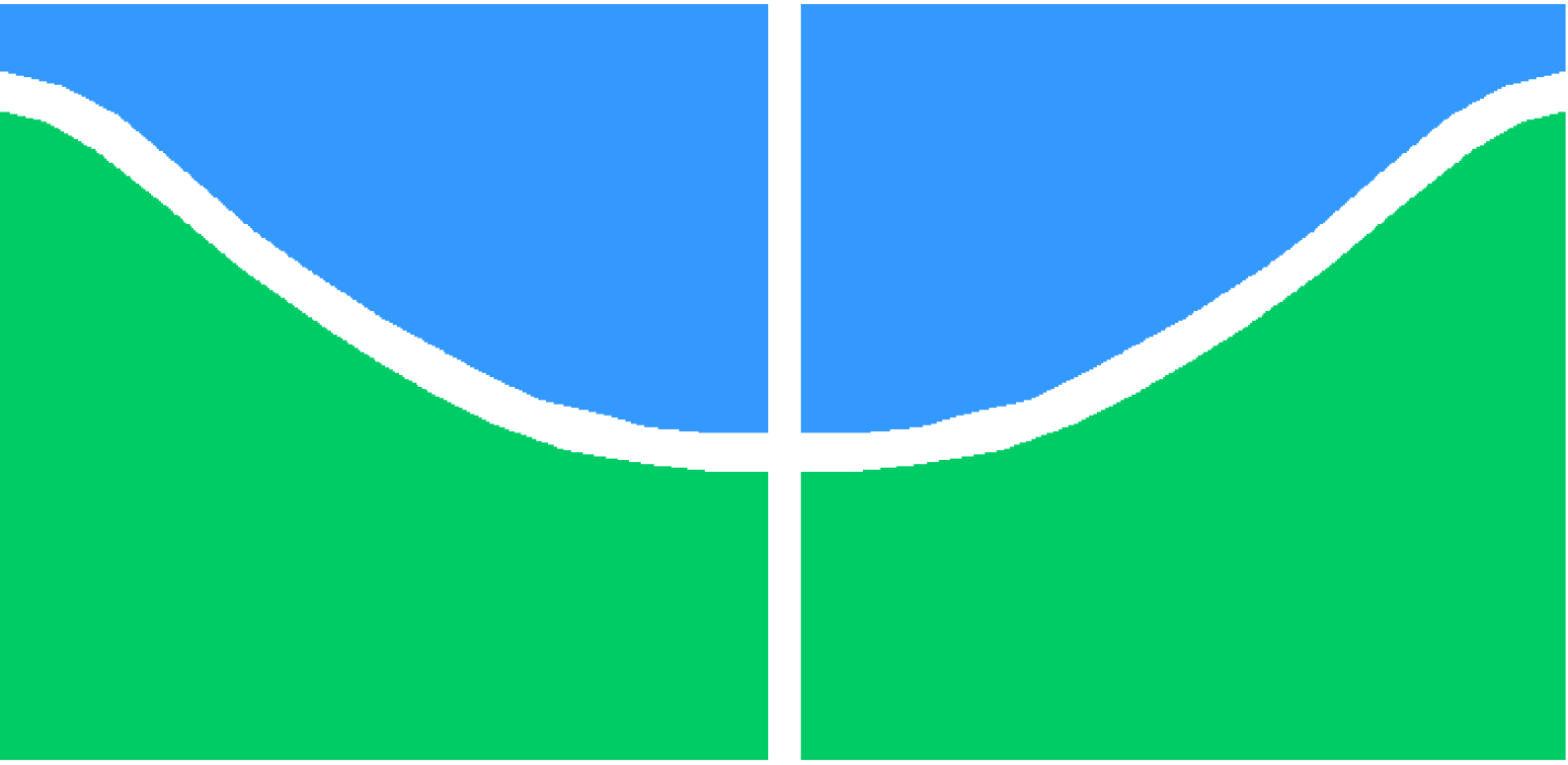

Universidade de Brasília - UnB
Faculdade UnB Gama - FGA
Engenharia de Software

Elaboração de estratégia de testes em equipe de voluntários geograficamente distribuída

Autor: Vítor Cardoso Xoteslem
Orientador: Dra. Fabiana Freitas Mendes

Brasília, DF
2021

Vítor Cardoso Xoteslem

Elaboração de estratégia de testes em equipe de voluntários geograficamente distribuída

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Universidade de Brasília - UnB

Faculdade UnB Gama - FGA

Orientador: Dra. Fabiana Freitas Mendes

Brasília, DF

2021

Vítor Cardoso Xoteslem
Elaboração de estratégia de testes em equipe de voluntários geograficamente distribuída/ Vítor Cardoso Xoteslem. – Brasília, DF, 2021-
94 p. : il. (algumas color.) ; 30 cm.

Orientador: Dra. Fabiana Freitas Mendes

Trabalho de Conclusão de Curso – Universidade de Brasília - UnB
Faculdade UnB Gama - FGA , 2021.

1. Testes. 2. Equipes. I. Dra. Fabiana Freitas Mendes. II. Universidade de Brasília. III. Faculdade UnB Gama. IV. Elaboração de estratégia de testes em equipe de voluntários geograficamente distribuída

CDU

Vítor Cardoso Xoteslem

Elaboração de estratégia de testes em equipe de voluntários geograficamente distribuída

Monografia submetida ao curso de graduação em Engenharia de Software da Universidade de Brasília, como requisito parcial para obtenção do Título de Bacharel em Engenharia de Software.

Trabalho aprovado. Brasília, DF, 25 de outubro de 2021:

Dra. Fabiana Freitas Mendes
Orientador

Dr. George Marsicano Corrêa
Convidado 1

Dra. Elaine Venson
Convidado 2

Brasília, DF
2021

Agradecimentos

Agradeço primeiramente a toda minha família, que sempre me apoiou incondicionalmente ao longo de minha vida, e foi essencial na formação dos valores nos quais acredito atualmente. Agradeço especialmente aos meus pais, Luciana e Wesley, pelo amor, carinho, suporte e apoio dado a mim até hoje.

Agradeço aos meus amigos, que sempre ao meu lado, tornaram essa jornada até aqui muito mais leve e agradável. Agradeço também, em especial, ao meu amor, Dinna, companheira indispensável, que me mostra todos os dias que sou capaz de amar de uma forma que nunca achei ser possível.

Por fim, e não menos importante, agradeço aos professores que fizeram parte da minha jornada na Universidade de Brasília, em especial a professora Fabiana Freitas, que me orientou, com muita paciência e disposição, no desenvolvimento deste trabalho.

Resumo

O Doarti é um aplicativo que aproxima as instituições beneficentes e doadores. A equipe de desenvolvimento do aplicativo é composta por voluntários, trabalhando de forma assíncrona e geograficamente distribuída, sobretudo por conta das medidas de segurança relativas ao enfrentamento à pandemia de COVID-19. Nesse contexto, o trabalho aqui apresentado tem o objetivo de elaborar e validar uma estratégia de testes no Doarti. Para isso, foi realizada uma revisão da literatura relacionada a equipes, desenvolvimento, testes e medição de software e avaliação de processos. Foi conduzida uma avaliação informal do processo de verificação e validação do Doarti, seguindo princípios do MPS.BR (Melhoria do Processo de Software Brasileiro) e CMMI (*Capability Maturity Model Integration* - Modelo Integrado de Capacidade e Maturidade), que apontou algumas oportunidades de melhoria no processo de verificação de validação do Doarti. A partir dos resultados da avaliação informal, foi possível definir áreas de foco a serem exploradas pela estratégia desenvolvida. Em seguida, foi realizada uma validação sobre a qualidade da estratégia proposta, com a participação de membros do Doarti, concluindo que a estratégia desenvolvida foi bem compreendida apesar de terem sido apontadas algumas oportunidades de melhoria.

Palavras-chave: Testes de Software. Melhoria de Processos de Software.

Abstract

Doarti is an application that brings together charities and donors. The application development team is made up of volunteers, working asynchronously and geographically distributed, mainly because of the security measures related to the fight against the COVID-19 pandemic. In this context, the work presented here aims to develop and validate a testing strategy in Doarti. For this, a review of the literature related to teams, software development, testing and measurement and process evaluation was carried out. An informal assessment of the Doarti verification and validation process was conducted, following the principles of the MPS.BR (Melhoria do Processo de Software Brasileiro - Brazilian Software Process Improvement) and CMMI (Capability Maturity Model Integration), which pointed out some opportunities for improvement in the Doarti validation verification process. From the results of the informal assessment, it was possible to define focus areas to be explored by the developed strategy. Afterwards, a validation was carried out on the quality of the proposed strategy, with the participation of Doarti members, concluding that the developed strategy was well understood, despite some opportunities for improvement having been pointed out.

Key-words: *Software test. Software process improvement.*

Lista de ilustrações

Lista de tabelas

Lista de abreviaturas e siglas

BID/FUMIN	Fundo Multilateral de Investimentos do Banco Interamericano de Desenvolvimento.
CETT	Conhecimento da Equipe relacionado a Testes e às Tecnologias utilizadas no projeto.
CIT	Complexidade das Implementações Testadas.
CMMI	Capability Maturity Model Integration - Modelo Integrado de Capacidade e Maturidade.
CNPq	Conselho Nacional de Desenvolvimento Científico e Tecnológico.
Cowtest	Cost Weighted Test Strategy - Estratégia de Teste de Custo Ponderado.
Cow_Suite	Cow pluS UIT Environment - Ambiente Cow mais UIT.
CP-G	Capacidade de Processo Nível G.
ED	Evidência Direta.
EI	Evidência Indireta.
FINEP	Financiadora de Estudos e Projetos.
FVED	Falhas Validadas pela Equipe de Desenvolvimento.
GQM	Goal, Question, Metric - Objetivo, Questão, Métrica.
IEC	International Electrotechnical Commission - Comissão Eletrotécnica Internacional.
ISO	International Organization of Standardization - Organização Internacional de Padronização.
MCTIC	Ministério da Ciência, Tecnologia, Inovações e Comunicações.
MPS.BR	Melhoria do Processo de Software Brasileiro.
MR-MPS-SW	Modelo de Referência do MPS.BR para software.
NCT	Nível de Conhecimento do Testador em relação à estratégia de testes aplicada no Doarti.
OE	Objetivo Específico.

ORD	Object Relation Diagram - Diagrama de Relação de Objeto.
PCT	Porcentagem de Código Testado.
PFA	Porcentagem de Falhas Aceitas.
SEBRAE	Serviço Brasileiro de Apoio a Micro e Pequenas Empresas.
SEI	Software Engineering Institute - Instituto de Engenharia de Software.
SOFTEX	Sociedade para Promoção da Excelência do Software Brasileiro.
SPICE	Software Process Improvement and Capability Determination - Melhoria e Determinação de Capacidade de Processos de Software.
SQwaRE	Software product Quality Requirements and Evaluation - Requisitos e Avaliação da Qualidade de produtos de Software.
TDD	Test Driven Development - Desenvolvimento orientado por testes.
TFRS	Total de Falhas Reportadas na Sprint por um Único Testador Específico.
TMMi	Test Maturity Model integration - Modelo Integrado de Maturidade de Testes.
UIT	Use Interaction Test - Teste de Interação de Uso.
VFET	Validade de Falhas Encontradas por Testador.
VV	Verificação e Validação.

Sumário

1 Introdução

O celular tem se tornado o principal meio de acesso à internet no Brasil (BRASIL, 2020). Como consequência, percebe-se a ascensão do uso de dispositivos móveis como um novo meio de relacionamento entre organizações e clientes, seja para realização de propagandas seja para prestação de serviços (ANDRADE; AGRA; MALHEIROS, 2013). Esse cenário mostra o quão oportuno é o momento atual para realização de ações sociais de impacto positivo através de aplicativos móveis, como é o caso do Doarti.

O desenvolvimento desses aplicativos exige atenção especial em diversos aspectos, tais como: tamanho da tela, métodos de entrada de dados, alta latência e baixa largura de banda (HARTMANN; STEAD; DEGANI, 2011). Esses aspectos são da natureza dos dispositivos móveis e podem ser diferentes dependendo do fornecedor (HARTMANN; STEAD; DEGANI, 2011).

Ao considerar os fatores supracitados, é possível notar a importância da adoção de práticas conhecidas na engenharia de software para que se garanta qualidade no produto final. Uma dessas práticas é a realização de testes de software, que ajudam a aumentar a qualidade do software (SOMMERVILLE, 2019).

