, foi construída a tabela de artefatos que será apresentada a seguir:

Figura 15 — Tabela de Artefatos

	Fases de Software			
Responsável	**1º Engenharia de Requisitos**	**2º Modelagem**	**3º Codificação e Testes**	**4º Entrega**
Gerente de Projeto	Projeto de Software			
Engenheiro de Requisitos	Documento de Requisitos			
	Registros de reuniões			
Engenheiro de Software		Documento de Arquitetura de Software		
		Diagrama de Caso de Uso		
		Diagrama de Atividades		
		Diagrama de Classes		
		Diagrama de Domínio		
		Diagrama de Banco de dados		
		Diagramas opicionais, caso necessário		
Desenvolvedor			Código Fonte	
Analista de Testes			Plano de Testes	
			Relatório de Testes	
Responsável pela Entrega				Plano de Implantação
				Plano de Treinamento
				Manuais de Uso
				Documento de Implantação
Cliente				Termo de aceite de Software

Fonte: Autor (2022).

Entendendo a importância dos artefatos de software, os mesmos foram distribuídos em suas respectivas fases. Cada uma das quatro fases possuem artefatos que necessitam ser gerados e entregues, a tabela além de fazer essa distribuição, apresenta quem é o responsável, ou responsáveis caso necessário, para gerar o artefato.

3.2.5.4 Descrição dos Artefatos, Pontos Fundamentais e Diretrizes

Após a criação da tabela, foi encontrada a necessidade de realizar uma breve descrição de cada artefato, enumerando pontos chave e diretrizes que poderão auxiliar os responsáveis por cada artefato a gerarem o mesmo.

Essa enumeração de pontos e diretrizes, como dito no documento, pode ter

um conteúdo maior ou menor dependendo da necessidade do projeto ou critério do responsável, dessa forma evitando a inflexibilidade de cada artefato.

4 **RESULTADOS**

A modelagem BPMN do processo de desenvolvimento de software do ESC/NIT foi concebida pela necessidade de apoiar o ciclo de desenvolvimento de software dos projetos produzidos pela UNITINS.

Para isso, durante a realização do presente trabalho foram definidos como procedimentos principais de pesquisa a bibliográfica e a participante. Posteriormente será discutido como a escolha dos tipos de pesquisa impactaram na construção da modelagem e no conhecimento adquirido durante a produção do mesmo, mas antes, será apresentado um tópico sobre limitações na etapa de testes da modelagem e análise dos resultados que poderiam ser coletados e analisados.

4.1 LIMITAÇÕES DA ANÁLISE

Durante a etapa do projeto de pesquisa, foi apontado um ponto pela banca examinadora que iria comprometer um dos objetivos específicos do trabalho, esse objetivo era “Realizar testes com o referido trabalho desenvolvido”. Esse objetivo tinha como finalidade colocar a modelagem trabalhada em um projeto real dentro do ESC e verificar a eficácia do seguimento das etapas durante o ciclo de desenvolvimento do software, entretanto, é nítido que a duração de um ciclo completo de desenvolvimento ultrapassaria o tempo de criação e entrega deste trabalho, sendo assim a análise dos resultados não pôde ser feita aplicando testes práticos na modelagem e apresentando os resultados graficamente e quantitativamente.

Com a devida instrução recebida, tanto por parte da banca, como também pela orientação, a etapa de análise de resultados e discussão deste trabalho não terá foco nos dados quantitativos e gráficos dos resultados, entretanto, em aspectos qualitativos proporcionados pela pesquisa bibliográfica e adoção de práticas afamadas utilizadas no mercado de trabalho.

4.2 PROCEDIMENTOS DE PESQUISA E SEUS RESULTADOS

Nas seções a seguir serão apresentados os tipos de pesquisa delimitados e como esses tipos foram colocados em prática, além da discussão do que foi aprendido em vista o que autores tem a dizer sobre o assunto.

4.2.1 **Análise e Discussão com a Pesquisa Participante**

Ao decorrer das atividades foi notório que a pesquisa participante foi

fundamental para entender com maior profundidade o cotidiano e os problemas encontrados no ambiente de desenvolvimento do ESC.

Essa visão também entra em concordância com Guerra (2014), onde a autora diz que a pesquisa e observação participante ocasionará a maior profundidade da compreensão sobre um tema. Também em concordância com essa visão, tanto LIMA (2008) quanto MINAYO (2008), citado por GUERRA (2014, p.32) reafirmam o ganho de experiência do pesquisador ao fazer parte do contexto da observação, além de apontarem que essa observação participante é a técnica mais utilizadas em pesquisas de natureza qualitativas atualmente.

Dessa forma, com a atuação bem próxima ao ESC, e com reuniões semanais durante todo o período de desenvolvimento do trabalho com o professor orientador Fredson Vieira Costa, esse que também é o Coordenador dos projetos desenvolvidos no ESC, logo, tendo grande conhecimento das tarefas, processos e rotinas encontradas no ESC, possibilitou o alcance de maior compreensão e domínio das atividades necessárias para realizar a modelagem desse projeto.

Sem a pesquisa participante pode se dizer que a construção deste trabalho seria inatingível ou no mínimo improvável, pois sem a clareza das atividades que o ESC busca alcançar atualmente, ou sem a orientação de alguém que esteja altamente envolvido nas atividades desenvolvidas, o produto final, no caso a modelagem, não seria condizente às necessidades do ESC.

4.2.2 **Análise e Discussão com a Pesquisa Bibliográfica**

A pesquisa bibliográfica foi realizada em parceria com a pesquisa participante para reafirmar a necessidade de atividades que garantam a qualidade de um software com base no que a literatura renomada e moderna tem a oferecer.

Ao reconhecer processos necessários que ainda não haviam sido definidos dentro do ESC, como por exemplo os testes unitários e code review, a bibliografia foi requisitada para entender melhor como, quando e por quem essas práticas poderiam ser realizadas.

Dessa forma é visível que cada fase de software modelada tem um referencial bibliográfico encontrado na seção de revisão de literatura, dessa maneira a produção da modelagem juntamente ao seu documento entre em conformidade com a visão transmitida pelos autores, essas com foco em processos bem definidos, com seus respectivos atores e logicamente ordenados.

4.3 CONSIDERAÇÕES FINAIS

Ao final do desenvolvimento do presente trabalho, foi visto que uma solução

para a problemática da falta de processos bem definidos dentro do ESC foi desenvolvida.

Para alcançar o objetivo almejado, inicialmente foi definido que a primeira etapa para o desenvolvimento do trabalho seria a pesquisa e aprofundamento sobre a temática e levantamento bibliográfico preliminar.

Na segunda etapa das atividades desenvolvidas, foi realizada a escrita da composição do texto, como por exemplo os objetivos, geral e específicos, justificativas, entre outros tópicos.

A terceira etapa do projeto teve o foco no desenvolvimento da revisão de literatura e definição das metodologias de pesquisa.

A quarta etapa do projeto teve foco no desenvolvimento do documento da modelagem, ou seja, o produto do trabalho, esse foi refeito e validado diversas vezes até ter aprovação do professor orientador e feedbacks de outros membros do ESC, esses feedbacks foram incorporados ao trabalho final.

O produto gerado com o presente trabalho foi o Documento “Modelagem BPMN do Processo de Desenvolvimento de Software do ESC/NIT da UNITINS", onde o detalhamento de sua concepção e divisão pode ser encontrada na seção de metodologia e o documento na íntegra pode ser encontrado no APÊNDICE A — Modelagem BPMN do Processo de Desenvolvimento de Software do ESC/NIT da UNITINS.

5 **CONCLUSÃO**

No que tange o desenvolvimento da modelagem dos processos do ESC, foi perceptível os proveitos e a entrega de valor gerados ao final do desenvolvimento deste trabalho. Através do documento da modelagem e dos itens que o compõem, a contribuição positiva deste estende a todos os envolvidos no ESC e aos futuros projetos desenvolvidos dentro da universidade.

Através da consulta do documento de modelagem, principalmente por parte dos acadêmicos, será possível ter uma melhor noção dos processos que necessitam ser desenvolvidos, as fases de desenvolvimento, os artefatos gerados, as diretrizes e pontos em cada artefato, as responsabilidades dos envolvidos, as diretrizes e fluxos das atividades, entre outros benefícios.

Existiram limitações durante o desenvolvimento do projeto, principalmente relacionado ao tempo de trabalho, esse que impactou na impossibilidade da testagem e avaliação do documento da modelagem por parte de outros integrantes do ESC, pois para obter um resultado satisfatório e avaliá-lo em um cenário de desenvolvimento real seria necessário um maior prazo que o proporcionado para a entrega deste trabalho.

Levando em consideração que a devida modelagem foi desenvolvida de forma a solucionar uma problemática personalizada em um contexto acadêmico, pode se notar que mesmo sendo uma proposta para ser implantada em contexto educacional, a mesma também pode facilmente ser utilizada em um cenário comercial, visto que os pontos visados durante a modelagem tem a função de garantir a qualidade dos produtos desenvolvidos e a gradual profissionalização dos envolvidos.

Dessa forma, é possível vislumbrar trabalhos futuros relacionados à comparação do modelo proposto com outras soluções modeladas, tanto em aspectos acadêmicos e corporativos, a fim de trazer uma visão mais detalhada e traçada a fim de levantar pontos de convergência e elaborar hipóteses sobre a semelhança e divergências que possam ser encontradas.

É possível vislumbrar também, atividades relacionadas a testagem e avaliação do documento da modelagem dentro de ciclos de vida de desenvolvimento, além disso a criação de novos trabalhos desenvolvidos por outros acadêmicos semelhantes no que se refere a adições benéficas ao ESC no que tange à áreas de desenvolvimento de software, de forma a encontrar novos problemas e propor soluções, a fim de contribuir positivamente para geração de conhecimento e melhoria da qualidade em produtos desenvolvidos dentro da UNITINS.

REFERÊNCIAS

ABPMP. **BPM CBOK**: Versão 3.0. 1 ed. Brasil: Association of Business Process Management Professionals, 2013.

BARTIÉ, Alexandre. **Garantia da qualidade de software**. 13 ed. Rio de Janeiro: Elsevier Editora, 2002.

BONITASOFT. **The Ultimate Guide to BPMN2**. 2021. 26 p.

BRITO, Rebeka Sales de. **Uma Proposta para Modelagem de Requisitos Não-Funcionais em Projetos Ágeis**. Recife, 2010 Dissertação (Pós-Graduação em Ciência da Computação) - Universidade Federal de Pernambuco, Recife, 2010.

CERQUEIRA, José Antonio Siqueira de. **Modelagem de processos do Código de Processo Penal com BPMN**. Brasília, 2017 Monografia (Ciência da Computação) - Universidade de Brasília, Brasília, 2017.

CGU, Controladoria-Geral da União; SE, Secretaria-Executiva; DIPLAD, Diretoria de Planejamento e Desenvolvimento Institucional. **Guia de Modelagem de Processos de Negócio da CGU**. 1 ed. Brasil, 2020.

DELAMARO, Márcio; JINO, Mario; MALDONADO, José. **Introdução ao Teste De Software**. 2 ed. Rio de Janeiro: Elsevier Brasil, 2016.

FABRIS, Polyanna Pacheco Gomes; PERINI, Luis Cláudio. **Processos de software**. 1 ed. Londrina: Educacional S.A., 2014.

FONSECA, Igor O. *et al*. **BPMN, SPEM e Essence no contexto da Modelagem de Processos de Software**: uma Revisão Sistemática da Literatura. Porto Alegre, 2021 Dissertação - Escola Regional de Engenharia de Software (eres), Evento Online, 2021.

GIL, Antonio Carlos. **Como elaborar projetos de pesquisa**. 6 ed. São Paulo: Atlas, 2017.

GUERRA, Elaine Linhares de Assis. **Manual de Pesquisa Qualitativa**. 1 ed. Belo Horizonte: Grupo Ănima Educação, 2014.

IBM. **Rational Software Architect Standard Edition**: Artifacts. Disponível em: https://www.ibm.com/docs/pt-br/rsas/7.5.0?topic=diagrams-artifacts. Acesso em: 19 out. 2022.

IPROCESS EDUCATION. **Guia BPMN 2.0**. 1 ed. iProcess, 2014. 1 p.

KOTONYA, Gerald; SOMMERVILLE, Ian. **Requirements Engineering**: Processes and Techniques. Chichester: John Wiley & Sons Incorporated, 1998.

LE BOTERF, Guy. Pesquisa participante: Propostas e reflexões metodológicas*. In:* BRANDÃO, Carlos Rodrigues. **Repensando a pesquisa participante**. 3 ed. São Paulo: Brasiliense, 1999.

MACHADO, Felipe Nery Rodrigues. **Análise e Gestão de Requisitos de Software**: Onde nascem os sistemas. 3 ed. São Paulo: Saraiva Educação S.A., 2016.

MINAYO, Maria Cecília de Souza. **O desafio do conhecimento**. 11 ed. São Paulo: Hucitec, 2008.

NETO, Roque Maitino. **Engenharia de software**. 1 ed. Londrina: Educacional S.A., 2016.

PRESSMAN, Roger S.; MAXIM, Bruce R.. **Engenharia de software**: Uma Abordagem Profissional. 8 ed. Porto Alegre: AMGH Editora, 2016.

PROJECT MANAGEMENT INSTITUTE: Chapters Brasileiros. **Estudo de Benchmarking em Gerenciamento de Projetos Brasil**. 2010. 123 slides. Disponível em: http://www.mp.go.gov.br/portalweb/hp/33/docs/benchmarking_gp_2010_geral.pdf. Acesso em: 19 mar. 2022.

REZENDE, Denis Alcides. **Engenharia de Software e Sistemas de Informação**. 3 ed. Rio de Janeiro: Brasport, 2005.

SOMMERVILLE, Ian. **Engenharia de Software**. 10 ed. São Paulo: Pearson Education, 2019.

SOMMERVILLE, Ian. **Engenharia de Software**. 9 ed. São Paulo: Pearson Education, 2011.

STANKIEWICZ, Alessandro. **Modelo de Interação Ágil**: UMA ADAPTAÇÃO DO MODELO CASCATA À ORGANIZAÇÃO DE PEQUENAS E MÉDIAS EMPRESAS. PONTA GROSSA, 2017 Trabalho de Conclusão de Curso (TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS) - Universidade Tecnológica Federal do Paraná, PONTA GROSSA, 2017.

VALENTE, Marco Tulio. **Engenharia de Sofware Moderna**: princípios e práticas para desenvolvimento de software com produtividade. Belo Horizonte, 2020.

VAZQUEZ, Carlos Eduardo; SIMÕES, Guilherme Siqueira. **Engenharia de Requisitos**: software orientado ao negócio. São Paulo: Brasport, 2016.

WAZLAWICK, Raul Sidnei. **Engenharia de software**: conceitos e práticas. 2 ed. Rio de Janeiro: GENLTC, 2019.

APÊNDICE A — MODELAGEM BPMN DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DO ESC/NIT DA UNITINS

UNIVERSIDADE ESTADUAL DO TOCANTINS - UNITINS
CURSO DE SISTEMAS DE INFORMAÇÃO

IAGO BATISTA ANTUNES LEOBAS

MODELAGEM BPMN DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DO ESC/NIT DA UNITINS

Palmas - TO
2022

IAGO BATISTA ANTUNES LEOBAS

MODELAGEM BPMN DO PROCESSO DE DESENVOLVIMENTO DE SOFTWARE DO ESC/NIT DA UNITINS

Projeto apresentado ao Curso de Sistemas de Informação da Universidade Estadual do Tocantins - UNITINS como parte dos requisitos para obtenção do grau de Bacharel em Sistemas de Informação, sob a orientação do Professor Fredson Vieira Costa.

Orientador: Fredson Vieira Costa

Palmas - TO
2022

LISTA DE ILUSTRAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

MVC	Model-View-Controller
MVVM	Model-View-ViewModel
BPMN	Business Process Model and Notation

SUMÁRIO

1 **INTRODUÇÃO**

O Escritório de Soluções Criativas foi instituído no ano de 2018 na Universidade Estadual do Tocantins, seu objetivo compreende o desenvolvimento de projetos por intermédio da pesquisa aplicada ao setor produtivo, dessa forma, através de demandas estruturadas tanto pelo setor público quanto o privado há aplicação mediante a orientação e coordenação de professores da UNITINS com a visão de ingressar novos acadêmicos de maneira prática às situações baseadas na realidade do dia a dia do setor de produção, no caso específico deste trabalho ao setor de software, oferecendo a possibilidade de experimentação da prática profissional exigida no mercado de trabalho.

Dado o cenário de desenvolvimento, o presente trabalho é uma solução auxiliadora que tem por objetivo apresentar de forma detalhada e visual uma modelagem do processo de desenvolvimento de software do ESC.

Para essa definição de processos a serem seguidos, a ferramenta de representação e modelagem dos processos utilizada é a BPMN, notação que auxiliará a visualização não apenas das atividades que devem ser realizadas, mas por quem as realiza e os artefatos de software que devem ser gerados ao decorrer do fluxo de desenvolvimento.

2 **DESENVOLVIMENTO DA MODELAGEM**

Inicialmente será apresentada a modelagem BPMN das atividades e a descrição dos papéis dos atores e das atividades em cada fase. Em seguida será apresentada a tabela de artefatos gerada juntamente com seu conceito, ambos serão apresentados posteriormente neste documento.