A motivação deste trabalho é a melhoria da qualidade do aplicativo Doarti, desenvolvido por uma equipe composta por voluntários trabalhando de forma remota e assíncrona. É necessária uma atenção especial a esses aspectos específicos da equipe, de modo a auxiliar na busca por estratégias de melhoria no gerenciamento que, consequentemente, podem impactar de forma positiva na qualidade final do produto (WONG; BURTON, 2000).

1.1 O Projeto Doarti

O Doarti, principal objeto e ambiente de estudo deste trabalho, consiste num projeto de extensão da Universidade de Brasília, de responsabilidade da Fábrica de Software da Faculdade do Gama, um grupo de pequisa certificado no Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq) pela Universidade de Brasília. O projeto tem um tempo de execução previsto para 24 meses, sendo este período de março de 2020 à março de 2022 (BRASíLIA, 2020).

1.1.1 Motivação

O projeto surgiu no contexto do agravamaento da pandemia de COVID-19, onde houve aumento no desemprego e vulnerabilidade na população, provocado em parte pela

adoção de medidas de isolamento social, tomadas para frear o avanço da doença (BRASíLIA, 2020). Junto com esses problemas surgiram ações para coleta e distribuição de doações, porem, com volume de doações reduzido em cerca de 80%, também por conta do isolamento social (BRASíLIA, 2020).

1.1.2 Solução

O Doarti foi criado com a proposta de identificar, de forma centralizada, as ações de doação ocorridas em Brasília, reunindo em um único local informações sobre doadores e entidades, facilitando o contato entre ambos (BRASíLIA, 2020).

De acordo com a documentação interna do projeto, a solução Doarti consiste num sistema voltado tanto para as entidades beneficentes quanto para os doadores, sendo composto por um site e um aplicativo móvel. O site tem como funções a gerência de entidades beneficentes e a disponibilização de informações acerca do projeto. Ao demonstrar intenção de participação, entidades beneficentes preenchem um formulário com seus dados, que são validados pela equipe responsável no Doarti. Caso os dados da entidade sejam válidos, ela passa para fase de confirmação da criação de seu usuário no sistema do Doarti, caso contrário, receberá um comunicado do Doarti informado a situação. O fluxo detalhado de intenção de participação das entidades, atividade realizada através do site, é representado pela Figura 10 apresentada no Anexo A.

Já o aplicativo móvel se destina tanto as entidades quanto aos doadores, e é a ferramenta responsável por estabelecer o relacionamento entre ambos. Para realizar uma doação, o doador deve, através do aplicativo móvel, informar se pode entregar ou se precisa que busquem a doação. Caso precise que a entidade busque a doação, o endereço e demais detalhes da entrega são combinados, caso contrário, o próximo passo já é a entrega da doação para a entidade. Por fim, a doação é confirmada e o doador recebe uma mensagem de agradecimento. O fluxo de doações, realizadas através do aplicativo móvel é representado pela Figura 11 apresentada no Anexo A.

É importante ressaltar, que o escopo deste trabalho está limitado apenas a equipe responsável pelo desenvolvimento e manutenção do aplicativo móvel.

1.2 Objetivo geral

Este trabalho tem como objetivo elaborar e validar uma estratégia de testes a ser implantada em uma equipe de desenvolvedores voluntários, trabalhando de forma assíncrona e geograficamente distribuída. O resultado principal é uma estratégia de testes robusta e detalhada, que seja adequada à realidade da equipe do Doarti. Para alcançar esse objetivo foram propostos os objetivos específicos, conforme apresentados na Tabela 1.

Tabela 1 – Objetivos específicos do trabalho

ID	Objetivo	Motivação
OE1	Entender o contexto do projeto Doarti.	Identificar aspectos do projeto que possuam impacto significante no processo de testes.
OE2	Desenvolver uma estratégia de testes para o Doarti.	Planejar melhorias significativas no processo de testes do projeto.
OE3	Validar estratégia de testes definida.	Concluir sobre a qualidade, e adequação da estratégia de testes desenvolvida.

1.3 Estrutura do trabalho

A estrutura deste trabalho é composta por este capítulo no qual foram apresentados contexto, motivação e objetivos da pesquisa. Logo em seguida, é apresentada a metodologia da pesquisa, com detalhes de cada uma das fases.

No desenvolvimento, a partir do Capítulo 3, foi exposto o referencial teórico, passando por temas relacionados a equipes de desenvolvimento de software, qualidade de software, metodologias ágeis, GQM (Goal, Question, Metric - Objetivo, Questão, Métrica) e MPS.BR (Melhoria do Processo de Software Brasileiro). Cada seção desse capítulo apresenta fundamentos teóricos referentes ao respectivo tema, bem como uma contextualização, relacionando esse tema ao Doarti, projeto alvo do presente trabalho.

Posteriormente, nos Capítulos 4 e 5, são apresentados os resultados, incluindo o conjunto de métricas selecionadas, a avaliação inicial realizada sobre o processo de verificação e validação do Doarti, a estratégia de testes elaborada e sua validação.

Por fim, o Capítulo 6 com as considerações finais, retomando o objetivo do trabalho e discutindo os resultados alcançados e trabalhos futuros.

2 Metodologia de pesquisa

Este capítulo descreve a metodologia de pesquisa utilizada no trabalho, a qual foi dividida em quatro fases: contextualização, coleta inicial de dados, elaboração da estratégia de teste e validação da estratégia de testes.

2.1 Fases

Foram definidas quatro fases, conforme ilustra a Figura 1. Observe que as fases foram dispostas em ordem cronológica na figura.

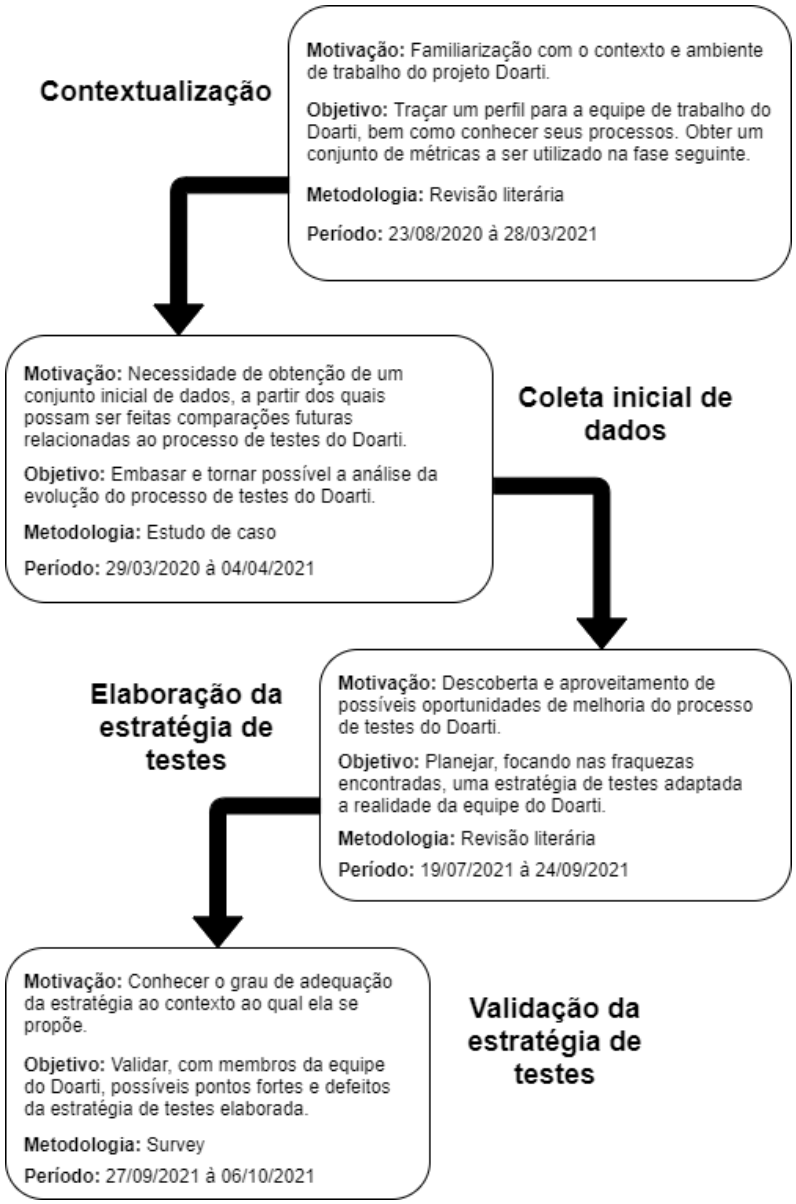

Figura 1 – Fases do trabalho

As fases de contextualização e coleta inicial de dados serviram como base para elaboração da estratégia de testes que, por sua vez, foi validada ao final do trabalho, durante a fase de validação da estratégia.

Um cronograma detalhado das atividades executadas ao longo do trabalho pode ser encontrado no Apêndice A. Nas subseções seguintes são detalhadas as fases descritas na Figura 1.

2.1.1 Contextualização

O escopo de pesquisa do trabalho concentra-se nos temas relacionados a equipes de desenvolvimento de software, dinâmicas de trabalho voluntário, remoto e assíncrono. Também são foco da pesquisa as definições e práticas conhecidas na área da engenharia de software relacionadas a avaliação, implementação e otimização de estratégias para testes de software.

A partir disso, foi feito o planejamento da revisão de literatura, na qual foram definidas as fontes, questões de pesquisa. Os objetivos e tópicos explorados da revisão de literatura são apresentados na Tabela 2.

Tabela 2 – Objetivo e tópicos de pesquisa

Objetivo	Identificar trabalhos sobre abordagens de testes no contexto de equipes voluntárias, geograficamente distribuídas, trabalhando de forma assíncrona.
Tópicos de pesquisa	- Características de qualidade de produto de software - Melhoria de Processos de Software - Abordagens de testes - Trabalho remoto e assíncrono - Equipes voluntárias

Em relação às fontes de pesquisa, foi realizada uma busca não sistemática no Google Acadêmico, através da qual foram selecionados documentos e estudos relevantes ligados aos tópicos de pesquisa. Tópicos como: Melhoria de processos de software, trabalho remoto e assíncrono e equipes voluntárias foram importantes para o alcance do objetivo específico OE1 e OE2, pois auxiliaram no entendimento do contexto do Doarti. Os tópicos Abordagem de testes e Características de qualidade de produto de software, auxiliaram no alcance do objetivo OE2 e OE3, estabelecendo uma base teórica para elaboração da estratégia de testes e para preparação da validação da estratégia.

Nessa fase, foram estudados dos artefatos produzidos pela equipe do Doarti. Além disso, foi realizada a ambientação com o modo de trabalho através da realização dos testes do código produzido pela equipe.

O estudo dos artefatos e a ambientação, juntamente ao estudo da literatura relaci-

onada viabilizaram a execução das próximas fases dessa pesquisa. A Seção 2.1.2 apresenta detalhes dos passos executados na fase de coleta inicial de dados.

2.1.2 Coleta inicial de dados

Durante a fase de coleta inicial de dados foram analisados os indicadores e artefatos relacionados ao Doarti. Esse material foi utilizado para realizar a avaliação do processo Verificação & Validação do MPS.BR. Essa avaliação inicial teve com o objetivo de conhecer, em um primeiro momento, o nível de maturidade e capacidade, expondo possíveis fraquezas no processo de testes atual do Doarti. O resultado da avaliação é apresentado no Capítulo 4. Demais informações sobre o MPS.BR e seus níveis de maturidade e capacidade de processos estão descritas no Capítulo 3.

2.1.3 Elaboração da estratégia de testes

Após a realização de uma avaliação inicial do processo de testes do Doarti, foi desenvolvida uma estratégia de testes. Para tanto, foi considerado o conhecimento obtido durante a revisão de literatura, principalmente as relacionadas a testes e estratégias de testes apresentadas no Capítulo 3. Também foram consideradas as evidências coletadas durante a avaliação informal do processo de verificação e validação.

O foco principal da estratégia de testes é que ela seja sustentável e capaz de lidar com os riscos relacionados às características da equipe e aos processos do Doarti. Foram definidos procedimentos que incluem a realização de testes manuais e automatizados.

2.1.4 Validação da estratégia de testes

Um dos principais objetivos considerados durante a elaboração da estratégia de testes para o Doarti foi permitir a adequação ao contexto da equipe. Portanto fez-se necessária validação da estratégia por meio de entrevistas guiadas através da qual foi possível avaliar quão adequada está a estratégia desenvolvida à equipe do Doarti.

Para realizar essa validação, foram utilizados conceitos relacionados à qualidade de modelagem de processos de software. Foram definidas questões para guiar as entrevias considerando o trabalho desenvolvido por Silva (2014).

Para a execução desta fase, foi selecionado um subconjunto de características baseadas de qualidade de processo, retirado do modelo de características de qualidade de processo de software apresentado por Silva (2014). A utilização de um subconjunto limitado de características ocorreu, principalmente, por conta da limitação do tempo disponível para realização da validação.

Demais características do modelo de qualidade de processos de software, apresentado por Silva (2014), não foram priorizadas neste questionário por se tratarem de características que abrangem avaliações de aspectos que vão além da validação da utilização e compreensão do processo proposto.

Além de abranger um escopo maior de validações, as demais características de qualidade não utilizadas nesta validação, como: arquitetura, sintática e evolubilidade, tratam-se de aspectos mais relacionados à elaboração do que à utilização da estratégia. Juntamente a esse fato, é importante ressaltar que os entrevistados, alvos desta validação, tem o principal papel de usuários da estratégia.

Em relação à adequabilidade, acessibilidade, compreensibilidade, adaptabilidade e semântica, subcaracterísticas de funcionalidade e usabilidade, foram criadas as questões apresentadas na Tabela 3.

Tabela 3 – Questionário para validação da estratégia

ID	**Questão**	**Característica**
Q1	Você acredita que seria necessário muito esforço para adequar a equipe para a execução desta estratégia?	Adequabilidade
Q2	Você sentiu falta da explicação de algum elemento presente na estratégia?	Completude
Q3	Você conseguiu encontrar com facilidade explicações referentes a todos os elementos citados no modelo de processo apresentado?	Acessibilidade
Q4	A disposição dos textos e documentos explicativos sobre a estratégia permitiram que o fluxo de leitura seguisse o fluxo cronológico das atividades?	Acessibilidade
Q5	A navegação entre os textos e documentos explicativos sobre a estratégia é fácil?	Acessibilidade
Q6	Existe algum termo citado nos documentos referentes à apresentação da estratégia que você não conheça?	Validade
Q7	Você consegue imaginar alguma situação do contexto do Doarti não prevista pela estratégia proposta?	Adaptabilidade
Q8	Quais resultados você acredita que a aplicação dessa estratégia pode trazer?	Compreensibilidade

A estratégia foi disponibilizada antes das entrevistas de modo que os entrevistados tivessem tempo para estudar o material. Depois disso, foi conduzida uma entrevista individual com dois dos membros da equipe de desenvolvimento do Doarti. A entrevista foi guiada pelas questões apresentadas na Tabela 3.

O resultado desta validação, acompanhado da definição das características de qualidade relacionadas as questões, é apresentado na Seção 5.2.

3 Referencial teórico

Este capítulo apresenta os principais conceitos utilizados durante o desenvolvimento deste trabalho e está organizado em seções conforme apresentadas a seguir. A Seção 3.1 apresenta conceitos relacionados a equipes de desenvolvimento de software. A Seção 3.2 aborda normas e definições relacionadas à qualidade e medição de produtos de software, com foco na manutenibilidade. Posteriormente é apresentada a Seção 3.3, que trata da teoria acerca da metodologia ágil e *frameworks* como o Scrum. Em seguida, é apresentada a Seção 3.4, que expõe conceitos relacionados à abordagem de medição GQM. Por fim, na Seção 3.5, são apresentados conceitos relacionados à avaliação de processos através do MPS.BR.

3.1 Equipes de desenvolvimento de software

A complexidade presente no desenvolvimento de software é suficiente para exigir, por meio de múltiplos indivíduos e de forma compartilhada, a realização de diversas ações, utilizando conhecimentos relacionados a muitas áreas da tecnologia e gestão de pessoas (OLIVEIRA, 2019). A partir dessa complexidade, torna-se importante a análise das características específicas relacionadas ao perfil de cada equipe, de modo a identificar possíveis oportunidades de melhoria.

É importante ressaltar que o objetivo deste trabalho não se encontra em mitigar, de forma direta, as dificuldades relacionadas a cada um dos aspectos apresentados nas subseções seguintes, mas em elaborar uma estratégia de testes capaz de produzir bons resultados em meio a essas realidades.

O Doarti, projeto alvo de estudo deste trabalho, conta com equipes caracterizadas como voluntárias e geograficamente distribuídas, particularidades essas que serão abordadas nas subseções seguintes.

3.1.1 Equipes voluntárias

Como previsto na legislação brasileira, o serviço voluntário é uma "atividade não remunerada prestada por pessoa física à entidade pública de qualquer natureza ou à instituição privada de fins não lucrativos que tenha objetivos cívicos, culturais, educacionais, científicos, recreativos ou de assistência à pessoa". O exercício desse tipo de atividade, mediado apenas por um termo de adesão, apresenta o objeto e as condições de seu exercício, sem geração de vínculo empregatício, nem obrigação de natureza trabalhista, previdenciária ou afim (BRASIL, 1998).

Com a ausência desse vínculo empregatício, é possível inferir que a ligação entre o prestador de serviço e a organização, resultante da dinâmica de serviço voluntário, pode ser vulnerável a fatores como a falta de engajamento e a alta taxa de rotatividade dos membros voluntários.

Skoglund (2006) sugere que a falta de engajamento por parte de membros voluntários de uma organização pode ser causada por falhas cometidas nas ações relacionadas à recepção, à integração e ao treinamento destes.

Segundo Fischer e Schaffer (1993 apud STARNES; WYMER, 2001), a alta taxa de rotatividade de mão de obra voluntária é uma característica ligada às organizações que requerem a realização de tarefas mais complexas, exigindo pessoal altamente capacitado, com habilidades singulares.

O Doarti é um projeto de desenvolvimento de um software de impacto social positivo. O projeto depende de pessoas com conhecimentos sólidos relacionados ao ciclo completo de desenvolvimento de aplicações para dispositivos móveis e web, gerenciamento de base de dados não relacional e utilização de metodologias ágeis, conhecimentos esses que exigem certo esforço para capacitação. Logo, tal complexidade pode fazer com que o Doarti se encaixe no perfil de projeto suscetível a problemas com alta taxa de rotatividade de membros voluntários.

3.1.2 Equipes geograficamente distribuídas

Outra característica relevante da equipe do projeto Doarti é a distribuição geográfica. O projeto surgiu num momento em que o trabalho presencial é pouco incentivado, devido a questões de segurança, por conta da pandemia do novo coronavirus. Contudo, o trabalho remoto não é inerente somente a esse contexto específico, podendo ser utilizado por diversas organizações em momentos distintos, logo, os desafios e oportunidades que essa dinâmica de trabalho oferece devem ser devidamente analisados.

Um dos principais desafios do trabalho em equipe de forma remota está centrado na escassez de comunicação face-a-face, que é substituída por comunicações mediadas por tecnologias apropriadas (PANTELI; YALABIK; RAPTI, 2019). Interações mediadas por tecnologias de comunicação remota e a sensação de se trabalhar com pessoas estranhas ao convívio presencial do indivíduo criam experiências de desconforto social e ansiedade, podendo impactar diretamente na qualidade das relações interpessoais internas da equipe (PANTELI; YALABIK; RAPTI, 2019).

Espinosa et al. (2002) indica que equipes de desenvolvimento de software geograficamente distribuídas possuem desvantagens em sua coordenação, por conta da menor frequência e espontaneidade das interações entre seus membros, ocasionando possíveis falhas de comunicação e demora na obtenção de respostas.

O Doarti conta com membros geograficamente distribuídos, em sua maioria, sob o mesmo fuso horário, contudo, é exigido de cada membro a publicação de seus horários de dedicação ao projeto, facilitando a realização de interações síncronas quando necessário. Apesar disso, contribuições provenientes de atividades assíncronas continuam sendo essenciais.

Equipes geograficamente distribuídas também estão ligadas à assincronia de trabalho entre os membros, pois carecem de interações presenciais, utilizando-se de ferramentas de comunicação síncronas e assíncronas para possibilitar a comunicação entre os membros (PANTELI; YALABIK; RAPTI, 2019).

A utilização de ferramentas de comunicação assíncronas adiciona ao projeto dificuldades na manutenção da riqueza e celeridade de informações (PANTELI; YALABIK; RAPTI, 2019).

No Doarti, apesar da distribuição geográfica da equipe e da publicação, por parte dos membros, de informações relacionadas ao horário de trabalho, não são feitas restrições pela coordenação do projeto relacionadas ao turno de trabalho. Isso proporciona certa liberdade para que os membros da equipe possam trabalhar em horários que sejam mais oportunos para cada um deles. Com essa liberdade, surge o desafio de manter a equipe em sintonia, no que se refere à produção individual dos membros.

3.2 Manutenibilidade de software

Uma parte considerável da contribuição para o projeto é realizada de forma assíncrona, por isso é interessante que a manutenibilidade seja uma característica cada vez mais presente no código do Doarti. Responsáveis por uma tarefa não estão sempre em contato, de forma síncrona, devido a isso, os artefatos gerados devem ser facilmente compreensíveis, permitindo a continuidade de desenvolvimento e manutenção por parte de outros membros.

Alguns fatores aos quais o Doarti está suscetível, como a alta taxa de rotatividade de membros e o trabalho assíncrono, justificam o enfoque deste trabalho na manutenibilidade e suas sub-características, detalhadas na Subseção 3.2.1.

3.2.1 ISO/IEC 25000

A ISO (*International Organization of Standardization* - Organização Internacional de Padronização) e a IEC (*International Electrotechnical Commission* - Comissão Eletrotécnica Internacional) constituem um sistema para criação de padrões internacionais (ISO/IEC, 2004). Neste trabalho serão abordados alguns padrões como a ISO/IEC 25000, ISO/IEC 9126, ISO/IEC 14598 entre outras relacionadas a processos e produtos

de software.

A ISO/IEC 25000, também conhecida como SQuaRE (*Software product Quality Requirements and Evaluation* - Requisitos e Avaliação da Qualidade de produtos de Software), é resultado de outros padrões como a ISO/IEC 9126, que define modelos e métricas para qualidade de software, e a ISO/IEC 14598, que apresenta o processo para avaliação de produtos de software (ISO/IEC, 2004).

A série de normas SQuaRE apresenta modelos de gerenciamento, qualidade, medição, requisitos e avaliação relacionados à qualidade de software (ISO/IEC, 2004).

Em relação à qualidade, são definidos modelos hierárquicos, compostos por características e sub características, tanto para o produto de software, quanto para a qualidade em uso, e cada sub característica está relacionada a, ao menos uma, propriedade mensurável (ISO/IEC, 2010).

O modelo de qualidade em uso considera características resultantes da interação dos usuários diretos e indiretos com o sistema, em contextos específicos (ISO/IEC, 2010). Tais características e respectivas sub características são apresentadas no modelo de qualidade em uso exposto na Figura 2.

Eficácia
Eficiência
Utilidade
Confiança
Prazer
Conforto
Liberdade de risco
Mitigação de riscos econômico
Mitigação de riscos de saúde e segurança
Mitigação de riscos ambientais
Cobertura de contexto
Completude de contexto
Flexibilidade
Reconhecibilidade

Figura 2 – Modelo de qualidade em uso. Fonte: Adaptado de ISO/IEC (2010)

Já o modelo de qualidade de produto de software, compreende características relacionadas diretamente ao produto de software em si, que podem afetar a qualidade em uso, seja na perspectiva do usuário direto, indireto seja nos demais envolvidos no projeto do software (ISO/IEC, 2010). As características e sub características do modelo de qualidade de produto de software são apresentadas no modelo ilustrado na Figura 3.

Dentre as características relacionadas à qualidade, nos dois modelos apresentados, a manutenibilidade foi selecionada como foco deste trabalho, por questões previamente explicitadas no início desta seção, relacionadas ao perfil da equipe do projeto.