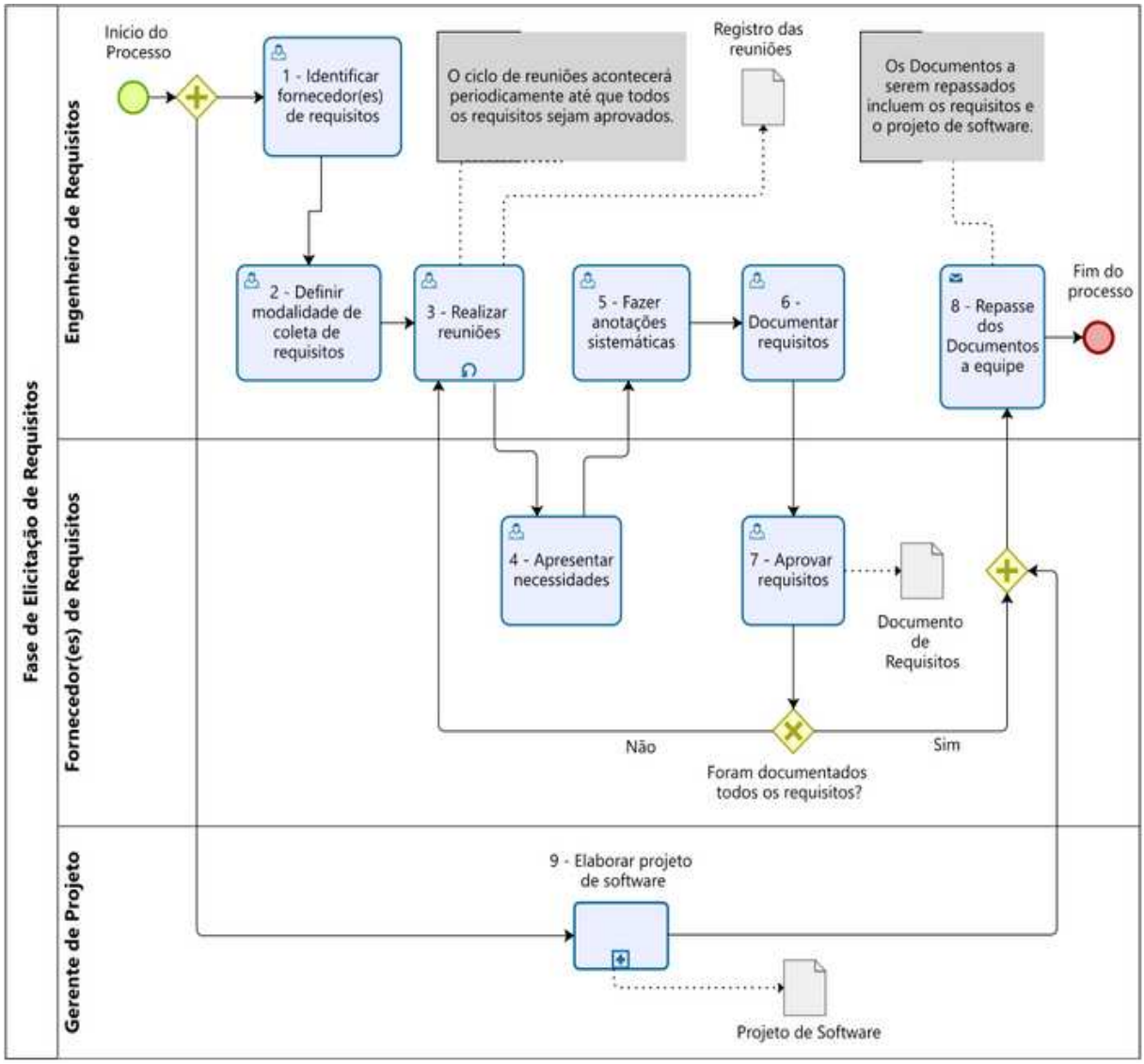
2.1 MODELAGEM BPMN

A partir do próximo capítulo será apresentada a modelagem BPMN de todas as fases de desenvolvimento. Logo após cada modelagem será apresentada a descrição de todos os atores envolvidos na fase e a descrição de cada atividade. Caso uma atividade possua um artefato gerado ou necessite de um artefato para seguimento, será informado seu número de identificação, correspondente aos artefatos informados na seção 2.2 deste documento, e seu respectivo nome.

2.1.1 **Modelagem da Fase de Elicitação de Requisitos**

A seguir será apresentada a modelagem da Fase de Elicitação de Requisitos e sua descrição detalhada.

Figura 1 — Fase de Elicitação de Requisitos



Fonte: Autor (2022).

Quadro 1 — Fase de Elicitação de Requisitos - Papéis (continua)

Papel	Definição
Engenheiro de Requisitos	Responsável por fazer toda a coleta e documentação dos requisitos com o cliente. Esse tem o papel de ter uma boa comunicação com o

Quadro 1 — Fase de Elicitação de Requisitos - Papéis (conclusão)

Papel	Definição
	cliente. Geralmente será um docente.
Fornecedor(es) de Requisitos	Responsável ou responsáveis por apresentar a demanda de forma clara e inteligível, geralmente esse será o cliente solicitante ou alguém que entenda a fundo a problemática do cliente.
Gerente de Projeto	Responsável pelo concepção projeto de software, precisando entender a fundo todos os aspectos do desenvolvimento. Geralmente será um docente.

Fonte: Autor (2022).

Quadro 2 — 1 - Identificar fornecedor(es) de requisitos

Atividade	**1 - Identificar fornecedor(es) de requisitos**
Entradas	Informações sobre o cliente.
Procedimentos	O engenheiro de requisitos identifica no ambiente do cliente, aquele ou aqueles que entendem os processos internos do cliente e são aptos a fornecer requisitos e realizar reuniões de alinhamento.
Saídas	Definição dos responsáveis por fornecer os requisitos ao engenheiro de requisitos.

Fonte: Autor (2022).

Quadro 3 — 2 - Definir modalidade de coleta de requisitos

Atividade	**2 - Definir modalidade de coleta de requisitos**
Entradas	Informações sobre disponibilidade e preferências sobre a modalidade de reuniões com o fornecedor de requisitos.
Procedimentos	O engenheiro de requisitos escolhe o formato mais adequado para o levantamento de requisitos com o cliente. Esses requisitos podem ser coletados utilizando um modelo a critério do engenheiro e em conformidade com a disponibilidade do fornecedor.
Saídas	Modalidade de coleta de requisitos a ser utilizada com o fornecedor de requisitos.

Fonte: Autor (2022).

Quadro 4 — 3 - Realizar reuniões e 4 - Apresentar necessidades

Atividade	**3 - Realizar reuniões** **4 - Apresentar necessidades**
Entradas	Modalidade de reuniões definida.
Procedimentos	O engenheiro de requisitos se reúne com o fornecedor de requisitos, este por sua vez apresenta ao engenheiro as necessidades da organização e a solução almejada. Esse ciclo de atividades pode ocorrer mais de uma vez dependendo da necessidade e seguimento do projeto.
Saídas	Artefato gerado: (2.2.3 Registro das Reuniões) e coletado informações chave sobre a solução.

Fonte: Autor (2022).

Quadro 5 — 5 - Fazer anotações sistemáticas

Atividade	**5 - Fazer anotações sistemáticas**
Entradas	Apresentação das necessidades por parte do cliente ao engenheiro de requisitos.
Procedimentos	O engenheiro de requisitos pode utilizar artifícios de anotações para elaborar um pré-documento de requisitos, esse que irá registrar as necessidades do cliente e auxiliá-lo durante a elaboração do documento oficial.
Saídas	Registro das necessidades do cliente.

Fonte: Autor (2022).

Quadro 6 — 6 - Documentar requisitos

Atividade	**6 - Documentar requisitos**
Entradas	Registro das necessidades do cliente.
Procedimentos	O engenheiro de requisitos cria o documento de requisitos de acordo com o que foi passado pelo fornecedor de requisitos.
Saídas	Documento de requisitos pré-aprovado.

Fonte: Autor (2022).

Quadro 7 — 7 - Aprovar requisitos

Atividade	7 - Aprovar requisitos
Entradas	Documento de requisitos.
Procedimentos	O fornecedor de requisitos aprova ou solicita alterações no (2.2.2 Documento de Requisitos).
Saídas	Artefato gerado: (2.2.2 Documento de Requisitos), ou nova reunião para alinhamento de necessidades.

Fonte: Autor (2022).

Quadro 8 — 8 - Repasse dos requisitos à equipe

Atividade	8 - Repasse dos requisitos à equipe
Entradas	Documento de requisitos aprovado pelo fornecedor de requisitos.
Procedimentos	O engenheiro de requisitos envia o (2.2.2 Documento de Requisitos) aprovado e (2.2.1 Projeto de Software) para a equipe para seguimento do projeto.
Saídas	Recebimento do (2.2.2 Documento de Requisitos) e (2.2.1 Projeto de Software) por parte da equipe.

Fonte: Autor (2022).

Quadro 9 — 9 - Elaborar projeto de software

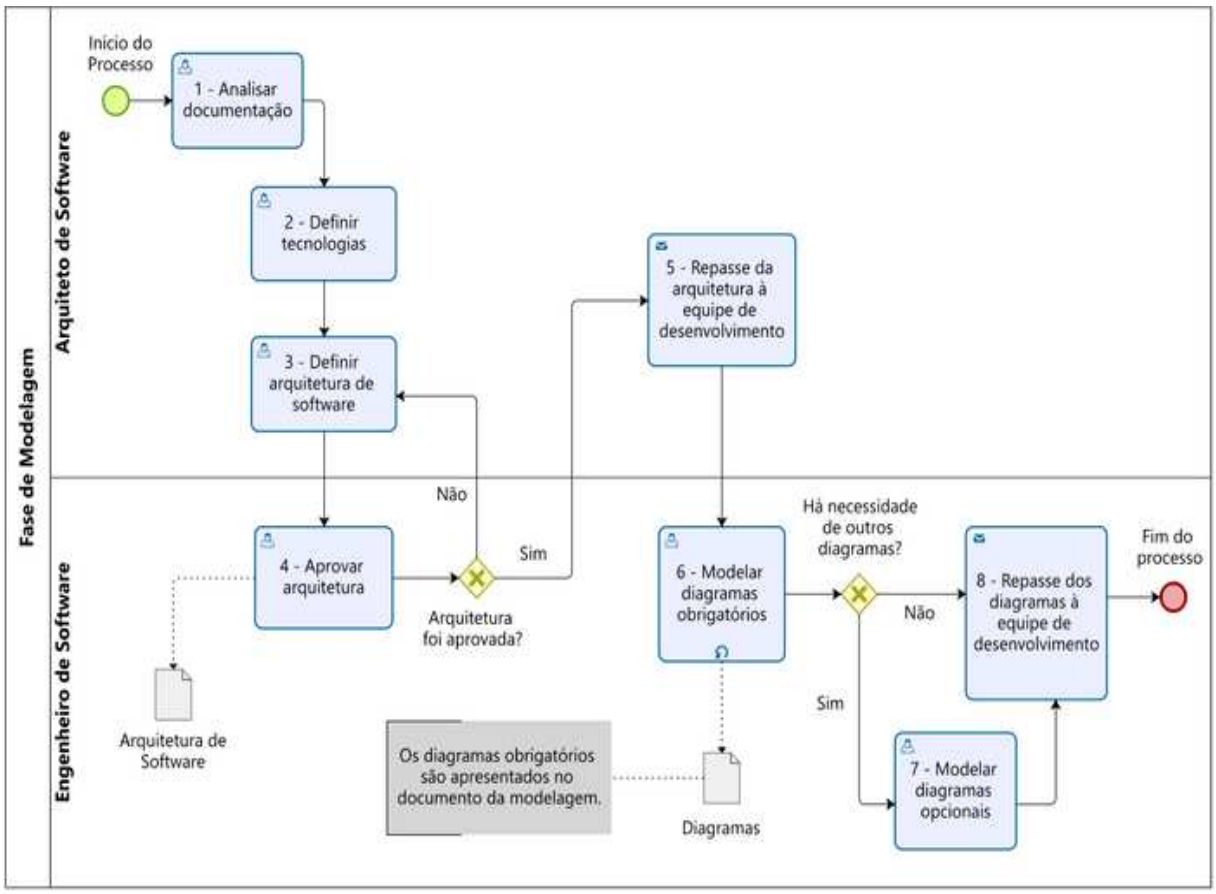
Atividade	9 - Elaborar projeto de software
Entradas	Início do desenvolvimento do projeto.
Procedimentos	O gerente de projeto elabora o (2.2.1 Projeto de Software).
Saídas	Artefato gerado: (2.2.1 Projeto de Software).

Fonte: Autor (2022).

2.1.2 **Modelagem da Fase de Modelagem**

A seguir será apresentada a modelagem da Fase de Modelagem e sua descrição detalhada.

Figura 2 — Fase de Modelagem



Fonte: Autor (2022).

Quadro 10 — Fase de Modelagem - Papéis

Papel	Definição
Engenheiro de Software	Responsável por aprovar a arquitetura e modelar os diagramas necessários para o projeto. Geralmente será um docente, entretanto pode ter auxílio de um discente durante a modelagem.
Arquiteto de Software	Responsável por fazer o estudo e definição da arquitetura de software que será adotada durante o ciclo de desenvolvimento do projeto. Geralmente será um docente.

Fonte: Autor (2022).

Quadro 11 — 1 - Análise da documentação

Atividade	1 - Análise da documentação
Entradas	Documento de requisitos aprovado.
Procedimentos	O arquiteto de software estuda a documentação do projeto para entender o sistema que será desenvolvido com o intuito de escolher as tecnologias que mais se adequam ao sistema e sua respectiva arquitetura.
Saídas	Documentação examinada pelo arquiteto de software.

Fonte: Autor (2022).

Quadro 12 — 2 - Definir tecnologias

Atividade	2 - Definir tecnologias
Entradas	Análise da documentação e compreensão da demanda.
Procedimentos	O arquiteto de software estabelece as tecnologias que serão utilizadas para a construção do sistema, analisando a demanda do projeto e a realidade da equipe de desenvolvimento.
Saídas	Determinação das tecnologias que serão utilizadas.

Fonte: Autor (2022).

Quadro 13 — 3 - Definir arquitetura de software e 4 - Aprovar arquitetura

Atividade	3 - Definir arquitetura de software 4 - Aprovar arquitetura
Entradas	Tecnologias definidas.
Procedimentos	O arquiteto de software estabelece toda a estrutura arquitetural do sistema e envia para aprovação do engenheiro de software, este por sua vez analisa o projeto de arquitetura e aprova ou solicita alterações.
Saídas	Artefato gerado: (2.2.4 Documento de arquitetura de software).

Fonte: Autor (2022).

Quadro 14 — 5 - Repasse da arquitetura para a equipe de desenvolvimento

Atividade	**5 - Repasse da arquitetura para a equipe de desenvolvimento**
Entradas	(2.2.4 Documento de arquitetura de software) aprovada.
Procedimentos	O arquiteto de software disponibiliza o (2.2.4 Documento de arquitetura de software) à equipe de desenvolvimento.
Saídas	Envio do (2.2.4 Documento de arquitetura de software) à equipe.

Fonte: Autor (2022).

Quadro 15 — 6 - Modelar diagramas obrigatórios e 7 - Modelar diagramas opcionais

Atividade	**6 - Modelar diagramas obrigatórios** **7 - Modelar diagramas opcionais**
Entradas	Arquitetura de software disponibilizada.
Procedimentos	O engenheiro de software desenha os (2.2.5 Diagramas) necessários de acordo com a necessidade do projeto.
Saídas	Artefatos gerados: (2.2.5 Diagramas).

Fonte: Autor (2022).

Quadro 16 — 8 - Repasse dos diagramas à equipe de desenvolvimento

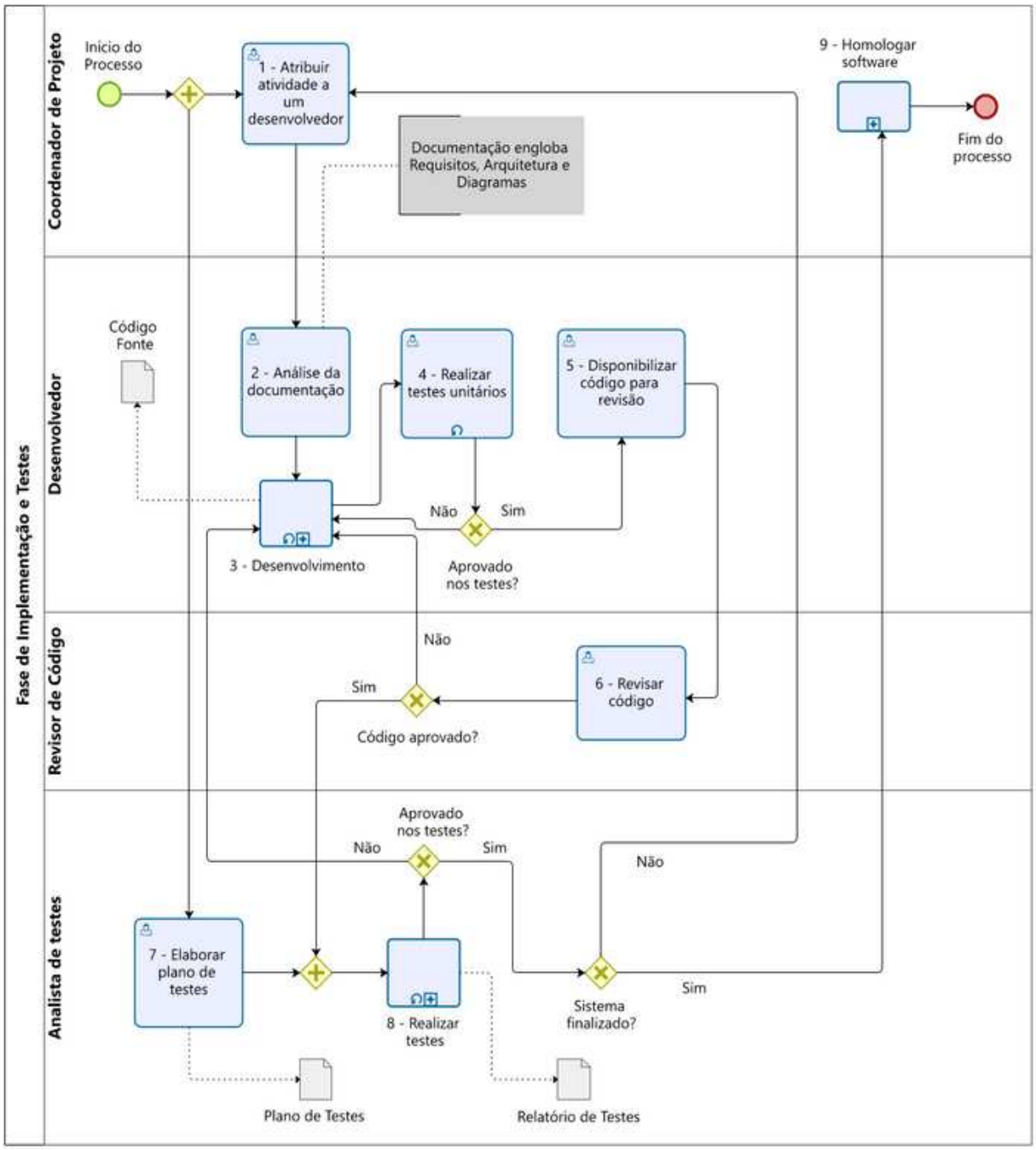
Atividade	**8 - Repasse dos diagramas à equipe de desenvolvimento**
Entradas	(2.2.5 Diagramas) gerados.
Procedimentos	O engenheiro de software disponibiliza os (2.2.5 Diagramas) à equipe de desenvolvimento.
Saídas	Envio dos (2.2.5 Diagramas) à equipe.

Fonte: Autor (2022).

2.1.3 **Modelagem da Fase de Implantação e Testes**

A seguir será apresentada a modelagem da Fase de Implantação e Testes e sua descrição detalhada.

Figura 3 — Fase de Implantação e Testes



Fonte: Autor (2022).

Quadro 17 — Fase de Implantação e Testes - Papéis

Papel	Definição
Coordenador de Projeto	Responsável por facilitar e garantir que o time de desenvolvedores estejam produzindo entregas e gerando valor dentro do projeto. Geralmente será um docente.
Desenvolvedor	Responsável por fazer a escrita do código fonte do projeto. Geralmente será um discente.
Revisor de Código	Responsável por fazer a revisão do código de outro desenvolvedor, esse papel deve ser realizado por um desenvolvedor que não escreveu o código para seguir as diretrizes da prática de revisão. Geralmente será um discente.
Analista de testes	Responsável por elaborar um plano de testes e garantir sua execução. Geralmente será um docente que fará o plano de testes e a execução dos testes podem ser feitas por um discente.

Fonte: Autor (2022).

Quadro 18 — 1 - Atribuir atividade a um desenvolvedor

Atividade	1 - Atribuir atividade a um desenvolvedor
Entradas	Atividades que necessitam ser desenvolvidas.
Procedimentos	O coordenador de projeto designa tarefas ao time de desenvolvimento ou a um desenvolvedor em específico, dependendo da metodologia de desenvolvimento.
Saídas	Atribuição de desenvolvedor à determinada atividade.

Fonte: Autor (2022).

Quadro 19 — 2 - Análise da documentação

Atividade	2 - Análise da documentação
Entradas	Atividade atribuída.
Procedimentos	O desenvolvedor faz um estudo preliminar ao processo de desenvolvimento sobre o contexto e especificações da funcionalidade ou tarefa que irá exercer.
Saídas	Entendimento da lógica de negócio e/ou aplicação.

Fonte: Autor (2022).

Quadro 20 — 3 - Desenvolvimento

Atividade	3 - Desenvolvimento
Entradas	Entendimento da lógica de negócio e/ou aplicação.
Procedimentos	O desenvolvedor inicia o processo de codificação, esse processo tende a ser retrabalhado diversas vezes e tem o intuito de passar a abstração inicial do sistema para linhas de código.
Saídas	Artefato gerado: (2.2.6 Código Fonte).

Fonte: Autor (2022).

Quadro 21 — 4 - Realizar testes unitários

Atividade	4 - Realizar testes unitários
Entradas	(2.2.6 Código Fonte) gerado
Procedimentos	O desenvolvedor aplica testes unitários ao código desenvolvido, a fim de identificar possíveis problemas nas entradas e saídas do (2.2.6 Código Fonte) gerado enquanto este é encontrado em pequeno tamanho unitário. As diretrizes para realização dos testes unitários podem ser recebidas com mais detalhes através do (2.2.7 Plano de Testes) criado pelo analista de testes.
Saídas	Testes unitários realizados no (2.2.6 Código Fonte).