Subcaracterísticas
Adequação Funcional
Completude funcional
Corretude funcional
Conformidade funcional
Eficiência de desempenho
Comportamento temporal
Utilização de recursos
Capacidade
Compatibilidade
Coexistência
Interoperabilidade
Usabilidade
Reconhecibilidade
Capacidade de aprendizagem
Operabilidade
Proteção de erros do usuário
Estética de interface do usuário
Acessibilidade

Confiabilidade
Maturidade
Disponibilidade
Tolerância a falhas
Recuperabilidade
Segurança
Confidencialidade
Integridade
Não-repúdio
Prestação de contas
Autenticidade
Manutenibilidade
Modularidade
Capacidade de reutilização
Analisabilidade
Modificabilidade
Testabilidade
Portabilidade
Adaptabilidade
Instalabilidade
Substituibilidade

Figura 3 – Modelo de qualidade de produto de software. Fonte: Adaptado de ISO/IEC (2010)

A manutenibilidade representa a facilidade com a qual o produto de software pode sofrer correções, melhorias ou adaptações por parte de seus mantenedores. Tal característica é composta por outras como modularidade, reusabilidade, analisabilidade, modificabilidade e testabilidade. A ISO/IEC (2010) define cada uma dessas sub características da seguinte maneira:

- Modularidade: Representa o quanto o sistema é constituído por componentes bem isolados, cuja alteração causa o menor impacto possível nos demais componentes.
- Reusabilidade: Define a capacidade de um elemento em ser reaproveitado em outros sistemas ou no desenvolvimento de outros elementos do sistema.
- Analisabilidade: Representa a capacidade do sistema de: permitir o diagnóstico de deficiências ou causas de falha; permitir identificação de partes a serem modificadas, bem como os impactos sobre o sistema.
- Modificabilidade: É a capacidade do sistema em ser modificado, de forma eficaz e eficiente, sem que haja adição de defeitos ou prejuízo à qualidade do produto.
- Testabilidade: Representa o grau de efetividade no qual critérios de testes podem ser definidos e executados no sistema, produto ou componente.

O modelo de medição, também definido pela série de normas ISO/IEC 25000, apresenta medições para qualidade interna e externa do produto de software. A qualidade interna se refere aos elementos não executados do produto, tais como documentos, especificações e código fonte (ISO/IEC, 2011). Já a qualidade externa, refere-se aos atributos ligados ao comportamento do software em execução e, por isso, podem ser realizadas na fase de testes (ISO/IEC, 2011).

Segundo a ISO/IEC (2006), as métricas de qualidade interna do produto de software conferem aos envolvidos, no desenvolvimento e gerenciamento do produto, a habilidade de medir a qualidade dos artefatos intermediários, assim como prever a qualidade final do produto de software. Tal habilidade também possibilita a tomada de ações corretivas, em cima de possíveis problemas de qualidade que possam surgir nos estágios iniciais do desenvolvimento do software (ISO/IEC, 2006).

Na ISO/IEC (2011) são listados exemplos de métricas para medir cada característica e sub característica presentes nos modelos de qualidade, contudo, não há obrigatoriedade de aplicação dessas métricas. Segundo a (ISO/IEC, 2011), métricas diferentes das listadas na série de normas ISO/IEC 2502x podem ser utilizadas pelo usuário da norma, desde que sejam especificados a relação entre as métricas utilizadas e os modelos de qualidade citados na série de normas ISO/IEC 25010 ou demais modelos de qualidade utilizados no contexto do projeto.

As medições, especificamente da manutenibilidade interna, são utilizadas para prever os esforços ou recursos gastos na realização de alterações no sistema (ISO/IEC, 2011). Já as medições relacionadas à manutenibilidade externa se referem ao comportamento dos mantenedores, usuários e do próprio software durante os processo de manutenção, modificação e teste do produto (ISO/IEC, 2011).

Como consequência do objetivo deste trabalho e, sabendo que o conjunto de métricas selecionadas tem como finalidade auxiliar a avaliação do processo de testes realizado pela equipe do Doarti, as medições planejadas para estratégia de testes deste trabalho serão relacionadas à qualidade interna e externa, focando em atributos de manutenibilidade do produto de software desenvolvido no Doarti. As métricas elaboradas para orientar tais medições são apresentadas e detalhadas no Apêndice B.

3.3 Metodologia ágil

A metodologia utilizada em um projeto de desenvolvimento de software representa as atividades desempenhadas e resultados associados para produção do software (SOARES, 2004). O Doarti utiliza alguns princípios de metodologias ágeis para o desenvolvimento do software. Portanto, o conhecimento relacionado a essas metodologias, bem como outras alternativas, auxilia na contextualização deste trabalho.

As metodologias tradicionais são caracterizadas por serem menos receptivas às mudanças, geralmente são orientadas a documentos e planejamentos e são mais adequadas para projetos em que os requisitos são mais estáveis e bem compreendidos (SOARES, 2004).

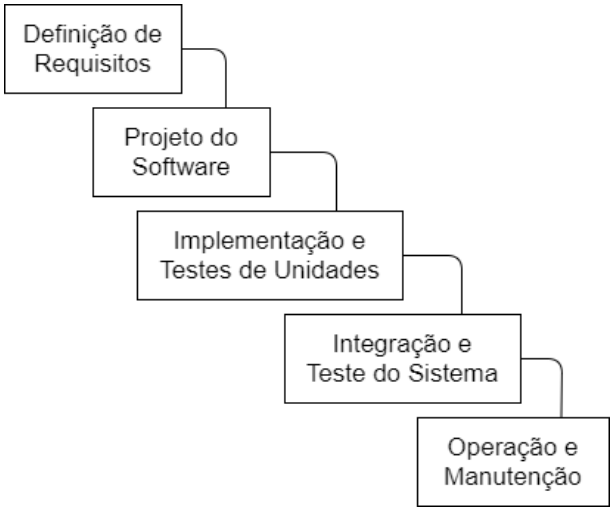

Figura 4 – Modelo Clássico. Fonte: Adaptado de Soares (2004)

O modelo Clássico ou Sequencial, ilustrado na Figura 4 foi um dos primeiros a ser proposto, no qual o desenvolvimento do software é dividido em etapas, que são validadas por meio dos documentos gerados ao final da cada fase (SOARES, 2004).

Metodologias tradicionais surgiram num contexto muito diferente do atual, e um dos principais problemas dessas metodologias é a inflexibilidade, que as torna menos receptivas a mudanças cada vez mais comuns nos projetos de desenvolvimento de software (SOARES, 2004).

Focada na mudança de valores, a metodologia ágil surge baseando-se nos indivíduos e interações, passando os demais elementos ligados às metodologias tradicionais para o segundo plano, sem abdicar completamente dos destes (SOARES, 2004). Segundo Beck et al. (2001), essa mudança de valores pode ser descrita, resumidamente, nas seguintes frases:

"Indivíduos e interações mais que processos e ferramentas"
"Software em funcionamento mais que documentação abrangente"
"Colaboração com o cliente mais que negociação de contratos"
"Responder a mudanças mais que seguir um plano"

Os princípios da metodologia ágil se baseiam, entre outras características, em priorizar o cliente, entregando software em funcionamento, de forma contínua, com atenção à excelência técnica e design, sempre agregando o maior valor possível (BECK et al., 2001).

3.3.1 Scrum

Segundo Sutherland e Schwaber (2013), o Scrum é um *framework* de estruturação de equipes ágeis, cujo objetivo é ser leve e de simples entendimento, sendo capaz de gerar produtos complexos de forma adaptativa, iterativa e incremental. O *framework* tem como base times associados a papéis, eventos, artefatos e regras. Cada time é auto-gerenciável, multifuncional e dono de suas atribuições específicas, essenciais para o sucesso do *framework* (SUTHERLAND; SCHWABER, 2013). O Scrum define alguns papéis a serem seguidos por indivíduos ou times, dentre eles estão:

- *Product Owner*: É o responsável por priorizar e maximizar o valor do produto, através do gerenciamento do *Backlog* do Produto (SUTHERLAND; SCHWABER, 2013). O *Backlog* do Produto é uma lista ordenada de itens necessários para completude do produto, cujo conteúdo, ordenação e disponibilidade, é de responsabilidade do *Product Owner* (SUTHERLAND; SCHWABER, 2013).

- Time de Desenvolvimento: É composto de pessoas responsáveis por criar incrementos para o produto, entregando, ao final de cada *sprint*, uma versão utilizável e de maior valor (SUTHERLAND; SCHWABER, 2013). A *Sprint* é um dos principais eventos do Scrum e representa uma iteração, composta por eventos de planejamento, acompanhamento, revisão e retrospectiva, bem como sua execução, que tem duração de um mês ou menos, a depender da complexidade de desenvolvimento do produto (SUTHERLAND; SCHWABER, 2013).

- *Scrum Master*: É um papel de liderança, com a responsabilidade de guiar a equipe para que as práticas do Scrum sejam seguidas corretamente, afim de obter a maior produtividade possível (SUTHERLAND; SCHWABER, 2013).

Na Figura 5 são representados alguns dos principais papéis, artefatos e eventos do Scrum.

É possível observar, de forma ilustrada e simplificada, a relação de liderança e apoio do *Scrum Master* para com o *Product Owner* e a equipe de desenvolvimento. Também é apresentada uma cronologia simples dos eventos de planejamento da *Sprint*, envolvendo o *Product Owner* na priorização do *Backlog* do produto para obtenção do *Backlog* da *Sprint*, bem como os eventos de revisão e retrospectiva.

Grande parte desses papéis, eventos e artefatos estão presentes no desenvolvimento do projeto Doarti. Dessa forma, considerando o objetivo de se propor uma estratégia de testes eficiente no Doarti, observa-se a necessidade de preservar cada um desses elementos, fazendo com que eles não sejam desfavorecidos, mas auxiliem no sucesso da estratégia em caso de sua aplicação prática.

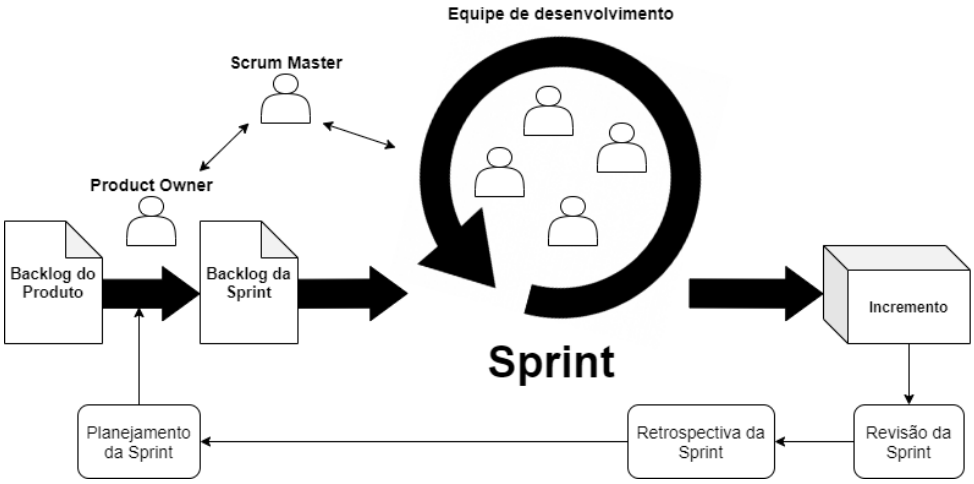

Figura 5 – Scrum. Fonte: Adaptado de Ramos (2019)

3.4 GQM

Pressman (2011) define que estratégias de teste de software devem fornecer os passos necessários para teste, quantificar o esforço necessário, coletar e avaliar os dados resultantes. Para tornar possível obtenção de dados resultantes na estratégia definida neste trabalho, foram elaboradas métricas, embasadas com auxílio da abordagem GQM.

Na engenharia de software, medições são muito importantes para tornar possíveis a análise e a aplicação de melhorias e correções em processos, recursos de trabalho ou artefatos entregáveis (CALDIERA; ROMBACH, 1994). O GQM é uma abordagem de medição dividida em três níveis: conceitual, operacional e quantitativo, e tem como base a definição de objetivos, questões e métricas de forma hierárquica, seguindo um processo definido (CALDIERA; ROMBACH, 1994).

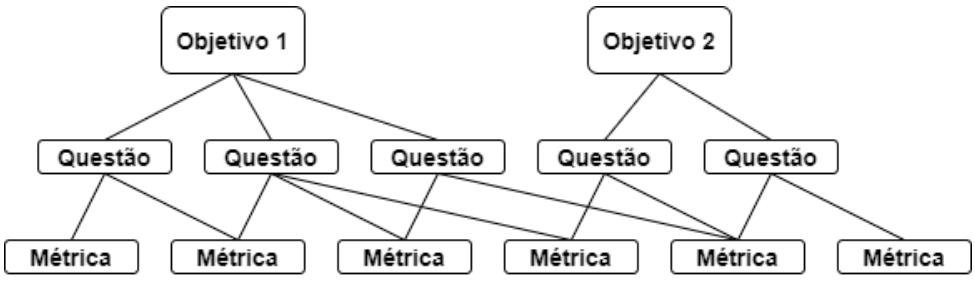

Figura 6 – Níveis do GQM. Fonte: Adaptado de Caldiera e Rombach (1994)

Os **objetivos** (*goal*) representam o nível conceitual do GQM. Esse nível é baseado nas necessidades de melhoria da organização e tem como finalidade definir, de forma clara, os problemas, propósitos, perspectivas e objetos sendo esse último, um processo, recurso ou produto da organização (CALDIERA; ROMBACH, 1994).