Fonte: Autor (2022).

Quadro 22 — 5 - Disponibilizar código para revisão

Atividade	5 - Disponibilizar código para revisão
Entradas	(2.2.6 Código Fonte) gerado e testado de forma unitária.
Procedimentos	O desenvolvedor disponibiliza o (2.2.6 Código Fonte) no versionador da equipe através de um commit para iniciar a etapa de revisão.
Saídas	(2.2.6 Código Fonte) enviado para revisão.

Fonte: Autor (2022).

Quadro 23 — 6 - Revisar código

Atividade	6 - Revisar código
Entradas	(2.2.6 Código Fonte) disponibilizado para revisão.
Procedimentos	O revisor de código inspeciona o (2.2.6 Código Fonte) postado e repassa o checklist elaborado pelo analista de testes no (2.2.7 Plano de Testes).
Saídas	Código revisado e enviado testes ou devolvido para correção.

Fonte: Autor (2022).

Quadro 24 — 7 - Elaborar plano de testes

Atividade	7 - Elaborar plano de testes
Entradas	Entendimento do sistema.
Procedimentos	O analista de testes constrói o (2.2.7 Plano de Testes).
Saídas	Artefato gerado: (2.2.7 Plano de Testes).

Fonte: Autor (2022).

Quadro 25 — 8 - Realizar testes

Atividade	8 - Realizar testes
Entradas	(2.2.7 Plano de Testes) gerado.
Procedimentos	O analista de testes é encarregado por realizar todos os testes já definidos no (2.2.7 Plano de Testes).
Saídas	(2.2.6 Código Fonte) testado e aprovado, ou devolvido para correção.

Fonte: Autor (2022).

Quadro 26 — 9 - Homologar software

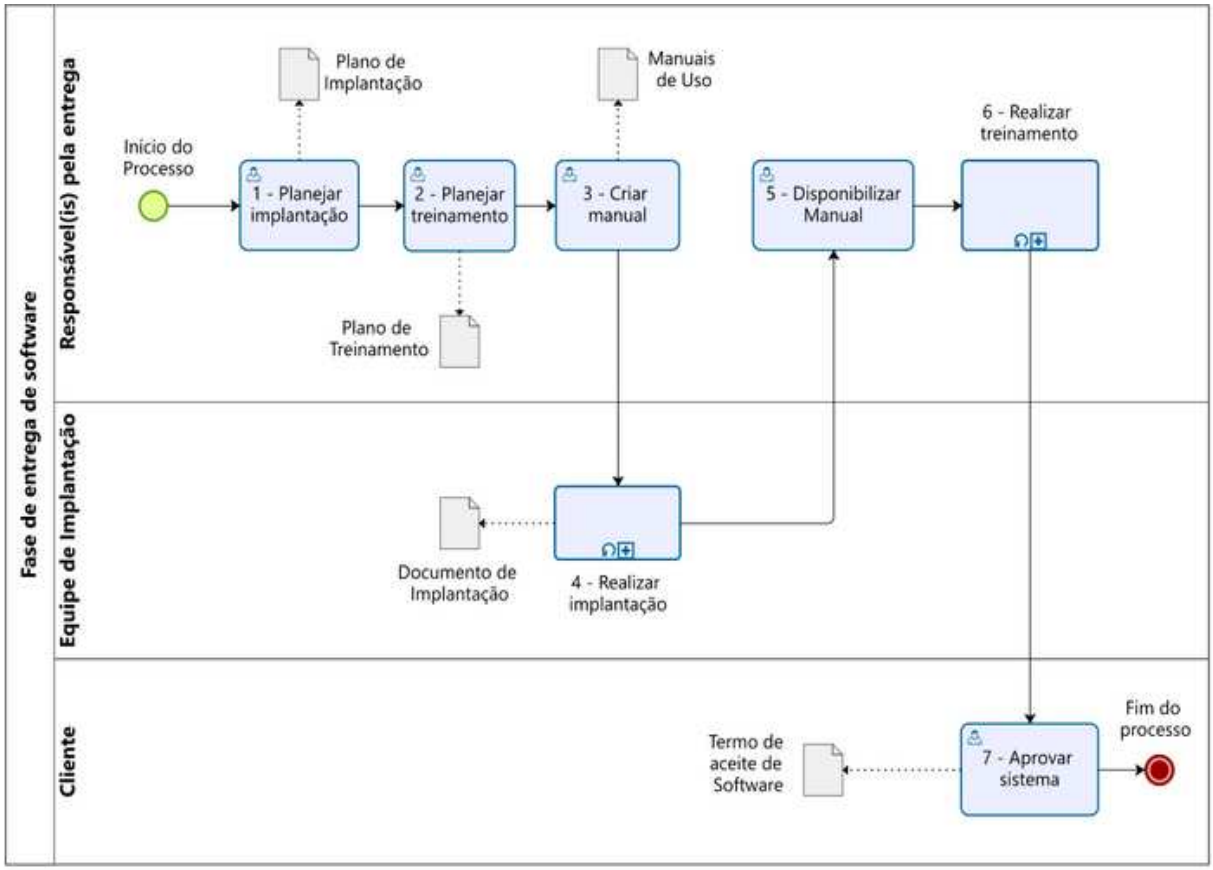
Atividade	9 - Homologar software
Entradas	Sistema finalizado
Procedimentos	O coordenador de projeto realiza os procedimentos de homologação de sistema.
Saídas	Sistema homologado e início da fase de entrega.

Fonte: Autor (2022).

2.1.4 **Modelagem da Fase de Entrega**

A seguir será apresentada a modelagem da Fase de Entrega e sua descrição detalhada.

Figura 4 — Fase de Entrega



Fonte: Autor (2022).

Quadro 27 — Fase de Entrega - Papéis (continua)

Papel	Definição
Responsável(is) pela entrega	Responsável(is) pelas atividades relacionadas ao estabelecimento do software, desde a gerência da implantação, até a entrega de manuais e treinamento dos usuários. Geralmente será o gerente de projeto que irá realizar os processos de planejamento, já os processos de treinamento podem ser realizados por outros docentes ou envolvidos na concepção do sistema.
Equipe de Implantação	Responsáveis pela implantação do software dentro do ambiente de trabalho do cliente. Geralmente serão os desenvolvedores ou

Quadro 27 — Fase de Entrega - Papéis (conclusão)

Papel	Definição
	envolvidos na concepção do sistema.
Cliente	Responsável por aprovar o software, esse ator deve ter poder de decisão para isso. Geralmente um coordenador, diretor ou outro cargo de decisão.

Fonte: Autor (2022).

Quadro 28 — 1 - Planejar implantação

Atividade	1 - Planejar implantação
Entradas	Software finalizado.
Procedimentos	O(s) Responsável(is) pela entrega realizam o planejamento da implantação e geram o (2.2.9 Plano de Implantação).
Saídas	Artefato gerado: (2.2.9 Plano de Implantação).

Fonte: Autor (2022).

Quadro 29 — 2 - Planejar treinamento

Atividade	2 - Planejar treinamento
Entradas	Software finalizado.
Procedimentos	O(s) Responsável(is) pela entrega realizam o planejamento do treinamento dos usuários que irão utilizar o software e geram o (2.2.10 Plano de Treinamento).
Saídas	Artefato gerado: (2.2.10 Plano de Treinamento).

Fonte: Autor (2022).

Quadro 30 — 3 - Criar manual

Atividade	3 - Criar manual
Entradas	Software finalizado e (2.2.10 Plano de Treinamento) gerado.
Procedimentos	O(s) Responsável(is) pela entrega criam o(s) manual(is) de uso e geram o(s) (2.2.11 Manuais de Uso).
Saídas	Artefato gerado: (2.2.11 Manuais de Uso).

Fonte: Autor (2022).

Quadro 31 — 4 - Realizar implantação

Atividade	4 - Realizar implantação
Entradas	(2.2.9 Plano de Implantação) gerado.
Procedimentos	A equipe de implantação realiza a implantação do software de acordo com o (2.2.9 Plano de Implantação) e gera assim o (2.2.12 Documento de Implantação).
Saídas	Artefato gerado: (2.2.12 Documento de Implantação).

Fonte: Autor (2022).

Quadro 32 — 5 - Disponibilizar Manual

Atividade	5 - Disponibilizar Manual
Entradas	(2.2.11 Manuais de Uso) gerado.
Procedimentos	O(s) Responsável(is) pela entrega disponibilizam o (2.2.11 Manuais de Uso) aos usuários do sistema.
Saídas	Entrega dos (2.2.11 Manuais de Uso) aos usuários.

Fonte: Autor (2022).

Quadro 33 — 6 - Realizar treinamento

Atividade	6 - Realizar treinamento
Entradas	Sistema implantado.
Procedimentos	O(s) Responsável(is) pela entrega realiza(m) o treinamento dos usuários do sistema.
Saídas	Usuários capacitados.

Fonte: Autor (2022).

Quadro 34 — 7 - Aprovar sistema (continua)

Atividade	7 - Aprovar sistema
Entradas	Sistema implantado.

Quadro 34 — 7 - Aprovar sistema (conclusão)

Atividade	7 - Aprovar sistema
Procedimentos	O Cliente aprova o sistema e gera o (2.2.13 Termo de aceite de Software).
Saídas	Artefato gerado: (2.2.13 Termo de aceite de Software).

Fonte: Autor (2022).

2.2 TABELA DE ARTEFATOS

Visualizando a necessidade de uma listagem dos artefatos de software que necessitam ser gerados dentro do ESC, foram levantados os artefatos fundamentais e colocados em ordem dentro de sua respectiva fase de software e sob a responsabilidade de um ator específico.

A figura a seguir apresenta a implementação da tabela de artefatos, essa que foi produzida focando na fácil visualização dos artefatos, a descrição dos artefatos será apresentada a partir do título terciário 2.2.1.

Figura 5 — Tabela de Artefatos

	Fases de Software			
Responsável	**1º Engenharia de Requisitos**	**2º Modelagem**	**3º Codificação e Testes**	**4º Entrega**
Gerente de Projeto	Projeto de Software			
Engenheiro de Requisitos	Documento de Requisitos			
	Registros de reuniões			
Engenheiro de Software		Documento de Arquitetura de Software		
		Diagrama de Caso de Uso		
		Diagrama de Atividades		
		Diagrama de Classes		
		Diagrama de Domínio		
		Diagrama de Banco de dados		
		Diagramas opicionais, caso necessário		
Desenvolvedor			Código Fonte	
Analista de Testes			Plano de Testes	
			Relatório de Testes	
Responsável pela Entrega				Plano de Implantação
				Plano de Treinamento
				Manuais de Uso
				Documento de Implantação
Cliente				Termo de aceite de Software

Fonte: Autor (2022).

A seguir serão apresentados os objetivos de cada um dos artefatos além de enumerar pontos para se basear durante a confecção de cada um dos artefatos. Vale ressaltar que dependendo da necessidade do projeto ou critério do responsável pelo artefato, este pode conter um maior ou menor conteúdo.

2.2.1 **Projeto de Software**

O projeto de software tem por objetivo organizar e definir todo o processo de desenvolvimento, esse é um dos documentos mais importantes do software e deve ser construído como garantia de consulta futura e registro do processo de planejamento.

Este documento deve conter alguns pontos fundamentais que devem ser incluídos, entre eles:

1. **Controle de Versões:** Deve conter o nome do usuário que fez a alteração no documento e a respectiva data da alteração.

2. **Introdução/Alinhamento Estratégico:** Deve apresentar de forma resumida e direta o contexto, finalidade e escopo geral a fim de definir o propósito do projeto.
3. **Estrutura Organizacional:** Deve conter como será composta a equipe que estará envolvida na construção do sistema e suas respectivas responsabilidades.
4. **Cronograma:** Deve apresentar como será organizado as entregas do sistema e realização das atividades, o método de criação e organização do cronograma é de escolha do gerente do Projeto.
5. **Relatórios e Métricas:** Deve conter uma etapa com instruções sobre como devem ser os relatórios obtidos, principalmente durante a etapa de implementação e testes de software, além das métricas definidas para garantir a qualidade.
6. **Análise de Riscos:** Deve conter uma etapa focada no planejamento, gerenciamento e controle de riscos.
7. **Testes de usabilidade e aceitação:** Deve apresentar o planejamento dos testes com o usuário e o relatório dos mesmo para registro e comprovação de aceite do software.

2.2.2 **Documento de Requisitos**

O Documento de requisitos tem por objetivo principal delimitar o escopo das funcionalidades que o projeto visa alcançar.

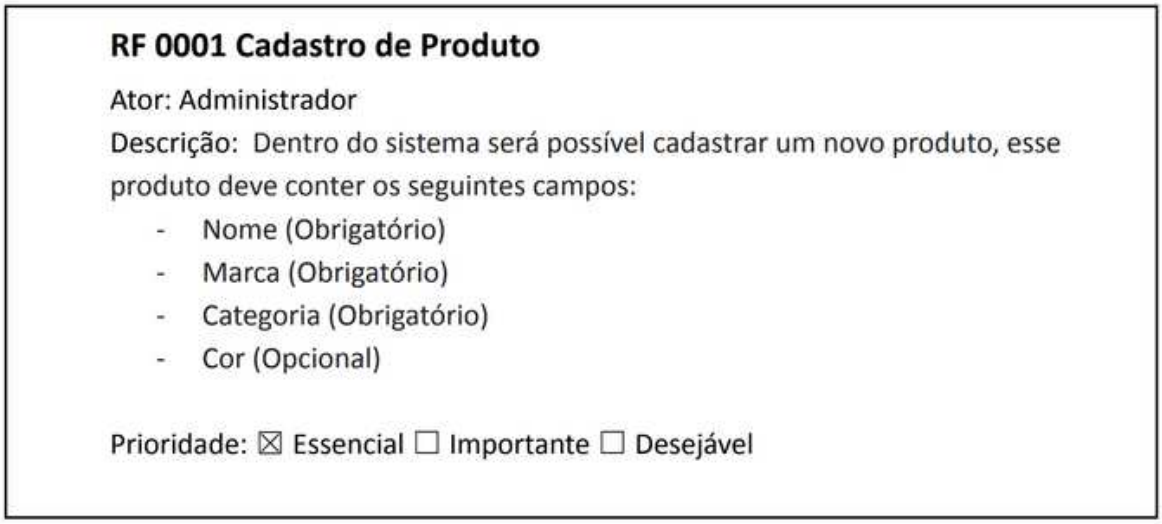
Este documento deve conter alguns pontos fundamentais que devem ser incluídos, entre eles:

1. **Controle de Versões:** Deve conter o nome do usuário que fez a alteração no documento e a respectiva data da alteração.
2. **Introdução/Apresentação:** Deve apresentar de forma resumida e direta o contexto e objetivo do presente documento.
3. **Listagem dos Requisitos Funcionais:** Deve apresentar todas as funcionalidades implementáveis. Para isso, é preciso que em cada requisito seja apresentado seu nome como título, o ator responsável por essa funcionalidade, sua descrição e sua prioridade, essa que pode ser "Essencial", "Importante" ou "Desejável".
4. **Listagem dos Requisitos Não-Funcionais:** Deve apresentar informações sobre como o sistema deve operar em termos de tecnologias envolvidas, desempenho, confiabilidade, usabilidade, segurança e etc.

Uma observação importante que deve ser atentada ao desenvolver o documento de requisitos é que a linguagem utilizada no na descrição dos requisitos deve ser simples e objetiva, visto que o documento será constantemente consultado pelos desenvolvedores e sua leitura não pode abrir margem para ambiguidade.

A seguir será apresentado um exemplo de requisito funcional.

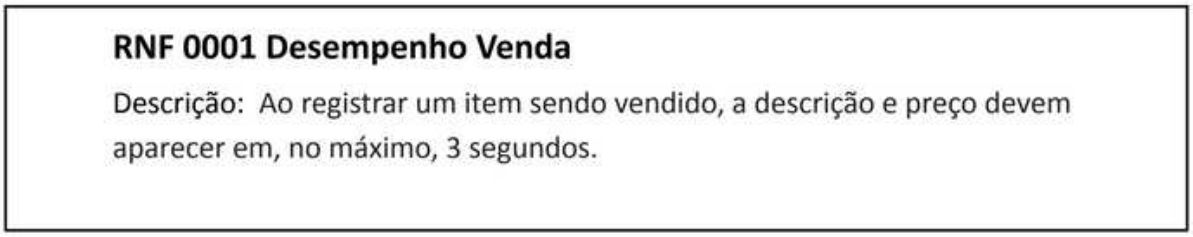
Figura 6 — Exemplo Requisito Funcional



Fonte: Autor (2022).

A seguir será apresentado um exemplo de requisito não-funcional.

Figura 7 — Exemplo Requisito Não-Funcional



Fonte: Autor (2022).

2.2.3 **Registros de reuniões**

O registro das reuniões tem por objetivo principal registrar decisões importantes, eventos, pendências, entre outros pontos relevantes.

Esse registro não necessita de um modelo já pré-definido, ele é utilizado de acordo com a decisão e necessidade do Engenheiro de Requisitos.

2.2.4 **Documento de Arquitetura de Software**

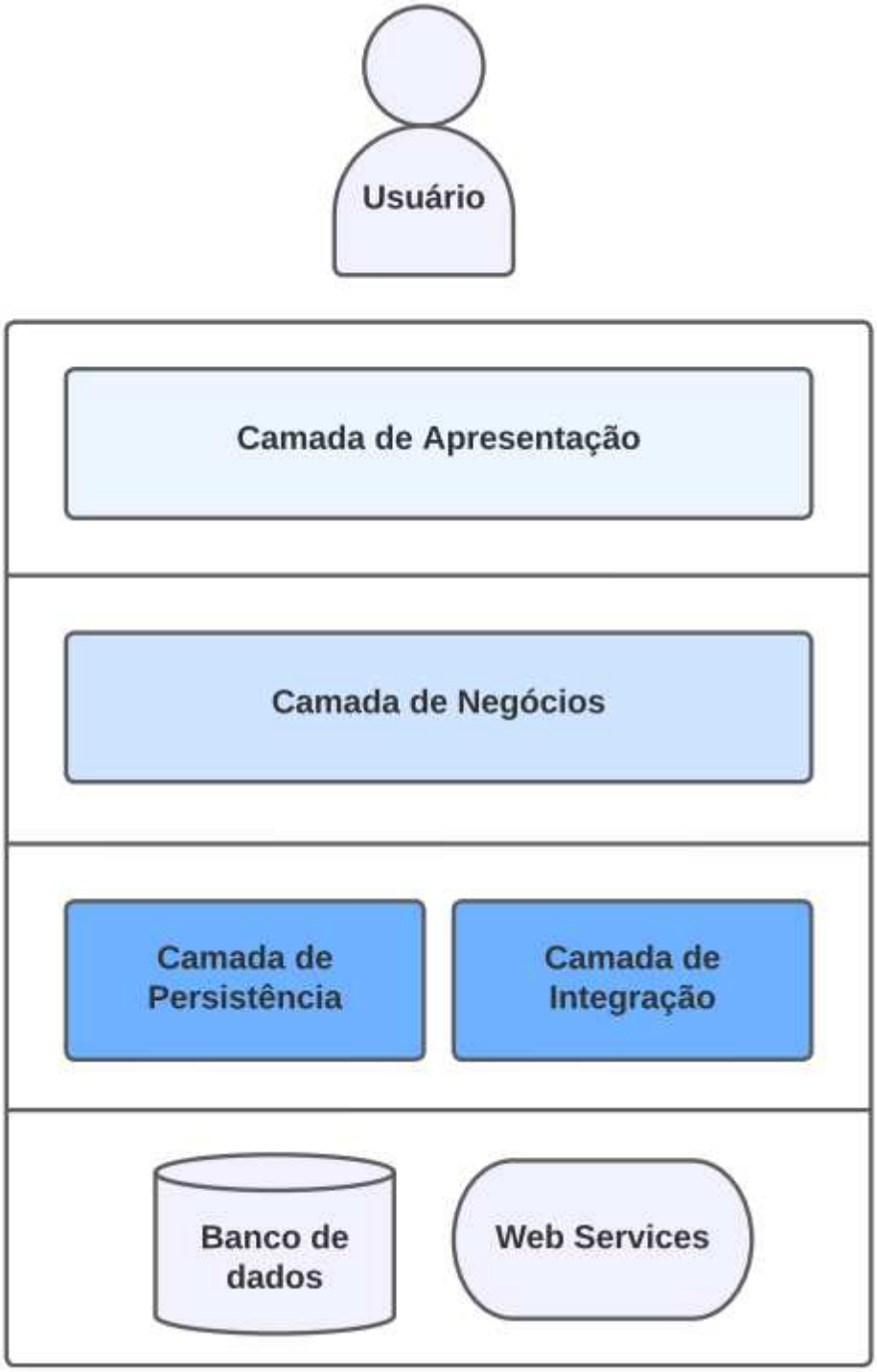
O Documento de Arquitetura de Software tem por objetivo definir estruturalmente como o sistema deve se comportar.