As **questões** (*question*) representam o nível operacional. São elaboradas de forma

a caracterizar elementos dos objetivos, com a intenção de auxiliar na seleção das métricas (CALDIERA; ROMBACH, 1994).

Por último, as **métricas** (*metric*), no nível quantitativo, têm o papel de tornar os objetivos mensuráveis e, por consequência, embasar a tomada de decisões na organização (CALDIERA; ROMBACH, 1994).

Neste trabalho, foram selecionadas métricas de acordo com modelos de qualidade definidos pela série de normas ISO/IEC 25000, abordada previamente na Seção 3.2. Essas métricas, por sua vez, foram atribuídas aos respectivos objetivos e questões referentes aos níveis conceitual e operacional do GQM. As métricas e suas atribuições aos níveis do GQM são detalhadas no Apêndice B.

3.5 MPS.BR

O MPS.BR é um programa que apresenta modelos em conformidade com padrões aceitos internacionalmente para auxiliar na definição, avaliação e melhoria de processos de software, especialmente no contexto das micro, pequenas e médias empresas privadas ou governamentais (SOFTEX, 2020).

Foi desenvolvido sob coordenação da SOFTEX (Sociedade para Promoção da Excelência do Software Brasileiro), com apoio do Ministério da Ciência, Tecnologia, Inovações e Comunicações (MCTIC), Financiadora de Estudos e Projetos (FINEP), Serviço Brasileiro de Apoio a Micro e Pequenas Empresas (SEBRAE), Banco Interamericano de Desenvolvimento (BID/FUMIN), e também recebe contribuições de universidades, instituições governamentais, centros de pesquisa e organizações privadas (SOFTEX, 2020).

O MPS.BR conta com um modelo de referência, detalhado na Subseção 3.5.1, responsável por definir alguns termos importantes para avaliação de processos.

3.5.1 Modelo de Referência

O Modelo de Referência do MPS.BR para software (MR-MPS-SW), responsável por definir os níveis de maturidade dos processos, foi desenvolvido com base na norma ISO/IEC 12207, na série de normas ISO/IEC 15504 (SPICE) e no modelo CMMI (Capability Maturity Model Integration - Modelo Integrado de Capacidade e Maturidade) (WEBER et al., 2015).

A Norma ISO/IEC 12207 tem como objetivo estabelecer um modelo para os processos presentes no ciclo de vida de softwares e contribuiu na concepção do MR-MPS-SW, com terminologias, definições, propósitos e resultados esperados, relacionados a processos. A série de normas ISO/IEC 15504 define padrões para avaliação de processos (PRESSMAN, 2011) e contribuiu na definição do MR-MPS-SW, ampliada e substituída, em par-

tes, pela família de normas ISO/IEC 330xx, com terminologias, definição de processos e capacidade de processos (SOFTEX, 2020).

O CMMI é um modelo de capacidade de maturidade de processos que também é utilizado como base para o MPS.BR e apresenta uma série de boas práticas, divididas em níveis de 0 a 5, de maturidade, a serem adotadas gradualmente pelas organizações (SOMMERVILLE, 2019). Foi desenvolvido pelo SEI (Software Engineering Institute - Instituto de Engenharia de Software) na década de 1990, com o intuito de integrar os modelos já existentes de capacidade e maturidade de processos da engenharia de software (SOMMERVILLE, 2019). O modelo é representado de duas formas diferentes, por estágios e contínua.

A representação por estágios do CMMI apresenta níveis de maturidade de 1 a 5 para caracterizar, em conjunto, os processos de gerenciamento de uma organização (SOMMERVILLE, 2019). Já a representação contínua, apresenta níveis de maturidade de 0 a 5 para caracterizar 22 áreas de processo de forma independente (SOMMERVILLE, 2019).

A avaliação de processo realizada no Doarti utilizou como referência os modelos e métodos do MPS.BR, seguindo a representação contínua do CMMI, pois apenas o processo de verificação e validação foi avaliado, de forma isolada.

Os níveis de maturidade definidos pelo MR-MPS-SW representam a combinação entre processos e sua capacidade, sendo que os processos são divididos em organizacionais e de projeto (SOFTEX, 2020).

A Figura 7, a seguir, representa a divisão dos conjuntos de processos organizacionais e processos de projeto.

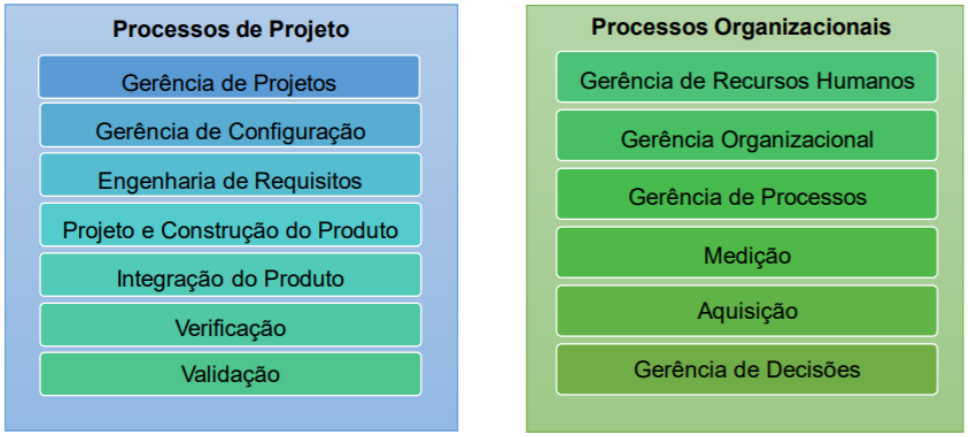

Figura 7 – Processos de Projeto e Organizacionais. Fonte: (SOFTEX, 2020)

Os processos organizacionais se referem àqueles responsáveis por fornecer recursos para execução adequada dos projetos da instituição (SOFTEX, 2020). Já os processos de

projeto, são relacionados ao desenvolvimento, à manutenção ou à evolução de produtos atribuídos a projetos específicos dentro de uma organização (SOFTEX, 2020).

Os níveis de maturidade dos processos são divididos em: A (Em otimização), B (Gerenciado quantitativamente), C (Definido), D (Largamente definido), E (Parcialmente definido), F (Gerenciado) e G (Parcialmente gerenciado) e são baseados nos níveis 2, 3, 4 e 5 do CMMI, como ilustra a Figura 8 (WEBER et al., 2015).

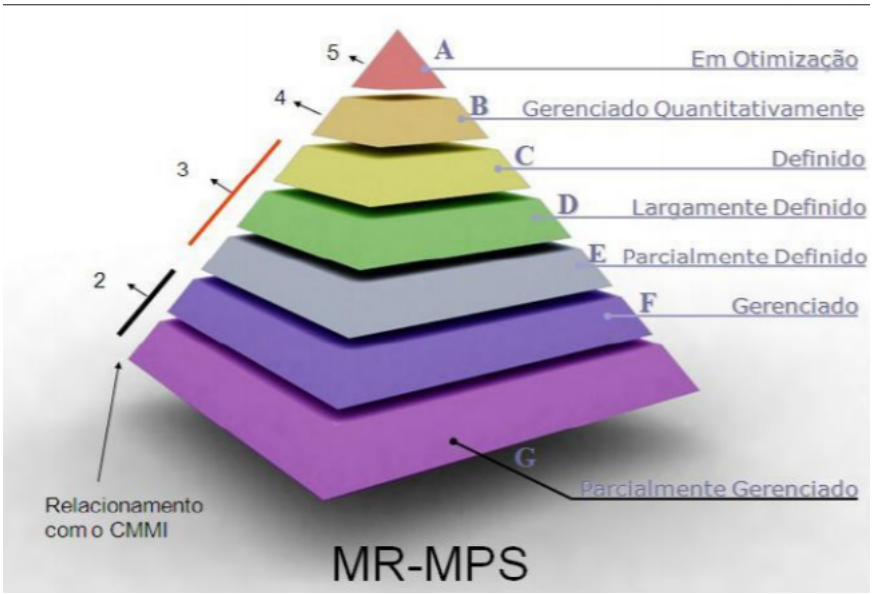

Figura 8 – MPS.BR x CMMI. Fonte: Ferreira (2009 apud PIMENTEL, 2013)

Um dos principais objetivos do MPS.BR é tornar mais acessível a implementação da avaliação e melhoria de processos no contexto das organizações brasileiras de menor porte, pois, comparado ao CMMI, o MPS.BR apresenta transições mais graduais entre os níveis de maturidade dos processos (PIMENTEL, 2013).

3.5.2 Método de Avaliação

O processo e o método de avaliação do MPS.BR definem formas de verificar a adequação das práticas inerentes a cada área de processo e dessas áreas de processos aos níveis de maturidade definidos no MR-MPS-SW (SOFTEX, 2021a). As práticas relacionadas a uma área de processo são avaliadas a partir de indicadores diretos, indiretos ou de afirmações, definidos pela organização avaliada (SOFTEX, 2021a). Indicadores diretos representam artefatos decorrentes da execução de alguma prática presente no processo (SOFTEX, 2021a). Indicadores indiretos se referem aos elementos capazes de indicar a realização de uma atividade (SOFTEX, 2021a). Afirmações são obtidas através de relatos, em entrevistas com os membros dos projetos da organização, que também podem indicar a realização de atividades (SOFTEX, 2021a).

Os níveis de implementação das práticas de processos são divididos em: TI - Totalmente Implementada; LI – Largamente Implementada; PI – Parcialmente Implementada e NI- Não Implementada (WEBER et al., 2015). As regras para definição da implementação das práticas são descritas na Tabela 4.

Neste trabalho foi avaliado o processo de verificação e validação do projeto Doarti. O objetivo era avaliar o processo em relação a capacidade de processo CP-G (Capacidade de processo Nível G), definido pelo MR-MPS-SW, para que se pudesse obter uma conclusão sobre o estado atual do processo em questão, no Doarti. A capacidade de um processo é representada por um conjunto resultados esperados por determinado nível de capacidade (SOFTEX, 2021b).

De forma resumida, o nível CP-G define um processo, cuja execução é gerenciada, ou seja, é planejada, monitorada, capaz de produzir resultados definidos com seus executores preparados para a atuação adequada nas atribuições (SOFTEX, 2020). Para que um processo satisfaça esse nível, é necessário que as práticas que indiquem tais características sejam definidas como totalmente ou largamente implementadas (SOFTEX, 2021a).

Tabela 4 – Regras de definição da implementação das práticas. Fonte: Adaptada de Weber et al. (2015)

Grau de Implementação da Prática	**Caracterização**	**Grau de alcance**
Totalmente Implementado	- O indicador direto está presente e julgado adequado - Existe pelo menos um indicador indireto e/ou afirmação para confirmar a implementação - Não foi notada nenhuma fraqueza substancial	>85% a 100%
Largamente Implementado	- O indicador direto está presente e julgado adequado - Existe pelo menos um indicador indireto e/ou afirmação para confirmar a implementação - Foi notada uma ou mais fraquezas	>50% a 85%
Parcialmente Implementado	- O indicador direto não está presente ou é julgado inadequado - Artefatos ou afirmações sugerem que alguns aspectos da prática estão implementados - Fraquezas foram documentadas	>15% a 50%
Não Implementado	- Qualquer situação diferente das acima	0 a 15%

3.6 Testes de software

Segundo Naik e Tripathy (2011), testes de software estão relacionados ao tópico de verificação e validação (V&V). As atividades V&V podem ser classificadas como de análise estática e dinâmica, em ambos casos o objetivo é analisar e melhorar a qualidade do software.

São consideradas atividades de análise estática aquelas que não incluem execução de códigos referentes ao desenvolvimento do produto, como inspeções, análises de algorítimos, revisão de código, entre outras (NAIK; TRIPATHY, 2011). Já as atividades de análise dinâmica, dependem da execução do código referente ao desenvolvimento do produto (NAIK; TRIPATHY, 2011).