Este documento deve conter alguns pontos fundamentais que devem ser incluídos, entre eles:

1. **Controle de Versões:** Deve conter o nome do usuário que fez a alteração no documento e a respectiva data da alteração.
2. **Introdução/Apresentação:** Deve apresentar de forma resumida e direta o contexto e objetivo do presente documento.
3. **Visão Geral do Sistema:** Deve conter os elementos que compõem a arquitetura do software e o padrão arquitetural que o sistema irá adotar, exemplo o MVC, MVVM e etc. É interessante conter uma representação visual para melhor entendimento da equipe ou envolvidos com o projeto, como pode ser visto no exemplo da figura 8, apresentada posteriormente nessa seção.
4. **Requisitos e Restrições Arquiteturais:** Deve apresentar informações sobre como o sistema deve operar em termos de tecnologias envolvidas, desempenho, confiabilidade, usabilidade, segurança e etc.
5. **Qualidade:** Deve apresentar metas de qualidade, como por exemplo no reaproveitamento de código, facilidade de acréscimo de novas funcionalidades, manutenção e etc.

A seguir será apresentado um exemplo da visão geral da arquitetura.

Figura 8 — Exemplo Visão Geral da Arquitetura



Fonte: Autor (2022).

2.2.5 **Diagramas**

Os diagramas tem por objetivo auxiliar graficamente a modelagem do software. Com o uso dos diagramas, é possível validar os ideias do sistema com o cliente e auxiliar os desenvolvedores melhorando o entendimento sobre o sistema a ser construído.

Atualmente o direcionamento do ESC instrui que existem alguns diagramas que são obrigatórios serem modelados a cada novo projeto, obviamente dependendo da necessidade do projeto ou escolha do Engenheiro de Software essa diretriz pode mudar. Os diagramas obrigatórios atualmente são:

1. **Diagrama de Caso de uso.**
2. **Diagrama de Atividades.**
3. **Diagrama de Classes.**
4. **Diagrama de Domínio.**
5. **Diagrama de Banco de dados.**

2.2.6 **Código Fonte**

O código fonte tem por objetivo definir os comandos que compõem um software. Esse artefato engloba todo o trabalho referente a escrita de código produzido durante o período de desenvolvimento do sistema, sendo também um dos maiores e mais cruciais entre os artefatos. Focando nessa etapa como artefato de software, é importante instruir que o código fonte deve ser não apenas escrito mas registrado e gerenciado.

Como forma de registro para futuras consultas, backups e/ou mudanças, o registro do código deve ser feito utilizando um versionador de código, prática que tem o intuito de gerenciar e controlar todas as versões do código fonte durante o período de desenvolvimento do sistema.

Por padrão, alguns pontos foram definidos até o momento como chaves para serem atentados durante a confecção desse artefato, são esses:

1. **Condizente à documentação:** O sistema deve ser implementado de acordo com as diretrizes especificadas na documentação do sistema.
2. **Testado antes do envio:** O código deve passar por uma etapa de teste unitário ou outro tipo de teste já pré-definido e especificado no plano de testes antes de ser disponibilizado no versionador.
3. **Disponibilizado no versionador:** O código deve ser disponibilizado regularmente no versionador de código já pré-definido pela equipe.

5. **Revisado:** A revisão de código deve ser feita em conjunto com outro desenvolvedor(a) da equipe, essa prática necessita da revisão de outro desenvolvedor(a) por este(a) estar mais hábil a encontrar possíveis bugs, erros ou práticas inadequadas que possam comprometer a qualidade do software, além de compartilhar conhecimento.

2.2.7 **Plano de Testes**

O plano de testes tem por objetivo definir a organização das atividades de testes a serem realizadas durante o projeto. Esse artefato documenta pontos fundamentais em relação ao cenário de testes do projeto, entre eles:

1. **Controle de Versões:** Deve conter o nome do usuário que fez a alteração no documento e a respectiva data da alteração.
2. **Introdução/Apresentação:** Deve apresentar de forma resumida e direta o contexto e objetivo do presente documento.
3. **Equipe de teste:** Deve conter a descrição da equipe composta para as atividades de teste e seus respectivos papéis.
4. **Instruções para revisão de código:** Deve conter um checklist de pontos a serem analisados durante a atividade de code review por um revisor de código.
5. **Ferramentas:** Deve apresentar todas as ferramentas que serão utilizadas para realização das atividades .
6. **Tipos de testes e Métodos:** Deve conter os tipos de testes, técnicas e procedimentos que serão utilizados, de forma detalhada e técnica.
7. **Critérios de aceitação:** Deve conter a descrição de como deve ser realizada a análise dos resultados e aceitação ou não dos resultados de acordo com limites que serão definidos.
8. **Cronograma:** Deve conter apenas marcos e prazos importantes que devem ser definidos, além de uma estimativa de duração das atividades.

2.2.8 **Relatório de Testes**

O relatório de testes tem por objetivo construir uma relação das atividades realizadas e resultados obtidos durante a etapa de testes de software.

Geralmente haverá a necessidade de mais de um documento de relatório de testes. Este documento deve conter alguns pontos fundamentais que devem ser incluídos, entre eles:

1. **Responsável pelo teste:** Deve conter o nome do usuário que fez o teste e a respectiva data de realização.
2. **Introdução/Apresentação:** Deve apresentar de forma resumida e direta o contexto e objetivo do presente documento.
3. **Cobertura do teste:** Deve conter qual parte do código ou funcionalidade esse teste contemplou.
4. **Critérios de aceitação e Análise:** Deve conter um tópico sobre as medidas de aceitação e a análise dos resultados obtidos, nessa seção pode ter uma representação visual com gráficos ou diagramas para representar os resultados caso necessário.
5. **Conclusão:** Deve apresentar se o critério de aceitação foi aprovado ou reprovado, e caso necessário, sugerir ações recomendadas para correção.

2.2.9 **Plano de Implantação**

O Plano de Implantação tem por objetivo organizar as atividades relacionadas à integração do sistema criado na organização do cliente.

Esse planejamento deve conter alguns pontos de atuação para reduzir os riscos e contratempos no futuro, exemplo:

1. **Análise da organização:** Deve ser feita uma pesquisa sobre a cultura do local de implantação, juntamente com fatores como conexão e configuração de máquinas caso estes fatores sejam relevantes.
2. **Impactos:** Deve ser feito um estudo sobre os possíveis impactos que uma possível paralisação das atividades no ambiente de trabalho pode ocasionar, e como remediá-los.
3. **Equipe e Atividades:** Deve ser definida as pessoas que participarão do processo de implantação e suas respectivas funções.
4. **Cronograma:** Deve ser planejado um cronograma de execução das atividades em concordância com a disponibilidade do cliente.

2.2.10 **Plano de Treinamento**

O Plano de Implantação tem por objetivo planejar como serão realizadas as atividades referentes ao treinamento dos usuários ao sistema desenvolvido.

Esse plano de treinamento deve conter alguns pontos fundamentais para sua execução, são eles:

1. **Definir objetivos e Público alvo:** Deve ser estabelecido os objetivos do treinamento e o que se busca alcançar com o mesmo, além do público alvo que receberá o treinamento.
2. **Modalidade:** Deve ser definida a modalidade do treinamento, por exemplo: online, presencial ou gravação, e em caso de presencial o local que ocorrerá.
3. **Metodologia:** Deve ser definido a metodologia de ensino e a seleção dos instrutores/capacitadores.
4. **Material de apoio/Manuais de uso:** Deve ser analisado como será a composição do material auxiliador dos usuários, bem como quem será responsável pela criação dos mesmos.

2.2.11 **Manuais de Uso**

O(s) manual(is) de uso tem por objetivo auxiliar os atuais e futuros usuários a utilizarem o sistema desenvolvido.

A quantidade de manuais de uso depende da necessidade e tamanho do software, caso haja uma grande quantidade de instruções a serem transmitidas, ou diversos tipos de usuários com diferentes visões do sistema, então pode ser desenvolvido mais de um manual de uso.

Não existe uma fórmula definida para criar um manual de uso adequado, pois depende do objetivo do manual e do sistema. Entretanto, existem algumas diretrizes fundamentais que podem servir como base para a criação de um bom manual, são elas:

1. **Índice:** O índice tem fundamental importância, pois auxilia o usuário a encontrar com maior facilidade determinados temas ou instruções dentro do manual.
2. **Organização do material:** Pontos chave como separação por tópicos, títulos auto explicativos, fontes legíveis e leitura clara são fundamentais para uma satisfatória leitura do usuário.
3. **Imagens de apoio:** Elementos visuais são meios transparentes de transmitir informações para o usuário.

2.2.12 **Documento de Implantação**

O Documento de Implantação tem por objetivo registrar o que foi realizado durante as atividades de implantação.