Testes são classificados como atividades de análise dinâmica. Eles podem ser realizados utilizando-se de dados fictícios tendo como foco cenários finitos, de uso real; ou cenários forjados com o objetivo explícito de encontrar defeitos (SOMMERVILLE, 2019).

Não é possível garantir que os casos de testes cobrem todos os cenários possíveis, uma vez que implica em um número enorme de cenários o que torna impraticável a realização de testes de forma exaustiva (NAIK; TRIPATHY, 2011). Portanto, é importante que os responsáveis pela elaboração desses casos de teste saibam escolher, criteriosamente, os mais eficazes, baseando-se em análises de risco relacionadas aos elementos alvo dos testes (NAIK; TRIPATHY, 2011).

Testes também são classificados em relação a característica de qualidade nas quais se propõem a explorar. Testes funcionais exploram a funções e funcionalidades do sistema, enquanto testes não-funcionais exploram características não funcionais do sistema, como confiabilidade, usabilidade, manutenibilidade, eficiência e portabilidade, segundo a ISO 9126 (HOMÈS, 2013).

3.6.1 Etapas de teste

As atividades de teste muitas vezes são desempenhadas pela equipe de desenvolvimento, e em alguns casos, por equipes independentes, encarregadas do processo de testes (SOMMERVILLE, 2019). Os testes realizados pela equipe de desenvolvimento possuem três níveis de granularidade:

- **Teste unitário**: são feitos para os menores componentes possíveis de um software, como métodos e classes de um objeto, que devem ser testados de forma isolada, sempre que possível (SOMMERVILLE, 2019).
- **Teste de componente**: são testes centrados na interface de componentes, que podem ser construídos a partir de várias unidades individuais integradas (SOMMERVILLE, 2019). Neste nível de teste o foco é a interface do componente, portanto, assume-se que as unidades que o compõem já estão contempladas pelos testes unitários (SOMMERVILLE, 2019).
- **Teste de sistema**: São responsáveis por testar a integração entre alguns ou todos os componentes de um sistema (SOMMERVILLE, 2019). É comum que testes de sistema sejam desenvolvidos com base em casos de uso do sistema, pois assim as interações desejadas entre os componentes envolvidos podem ser provocadas (SOMMERVILLE, 2019).

No caso deste trabalho, que tem como alvo o Doarti, projeto que tem seu aplicativo escrito seguindo o *framework* Flutter, os testes de componentes possuem como foco os

widgets, que são os elementos que representam os componentes no contexto do Flutter. Já os testes unitários, são aplicados, usualmente, as classes, métodos e objetos que compõem a lógica de negócio por trás dos *widgets*. Por fim, os testes de sistema, no Doarti, são aplicados a múltiplos *widgets*, e suas integrações com classes e métodos responsáveis pela lógica de negócio.

Além dos testes realizados durante a fase de desenvolvimento, um software pode ser realizado em outras etapas do desenvolvimento, representado pelos testes de *release* e os testes de usuário.

- **Teste de *release***: com o objetivo de verificar se o sistema é adequado e está pronto para uso, o teste de *release* é realizado por um equipe externa à equipe de desenvolvimento, e representa o processo de testes sob uma versão do sistema que está prestes a ser liberada para uso (SOMMERVILLE, 2019).

- **Teste de usuário**: representa uma etapa de testes na qual o usuário final ou cliente avalia, experimenta e faz sugestões no sistema (SOMMERVILLE, 2019). Esta etapa é importante por conta da exposição do sistema ao seu ambiente alvo, que não pode ser totalmente replicado nas demais etapas de teste (SOMMERVILLE, 2019) .

No Doarti, formalmente, apenas o teste de *release* está presente no processo de testes e são executados manualmente pela equipe de testes. Os testes automatizados, sejam eles unitários, de componentes ou de sistema, existem apenas para alguns elementos específicos do sistema e foram criados para fins de experimentação e aprendizagem da equipe em relação aos testes automatizados com Flutter.

Percebe-se que, assim como no Doarti, é comum que processos de teste sejam compostos por uma mistura de testes manuais e automatizados, que englobam as etapas de testes de desenvolvimento *release* e usuário.

Segundo Sommerville (2019), testes manuais são executados manualmente pelo testador, tendo como base dados de teste selecionados e resultando em reportes decorrentes de discrepâncias entre os resultados do teste e as expectativas do testador.

Já os testes automatizados são codificados e podem ser executados, preferencialmente, na realização dos testes da etapa de desenvolvimento. Testes automatizados se mostram como uma forma mais eficiente, comparada aos testes manuais, na realização de testes de regressão, que focam na detecção da inserção de novas falhas decorrentes de alterações no código (SOMMERVILLE, 2019).

3.6.2 Técnicas de teste

Os testes também podem ser classificados em duas categorias, que representam a abordagem adotada em sua realização, e são elas: caixa-branca e caixa-preta. Testes de caixa-preta são orientados a dados de entrada e saída, adquiridos por meio das especificações do sistema. Para adoção desta abordagem, a pessoa responsável pelo teste não pode se orientar por meio do código do sistema(MYERS; SANDLER; BADGETT, 2011). Já os testes de caixa-branca permitem que o testador se guie por meio de uma análise do código-fonte do sistema, podendo se orientar por meio de sua lógica (MYERS; SANDLER; BADGETT, 2011).

No Doarti, não há limitações quanto ao uso de testes de caixa-branca ou caixa-preta, em qualquer um dos níveis de granularidade de teste adotados, pois tanto os desenvolvedores quanto os testadores tem acesso livre ao código-fonte do sistema.

3.7 Estratégias de testes

Uma estratégia de teste é geralmente desenvolvida pelo gerente de projeto, engenheiros de software ou especialistas em testes. Ela deve conter o planejamento da execução dos testes, recursos necessários e formas de coleta e avaliação dos dados resultantes (NAIK; TRIPATHY, 2011).

As subseções a seguir apresentam algumas estratégias de testes que se assemelham àquela proposta nesse trabalho.

3.7.1 Cow_Suite

Basanieri et al. (2001) aplica em seu trabalho estratégias para guiar a seleção e desenvolvimento de testes automatizados baseando-se nos diagramas UML (Unified Modeling Language - Linguagem de Modelagem Unificada) e nos custos previstos para realização dos testes.

A estratégia é composta por dois métodos para seleção e desenvolvimento de testes automatizados, são eles: UIT (Use Interaction Test - Teste de Interação de Uso) e Cowtest (Cost Weighted Test Strategy - Estratégia de Teste de Custo Ponderado) (BASANIERI et al., 2001). O ambiente composto pela mesclagem desses dois métodos, apelidado de Cow_Suite (Cow pluS UIT Environment - Ambiente Cow mais UIT), utiliza o UIT como fonte para criação dos casos de teste, e o Cowtest para priorizar os testes desenvolvidos (BASANIERI et al., 2001).

O Cow_Suite utiliza-se de uma estrutura hierárquica de diagramas de caso de uso, associados a seus respectivos diagramas de sequência, que servem como base para elaboração dos casos de teste por meio do método UIT (BASANIERI et al., 2001). Já

o Cowtest propõe que sejam atribuídos pesos no intervalo de 0 a 1 a essa estrutura (BASANIERI et al., 2001). Esses pesos definem a importância, em termos de risco, da estrutura ponderada e o grau de esforço necessário para testá-la (BASANIERI et al., 2001).

Essa abordagem tem um ponto interessante, principalmente no quesito da análise de riscos, que é feita de forma mais criteriosa para guiar a priorização dos testes a serem executados.

3.7.2 Algorítimo de ordem de teste

Proposta por Kung et al. (1995), a estratégia define uma ordem para construção de testes unitários e de integração para os elementos desejados, cujo esforço para tal seja mínimo. Para tanto, é utilizado um algorítimo chamado pelos autores de Test Order (Ordem de Teste) que, por sua vez, utiliza-se do ORD (Object Relation Diagram - Diagrama de Relação de Objeto), um grafo direcionado, representando as classes do sistema e seus relacionamentos, para computar a melhor ordem de criação dos testes (KUNG et al., 1995). O principal objetivo da estratégia é permitir a reutilização de casos de testes previamente gerados, em novos contextos de utilização (KUNG et al., 1995).

Aspectos dessa estratégia se mostram muito valorosos para este trabalho, pois foi pensada especificamente para o paradigma orientado a objetos, que também é utilizado no Doarti.

4 Avaliação inicial

Como já mencionado ao longo deste trabalho, a avaliação inicial é resultado da segunda fase do trabalho, detalhada no Capítulo 2, e teve como objetivo estabelecer uma base para o desenvolvimento da estratégia de testes para o Doarti, que se concentra em explorar falhas encontradas no resultado desta avaliação.

Dentre os processos previstos no guia de avaliação do MPS.BR, o de verificação e validação é o que mais se encaixa como alvo da estratégia de testes elaborada. Portanto, o resultado dessa avaliação apresenta o estado inicial deste processo no Doarti.

A avaliação foi realizada de acordo com a representação contínua do CMMI, considerando apenas o processo de verificação e validação do Doarti, de forma isolada, e seguindo o método de avaliação do MPS.BR, com adaptações relacionadas ao contexto do trabalho. Tais adaptações se referem ao fato dessa avaliação focar apenas um um processo e não ter sido realizada por uma instituição avaliadora, credenciada, como prevê o guia de avaliação do MPS.BR. Dessa forma, optou-se por nomear a avaliação realizada como avaliação informal, servindo apenas para fins acadêmicos. Informações sobre o CMMI e MPS.BR estão detalhados na Seção 3.5.

4.1 Resultados esperados do processo de Verificação e Validação

Nesta avaliação, excepcionalmente, o Doarti é tratado como uma organização, visto que, em seu contexto, não existem projetos definidos. Para esta avaliação foi realizada considerando o nível G de capacidade de processos do MPS.BR.

Segundo o SOFTEX (2021b), o processo de V&V (Verificação e Validação) possui resultados esperados listados de V&V1 a V&V5, que podem ser atendidos, em uma avaliação, por meio da apresentação de evidências e afirmações, mostrando que o projeto ou organização de fato implementa, em determinado grau, o processo de V&V.

Nas seções seguintes são explicados os resultados esperados para o processo de V&V e apresentadas as evidências, fraquezas e sugestões de melhoria encontradas para este processo no Doarti.

4.1.1 Resultado esperado V&V1

O resultado esperado V&V1 é: “Produtos de trabalho a serem verificados e validados são selecionados” (SOFTEX, 2021b). Portanto, para atestar o alcance desse resultado, no Doarti, são necessárias evidências que comprovem a existência de atividades de seleção

do que será ou não testado.

Para este resultado esperado, foram encontradas evidências diretas (ED), indiretas (EI) e fraquezas que serão listadas e detalhadas a seguir.

- ED: O documento: "Abordagem e Procedimentos no Desenvolvimento de Software DOARTI", presente no ambiente de comunicação e documentação do Doarti, define o procedimento de realização de testes manuais, indicando que implementações devem ser selecionadas para teste ao serem concluídas, entrando em uma etapa específica de testes.
- EI: É verificada a existência de uma etapa, em que são selecionadas implementações prontas para verificação e validação, ambiente de planejamento de demandas do Doarti.
- EI: É verificada a existência de testes automatizados desenvolvidos para alguns componentes, no código fonte do aplicativo móvel do Doarti.

Apesar da existência de tais evidências, foram encontradas algumas fraquezas substanciais. Não foi encontrada uma definição clara que determine quais componentes do código devem receber testes automatizados. Além disso, dentre os testes automatizados encontrados, alguns estão incompletos.

Como sugestão de melhoria, a ser estudada durante a elaboração da estratégia de testes para o Doarti, pode-se considerar o foco na documentação de um planejamento para realização de testes automatizados, para que esses sejam implementados com mais clareza pelos desenvolvedores.

Seguindo as regras para definição do grau de implementação das práticas, apresentadas na Tabela 4 e, considerando as evidências e fraquezas encontradas, esse resultado esperado é classificado como **largamente implementado**.

4.1.2 Resultado esperado V&V2

O resultado esperado V&V2 é: "Procedimentos e material de apoio são definidos, mantidos atualizados e usados para preparação e realização de revisões por pares" (SOFTEX, 2021b). Logo, para atestar o alcance desse resultado no Doarti, é necessária a apresentação de evidências que demonstrem a existência de tais procedimentos e materiais de apoio.

Contudo, esse resultado esperado é considerado como **não implementado**, visto que não foram encontradas, durante a avaliação, evidências de que revisões por pares sejam realizadas no Doarti.