Esse documento deve conter alguns pontos fundamentais, são eles:

1. **Apresentação/Objetivo do Documento:** Deve ser apresentado o contexto e objetivo do documento.
2. **Relatório das atividades:** Deve ser registrado um relatório das atividades realizadas durante o processo de implantação, incluindo o treinamento e entrega dos manuais.
3. **Registro de data e assinaturas:** Deve ser coletado a respectiva data e assinaturas dos responsáveis tanto pela equipe de implantação quanto os responsáveis pela organização.

2.2.13 **Termo de aceite de Software**

O Termo de aceite de Software tem por objetivo garantir que o cliente está declarando conhecimento que recebeu o software, assim como o responsável pelo mesmo está ciente que o entrega.

É importante conter alguns pontos chave dentro do termo de aceite, entre eles:

1. **Identificação das partes:** Devem ser identificadas as partes interessadas, no caso quem recebe o sistema e quem o disponibiliza, e suas respectivas assinaturas.
2. **Data do aceite:** Deve conter a respectiva data no documento.
3. **Regras do termo:** Deve haver um detalhamento do termo de aceite, como exemplo a autorização de uso de imagens e concordância de uso de serviços.

APÊNDICE B — GUIA DE ELEMENTOS BPMN

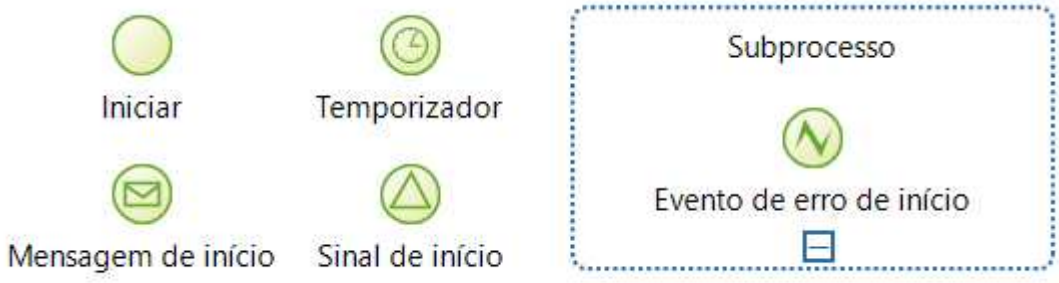
5.1 IDENTIFICAÇÃO DOS ELEMENTOS BPMN

Para auxiliar a compreensão e entendimento do fluxo de atividades, é fundamental a apresentação e explicação dos elementos BPMN à disposição. Para isso, serão apresentados alguns dos elementos disponíveis dentro da ferramenta Bonita Studio e Bizagi Modeler, os elementos escolhidos para identificar nessa seção serão apenas os grupos que tiveram pelo menos um elemento utilizado neste trabalho.

5.2 EVENTOS DE INÍCIO

Segundo o guia (THE ULTIMATE GUIDE TO BPMN2, 2021), os eventos de início, também chamados de start events na ferramenta Bonitasoft, são essenciais para na modelagem BPMN pois são eventos necessários para iniciar um fluxo. A seguir serão apresentados alguns eventos de início e posteriormente sua função dentro de um fluxo.

Figura 16 — Eventos de Início



Fonte: BonitaSoft (2022).

- Iniciar:
- Cria um novo evento de início rápido, sem nenhum fato particular ao início do processo.
- Mensagem de início: Cria um novo evento com mensagem de início.
- Temporizador: Cria um novo evento de início com uma condição de tempo, sendo essa uma data relativa ou período.
- Sinal de início: Cria um novo evento de sinal de início quando um sinal vindo de outro processo é capturado.
- Evento de erro de início: Cria um evento erro de início. Só pode ser usado

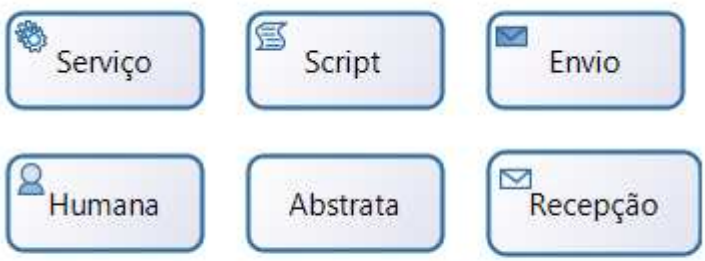
em um evento de subprocesso.

Dentro da modelagem dos processos feita, apenas o "Iniciar" foi utilizado, visto que os outros não foram necessários para o fluxo.

5.3 TAREFAS

As tarefas dentro da modelagem são ações que podem ser realizadas por um ser humano ou um sistema, sempre tendo como propósito ter o menor grau de granularidade possível para a obtenção de maiores detalhes, segundo o guia (GUIA BPMN 2.0 IPROCESS, 2014).

Figura 17 — Tarefas



Fonte: BonitaSoft (2022).

- Tarefa de serviço: Cria uma nova tarefa de serviço, o qual é um passo automatizado.
- Tarefa humana: Cria uma nova tarefa humana, o qual é um passo que deve ser realizado por uma pessoa.
- Tarefa de script: Cria uma nova tarefa de execução de script.
- Tarefa abstrata: Cria uma nova tarefa abstrata, que não possui nenhum tipo de especificação. Geralmente usada em tarefas ainda não definidas ou em situações onde a tipificação não é relevante.
- Tarefa de envio: Cria uma nova tarefa de envio de mensagem, é enviada uma mensagem a um participante.
- Tarefa de recepção: Cria um novo passo de recepção de mensagem, é recebida uma mensagem de um participante.

Dentro da modelagem dos processos feita, foram utilizadas "tarefa humana", "tarefa de envio" e "tarefa abstrata".

5.4 ATIVIDADES

De acordo com o guia (GUIA BPMN 2.0 IPROCESS, 2014), as atividades são responsáveis por retratar a abstração de uma coleção lógica de atividades com uma finalidade específica.

Figura 18 — Atividades

Fonte: BonitaSoft (2022).

- Atividade de chamada: Cria uma nova atividade de chamada, essa que é um subprocesso, entretanto não havendo detalhamento das atividades dentro do subprocesso.
- Evento de subprocesso: Cria um novo subprocesso de evento, esse por sua vez detalha o fluxo de atividades necessárias para ser concluído.

Dentro da modelagem dos processos feita, apenas a "atividade de chamada" foi utilizada, visto que foi a única necessária esse fluxo.

5.5 EVENTOS DE FIM

De acordo com o guia (THE ULTIMATE GUIDE TO BPMN2, 2021), os eventos de fim são aqueles responsáveis por finalizar um fluxo de processos anteriormente já iniciado.

Figura 19 — Eventos de Fim

Fonte: BonitaSoft (2022).

- Fim: Cria um novo evento de fim, encerrando um fluxo.
- Erro de fim: Cria um novo evento de erro de fim, finalizando um fluxo com uma falha sistêmica.
- Mensagem de fim: Cria um novo evento de mensagem de fim, enviando uma mensagem a outro processo quando o fluxo terminar.
- Evento de término: Cria um novo evento de término, esse tipo de fim indica que todas as atividades no processo devem ser paradas imediatamente.
- Sinal de fim: Cria um novo evento de sinal de fim, o fluxo termina enviando um sinal broadcast para outros processos.

Dentro da modelagem dos processos feita, foram utilizados "fim" e "evento de término".

5.6 ENTRADAS

Os Gateways, ou entradas, são elementos utilizados para separar ou juntar o fluxo do processo, segundo o guia (THE ULTIMATE GUIDE TO BPMN2, 2021).

Figura 20 — Gateways

Fonte: BonitaSoft (2022).

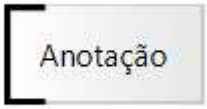
- Desvio: Cria um novo desvio paralelo, é usado para representar duas tarefas simultâneas.
- Desvio exclusivo: Cria um novo desvio exclusivo, é possível continuar o fluxo por apenas uma saída e essa é condicional.
- Desvio inclusivo: Cria um novo desvio inclusivo, possibilita a divisão do fluxo único em um ou mais de forma condicional.

Dentro da modelagem dos processos feita, foram utilizados "desvio" e "desvio exclusivo".

5.7 ANOTAÇÃO DE TEXTO

As anotações são utilizadas para adicionar notas complementares ao diagrama para melhor entendimento de determinado processso ou elemento (GUIA BPMN 2.0 IPROCESS, 2014).

Figura 21 — Anotação de Texto



Fonte: BonitaSoft (2022).

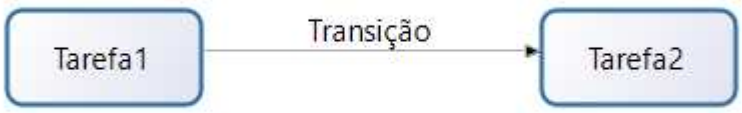
- Anotação: Cria uma nova anotação de texto para adicionar notas complementares.

Dentro da modelagem dos processos feita, as anotações foram usadas.

5.8 TRANSIÇÃO

A transição mostra a sequência em que as atividades são executadas dentro do processo (GUIA BPMN 2.0 IPROCESS, 2014).

Figura 22 — Transição



Fonte: BonitaSoft (2022).

- Transição: Cria uma nova transição mostrando a direção do fluxo.

Dentro da modelagem dos processos feita, as transições foram utilizadas.

5.9 LOOP

O loop é usado para indicar que determinada tarefa será repetida mais de uma vez, geralmente utilizada em tarefas mais complexas que necessitam ser feitas e refeitas até chegar a um estado satisfatório (THE ULTIMATE GUIDE TO BPMN2, 2021).

Figura 23 — Loop

Fonte: Bizagi (2022).

- Loop: Cria uma tarefa de repetição, essa tarefa será realizada mais de uma vez.

Dentro da modelagem dos processos feita, os loops foram utilizados.

5.10 ARTEFATO DE SOFTWARE

Os artefatos representam unidades físicas de execução, podem ser subprodutos gerados dentro do processo de software como: bibliotecas, componentes de software, documentos, entre outros (RATIONAL SOFTWARE ARCHITECT STANDARD EDITION 7.5.0, 2021).

Figura 24 — Artefato de Software

Fonte: Bizagi (2022).

- Artefato: Cria um novo artefato, é usado para representar um produto gerado durante um processo.

Dentro da modelagem dos processos feita, os artefatos de software foram utilizados.