4.1.3 Resultado esperado V&V3

O resultado esperado V&V3 é: "Métodos, procedimentos, critérios e ambientes são definidos, mantidos atualizados e usados durante as atividades de teste com fins de verificação e validação"(SOFTEX, 2021b). Portanto, para alcance desse resultado, é necessária a apresentação de evidências que demonstrem a existência de tais métodos, procedimentos, critérios e ambientes, responsáveis por guiar as atividades de teste do Doarti.

Para esse resultado esperado foram encontradas evidências diretas (ED), indiretas (EI) e fraquezas que serão listadas e detalhadas a seguir.

- EI - Foi verificada a existência de ambientes distintos, de desenvolvimento e produção para o aplicativo móvel do Doarti.
- ED - O documento "Abordagem e Procedimentos no Desenvolvimento de Software DOARTI", também utilizado como evidência direta para o resultado esperado V&V1, define o procedimento para realização de testes manuais.

Como fraqueza, foi identificada apenas a falta da definição de um procedimento para testes automatizados, justificando, novamente, a necessidade do foco nesse nível de testes por parte da estratégia de testes a ser elaborada para o Doarti. Como sugestão de melhoria a ser incluída nos estudos para elaboração dessa estratégia de testes, é considerada a realização de treinamentos e o desenvolvimento de documentos para auxílio, relacionados a testes automatizados no aplicativo móvel do Doarti.

De acordo com as regras para definição do grau de implementação das práticas, descritas na Tabela 4 e, considerando as evidências e fraquezas encontradas, esse resultado esperado é classificado como **parcialmente implementado**. Apesar da presença de evidências diretas e indiretas relacionadas ao procedimento de testes manuais, não foram encontradas evidências relacionadas ao método de realização de testes automatizados, que consiste em parte essencial ao processo de testes em geral.

4.1.4 Resultado esperado V&V4

O resultado esperado V&V4 é: "Atividades de verificação e validação são realizadas e problemas identificados são tratados"(SOFTEX, 2021b). Então, para atestar o alcance desse resultado é necessário apresentar evidências que demonstrem que os testes são realizados e que os problemas encontrados são tratados pela equipe do Doarti.

Para esse resultado, foi encontrada apenas uma evidência indireta e fraquezas, listadas e detalhadas a seguir.

- EI - Foi verificada a existência, no ambiente de planejamento do Doarti, de demandas concluídas em releases passadas que, segundo o documento "Abordagem e Procedimentos no Desenvolvimento de Software DOARTI", também listado como ED para os resultados V&V1 e V&V3, passaram por testes para serem consideradas concluídas.

Como fraqueza, foi identificada a falta da documentação relacionada aos casos de testes concluídos nas atividades de testes manuais, portanto, não é possível garantir a execução adequada desses testes, nem verificar seus detalhes. Também foi identificada como fraqueza a falta de atividades de rastreamento de falhas encontradas no processo de V&V.

Como sugestão de melhoria a ser abordada na estratégia de testes do Doarti, é considerada a adoção de um método de rastreamento de falhas e a criação de um modelo adequado à realidade do Doarti para documentação dos casos de teste.

De acordo com as regras descritas na Tabela 4 e, considerando a evidência e fraquezas encontradas, esse resultado esperado é classificado como **parcialmente implementado**.

4.1.5 Resultado esperado V&V5

O resultado esperado V&V5 é: "Os resultados das atividades de verificação e validação são analisados, registrados e comunicados"(SOFTEX, 2021b). Portanto, são necessárias evidências que comprovem tais análises, comunicações e registros nas atividades de teste do Doarti.

No entanto, não foram encontrados comunicados, registros e análises a cerca dos resultados de tais atividades no Doarti, caracterizando esse resultado esperado como **não implementado**.

4.2 Resultados de atributos de processo para o nível CP-G

Além da realização da verificação dos resultados esperados V&V1 a V&V5, também são realizadas verificações dos resultados esperados no nível G de capacidade de processos.

Os resultados esperados para o nível G de capacidade de processos são listados de CP-G1 a CP-G3, considerando, nesta avaliação, o processo de verificação e validação.

Nas subseções seguintes serão explicados cada um desses resultados esperados e apresentadas as evidências encontradas.

4.2.1 Resultado de atributo de processo CP-G1

O resultado de atributo de processo CP-G1 é: "O processo produz os resultados definidos" (SOFTEX, 2021a). Para que esse resultado seja alcançado, é necessário verificar se todos os demais resultados esperados para o processo de V&V, no caso os resultados V&V1 a V&V5, são satisfeitos.

Como foi exposto nas seções anteriores, apenas o resultado esperado V&V1 apresentou o grau de largamente implementado, ainda com falhas substanciais. Os demais resultados esperados possuem graus de implementação inferiores ou não são implementados. Portanto, não há evidências de que o processo de V&V produz os resultados definidos, concluindo, assim, a caracterização **não implementado** para o resultado esperado CP-G1.

4.2.2 Resultado de atributo de processo CP-G2

O resultado de atributo de processo CP-G2 é: "A execução do processo é planejada e monitorada" (SOFTEX, 2021a). Para que esse resultado seja alcançado, é necessário verificar a existência de evidências que indiquem que o processo de testes do Doarti é planejado e monitorado.

Uma fraqueza e uma evidência direta (ED) foram encontradas e serão listadas e explicadas a seguir.

- ED: O documento "Abordagem e Procedimentos no Desenvolvimento de Software DOARTI", também utilizado como ED para os resultados V&V1, V&V3 e V&V4, apresenta um trecho indicando o planejamento dos testes manuais, apontando a etapa na qual devem ser realizados.

Como fraqueza, foi observada a falta de monitoramento das atividades de teste. Como sugestão de melhoria a ser inclusa na estratégia de testes do Doarti, é considerada a adoção de um processo de integração contínua para monitoramento dos testes automatizados.

Com uma fraqueza e uma evidência direta presentes, seguindo as regras para definição do grau de implementação do MPS.BR, apresentado na Tabela 4, o CP-G2 é considerado **largamente implementado**

4.2.3 Resultado de atributo de processo CP-G3

O resultado de atributo de processo CP-G3 é: "As pessoas estão preparadas para executar suas responsabilidades no processo." (SOFTEX, 2021a). Para o alcance desse

resultado, é necessária a verificação de evidências que comprovem que a equipe do Doarti é capacitada para realização dos testes.

Apesar da existência de um documento contendo o roteiro de atividades de preparação e avaliação de novos membros, não foram identificadas, neste documento, atividades relacionadas a testes de software. Assim, pode ser considerada a inclusão, na estratégia de testes do Doarti, da realização de treinamentos para capacitação de novos membros, incluindo atividades relacionadas a testes manuais e automatizados.

Por conta da falta de evidências que indiquem a preparação da equipe em relação aos testes, o CP-G3 é caracterizado como **não implementado**.

4.3 Conclusão da avaliação inicial

De acordo com a SOFTEX (2021a), após a realização da avaliação, para concluir um processo e um nível de capacidade como **satisfeito**, é necessário que todos os resultados esperados, tanto do processo, quanto do nível de capacidade, nesse caso o CP-G, sejam caracterizados como **totalmente implementado** ou **largamente implementado**.

Como explicado nas seções anteriores, os resultados esperados para o processo de V&V apresentaram as seguintes caracterizações:

- V&V1: Largamente implementado.
- V&V2: Não implementado.
- V&V3: Parcialmente implementado.
- V&V4: Parcialmente implementado.
- V&V5: Não implementado.

Já em relação à capacidade de processos, os resultados esperados para o nível CP-G foram caracterizados, nesta avaliação, como:

- CP-G1: Não implementado.
- CP-G2: Largamente implementado.
- CP-G3: Não implementado.

Visto que grande parte dos resultados esperados se encontram caracterizados como parcialmente ou não implementados, considera-se que o processo de verificação e validação do Doarti não satisfaz o nível G de capacidade de processos do MPS.BR.

Assim, é alcançado o resultado da segunda fase deste trabalho, identificando o estado inicial do processo de V&V do Doarti e encontrando oportunidades de melhorias acerca das fraquezas encontradas.

5 Estratégia de testes

Como resultado das fases de elaboração da estratégia de testes e validação da estratégia detalhadas no Capítulo 2, foi desenvolvido um modelo de processo de testes para o Doarti, amparado por documentos de auxílio e métricas para acompanhamento de sua execução. Também foi conduzida uma validação dessa estratégia, através de entrevistas com questionários. Tanto a definição da estratégia quanto sua validação são apresentadas nas seções seguintes.

5.1 Definição da estratégia

Como resultado da avaliação informal do processo de V&V do Doarti, detalhada na Seção 4, foram encontradas fraquezas relevantes relacionadas ao processo de testes. Portanto, a estratégia de testes deste trabalho propõe-se a explorar as oportunidades de melhoria relacionadas a tais fraquezas, considerando as características da equipe e do projeto.

Uma das fraquezas encontradas na avaliação informal do processo de V&V foi a falta de testes automatizados. Além disso, visto que a equipe de desenvolvimento do Doarti é suscetível à alta taxa de rotatividade de membros, como descrito na Seção 3.1, percebe-se que a utilidade dos testes de regressão é muito bem-vinda no projeto.

Segundo Sommerville (2019), testes de regressão consistem na reexecução de testes já existentes, com o objetivo de verificar se novas alterações no código não inseriram novos *bugs* no sistema. Sommerville (2019) também cita a utilidade da rapidez inerente aos testes automatizados quando se trata de testes de regressão. Executar todos os testes já existentes, após cada alteração significante do código, de forma manual, seria algo muito mais dispendioso.

Uma outra fraqueza identificada na avaliação informal foi a falta de documentação referente ao processo de testes. Portanto, a estratégia de testes proposta neste trabalho foi devidamente documentada e apresenta-se como uma forma de enfrentamento a essa fraqueza. Na Figura 9, é apresentado fluxograma da estratégia criada a qual contém atividades e todos os artefatos gerados. Esse fluxo de atividades deve ser iniciado juntamente ao início da *Sprint*, em que são utilizados como entrada, as demandas de implementação ou refatoração do código do aplicativo móvel do Doarti. Essas demandas estão presentes no *Backlog* da *Sprint*, no qual já se encontram pontuadas em relação à suas complexidades, preenchendo, dessa forma, a métrica CIT, detalhada, assim como as demais métricas, no Apêndice B.

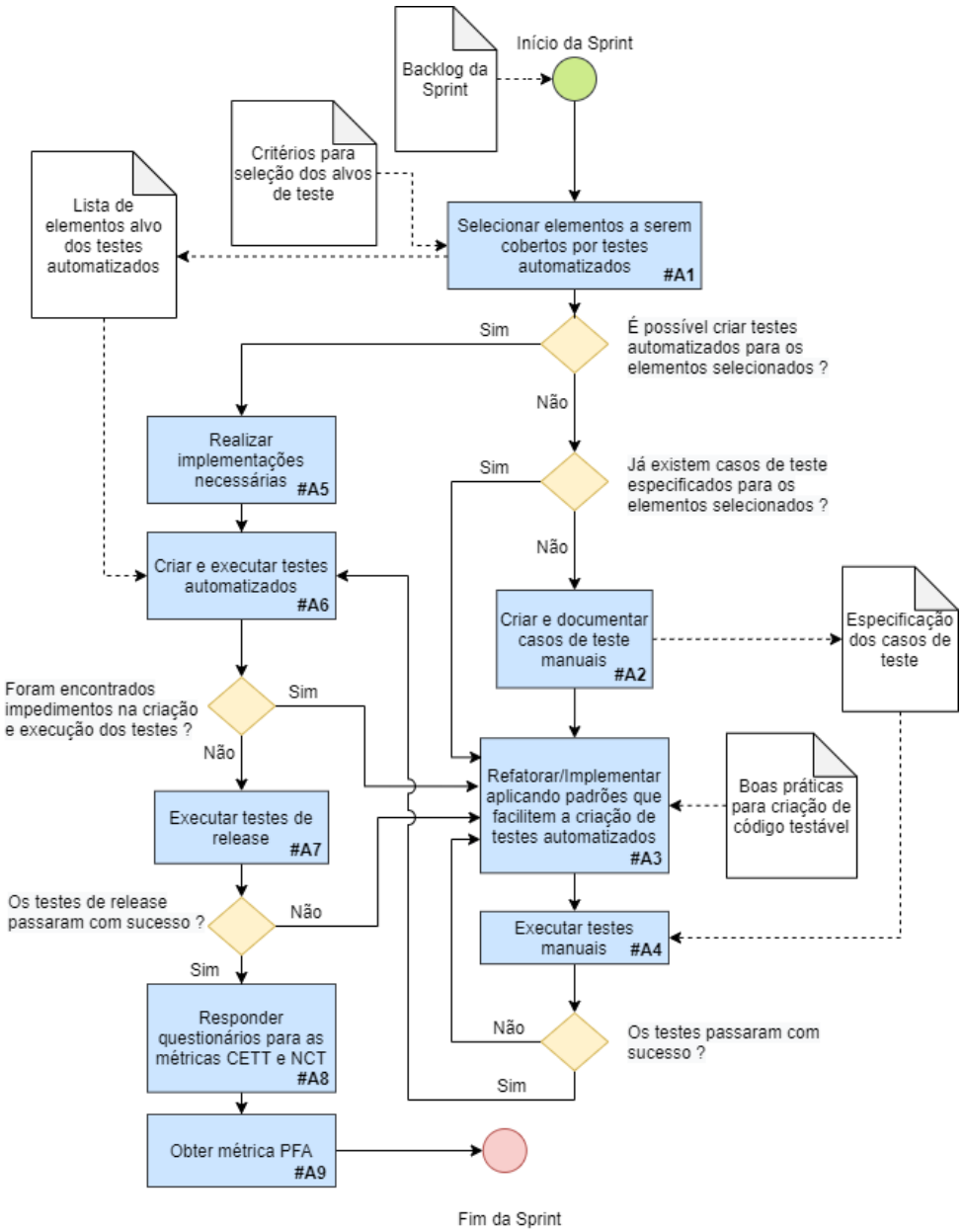

Figura 9 – Fluxograma da estratégia de testes.

5.1.1 Detalhamento das atividades

Nas Tabelas 5 a 13 são detalhados os objetivos, procedimentos, entradas, saídas e participantes de cada atividade presente na estratégia de testes ilustrada na Figura 9. Alguns elementos estão representados em forma de links, acompanhados da especificação da seção onde se encontram, para facilitar a navegação no documento.

É importante ressaltar que os testes automatizados, citados ao longo desta estratégia, representam testes unitários, de componente e de integração, a serem feitos para o aplicativo móvel do Doarti, na linguagem de programação Dart.

Tabela 5 – Atividade #A1

Atividade #A1: Selecionar elementos a serem cobertos por testes automatizados	
Objetivos	- Filtrar elementos alvo dos testes automatizados
Procedimentos	- Definir quais classes métodos e componentes serão alvo de testes unitários de componentes ou de integração.
Entradas	**Documentos:** - *Backlog* da *sprint* (Seção 5.1.2.2). - Considerações acerca da seleção dos elementos alvos de teste (Seção 5.1.2.3).
Saídas	- Lista de elementos que serão testados (Seção 5.1.2.4).
Participantes	- Desenvolvedores responsáveis pela demanda.

Tabela 6 – Atividade #A2

Atividade #A2 Criar e documentar casos de teste manuais	
Objetivos	- Definir expectativas de funcionamento para cenários mais comuns de uso sistema, envolvendo os elementos a serem testados.
Procedimentos	- Identificar casos de uso do sistema que envolvam os elementos alvo dos testes. - Definir resultados esperados. - Definir passos para execução dos testes manuais.
Entradas	**Documentos:** - Elementos alvo dos testes automatizados selecionados (Seção 5.1.2.4). **Condições:** - É impossível ou muito dispendioso criar testes automatizados para os elementos selecionados na Atividade #A1 **e** - Ainda não existirem casos de testes manuais especificados, que envolvam os elementos selecionados na Atividade #A1.
Saídas	- Especificação de casos de teste (Seção 5.1.2.6).
Participantes	- Desenvolvedores responsáveis pela demanda. - Testadores (para auxílio, caso necessário).

Tabela 7 – Atividade #A3

Atividade #A3 Refatorar/implementar aplicando padrões que facilitem a criação de testes automatizados	
Objetivos	- Implementar as demandas da *sprint*. - Manter ou aumentar a testabilidade do código.
Procedimentos	- Escrever código referente à demanda da *sprint*, atentando-se as boas práticas de escrita de código testável.
Entradas	**Documentos:** - Especificação de boas práticas de escrita de código testável (Seção 5.1.2.5). **Condições:** - É impossível ou muito dispendioso criar testes automatizados para os elementos selecionados na Atividade #A1 **e** - Já existem casos de testes manuais especificados para os elementos selecionados na Atividade #A1 **ou** - Impedimentos foram encontrados na criação de testes durante a Atividade #A6 **ou** - Os testes executados na Atividade #A4 não passaram com sucesso.
Saídas	- Demanda pronta para execução de testes manuais.
Participantes	- Desenvolvedores responsáveis pela demanda.

Tabela 8 – Atividade #A4

Atividade #A4 Executar testes manuais	
Objetivos	- Verificar se o funcionamento do sistema foi mantido após as implementações e/ou refatorações realizadas.
Procedimentos	- Executar os testes manuais especificados para o conjunto de elementos envolvidos na Atividade #A3.
Entradas	**Condições:** - Demandas prontas para execução de testes manuais.
Saídas	- Demanda pronta para criação de testes automatizados.
Participantes	- Desenvolvedores responsáveis pela demanda. - Testadores (para auxílio, caso necessário).

Tabela 9 – Atividade #A5

Atividade #A5 Realizar implementações necessárias	
Objetivos	- Implementar as demandas da *sprint*.
Procedimentos	- Escrever código referente à demanda da *sprint*.
Entradas	**Condições:** - É possível criar testes automatizados para os elementos selecionados na Atividade #A1.
Saídas	- Demanda pronta para criação de testes automatizados.
Participantes	- Desenvolvedores responsáveis pela demanda. - Testadores (para auxílio, caso necessário).

Tabela 10 – Atividade #A6

Atividade #A6 Criar e executar testes automatizados	
Objetivos	- Aumentar a cobertura de testes do código.
Procedimentos	- Escrever e executar código referente aos testes automatizados.
Entradas	**Documentos:** - Lista de elementos alvo dos testes automatizados (Seção 5.1.2.4). **Condições:** - Demanda pronta para criação de testes automatizados.
Saídas	- Elementos alvo de testes devidamente cobertos por testes automatizados.
Participantes	- Desenvolvedores responsáveis pela demanda. - Testadores (para auxílio, caso necessário).

Tabela 11 – Atividade #A7

Atividade #A7 Executar testes de release	
Objetivos	- Verificar o correto funcionamento das novas implementações, integradas ao restante do sistema para o lançamento da próxima versão.
Procedimentos	- Executar casos de testes interessantes, na visão do testador, capazes de avaliar o correto funcionamento do sistema como um todo, ou de encontrar prováveis falhas.
Entradas	**Condições:** - Não foram encontrados impedimentos na criação e execução dos testes automatizados, na Atividade #A6.
Saídas	- Demandas da *sprint* finalizadas.
Participantes	- Testadores.

Tabela 12 – Atividade #A8

Atividade #A8 Responder questionários para as métricas CETT e NCT	
Objetivos	- Coletar dados para as métricas CETT e NCT.
Procedimentos	- Responder os questionários referentes às métricas CETT e NCT (Seção 5.1.2.1).
Entradas	**Condições:** - Os testes de *release* passaram com sucesso.
Saídas	Métricas CETT e NCT coletadas (Seção 5.1.2.1).
Participantes	- Desenvolvedores. - Testadores.

Tabela 13 – Atividade #A9

Atividade #A9 Obter métrica PFA	
Objetivos	- Coletar métricas.
Procedimentos	- Realizar reunião entre desenvolvedores e testadores, com o objetivo dos desenvolvedores validarem as falhas encontradas pelos testadores durante os testes de *release*.
Entradas	**Condições:** - Questionários CETT e NCT respondidos (Apêndice E).
Saídas	- Métrica PFA coletada (Seção 5.1.2.1).
Participantes	- Desenvolvedores. - Testadores.

5.1.2 Detalhamento dos documentos

Ao longo do processo de testes ilustrado na Figura 9, são utilizados e produzidos alguns documentos contendo dados, tanto como entrada quanto como saída de atividades. Cada um é explicado com mais detalhes ao longo desta subseção.

5.1.2.1 Métricas

Como resultado da fase de contextualização, foram selecionadas métricas com o objetivo de auxiliar no acompanhamento dos impactos da aplicação da estratégia de testes no Doarti.

Dentre as sub-características de manutenibilidade relacionadas à qualidade de produto de software, apresentadas na Seção 3.2, foi selecionada a testabilidade como alvo de medição, por conta da temática deste trabalho, que é voltada a testes de software.

Foram selecionados dois conjuntos de métricas. Seus respectivos objetivos, questões e observações são detalhados no Apêndice B.

5.1.2.2 *Backlog* da *Sprint*

Este é um documento já existente no processo atual de desenvolvimento do Doarti. Nele estão inclusas as demandas que foram priorizadas para a *sprint* atual, pontuadas

previamente pela equipe do Doarti, em conjunto, em relação a suas complexidades.

As pontuações variam entre 1 e 13, seguindo a sequência de Fibonacci, em que o menor valor representa a menor complexidade na visão da equipe.

5.1.2.3 Considerações acerca da seleção dos elementos alvos de teste

O Apêndice D descreve algumas considerações importantes a serem feitas no momento de escolherem os casos de teste automatizados que serão escritos e os elementos neles envolvidos.

O objetivo deste documento é auxiliar o desenvolvedor na tomada de decisão sobre o que testar e o que não testar, para que o tempo gasto em uma demanda seja otimizado.

5.1.2.4 Lista de elementos alvo dos testes automatizados

Este documento é proveniente da primeira atividade da *sprint*. Após uma análise dos critérios para seleção dos alvos de teste, são listados neste documento todos os elementos de código que irão receber testes automatizados durante a *sprint* atual.

Supondo um exemplo no aplicativo Doarti, após atentar-se às considerações acerca da seleção dos elementos alvos de teste, o desenvolvedor conclui que é interessante testar o caso no qual um usuário realiza uma doação anônima a uma entidade no Doarti. Nessa ocasião, com o caso e objetivo de teste definidos, cabe ao desenvolvedor encontrar e listar os elementos do código pelos quais o objetivo de teste percorre.

Um exemplo dessa lista, considerando este caso fictício descrito utilizando o aplicativo Doarti é apresentado na Tabela 14.

Tabela 14 – Exemplo de lista de elementos alvo de testes

Elementos alvo de testes para o caso "Doação anônima de usuário para entidade"
- Classe Usuário - Formulário de doação anônima - Classe de lógica de negócio envolvendo doação anônima

5.1.2.5 Especificação de boas práticas para escrita de código testável

Este documento apresenta um conjunto de práticas para escrita de código testável, baseadas em exemplos detalhados por Wolter, Ruffer e Hevery (2009). Tais práticas são apresentadas no Apêndice C de forma resumida e acompanhadas de exemplos elaborados na linguagem Dart, utilizada no aplicativo Doarti.

Para essa estratégia de testes, é recomendado que os desenvolvedores se atentem a tais práticas.

5.1.2.6 Especificação dos casos de teste

Este documento apresenta detalhes para execução de testes manuais, com objetivo de verificar o funcionamento dos elementos presentes na demanda em questão. Caso não existam especificações de casos de teste para demanda em questão, elas são escritas durante a Atividade #A2. Caso essas especificações já existam, o desenvolvedor deve pular a Atividade #A2, indo direto para a Atividade #A3.

Na Atividade #A4, este documento, seja ele recém escrito, ou já existente proveniente de *sprints* anteriores, é utilizado para guiar a execução dos testes de uma demanda, após a finalização de suas respectivas implementações.

Um modelo a ser seguido para elaboração deste documento se encontra no Apêndice F.

5.2 Validação da Estratégia

Após a definição da estratégia de testes, foi realizada sua validação. Esta validação foi conduzida por meio de entrevistas nas quais foi respondido um questionário apresentado no Capítulo 2. As entrevistas foram realizadas através de videochamadas, com dois desenvolvedores membros da equipe do Doarti e com duração média de 20 minutos.

Nas próximas seções são apresentadas as respostas de cada membro, de forma anônima, agrupadas por característica de qualidade de modelo de processos, seguidas de uma análise do resultado desta validação.

5.2.1 Adequabilidade

A adequabilidade, sub-característica de funcionalidade, representa a capacidade do processo definido de produzir resultados esperados, de forma apropriada, considerando-se as limitações, recursos e contexto do projeto (KROEGER; DAVIDSON; COOK, 2014).

Essa sub-característica é representada pela questão Q1- Você acredita que seria necessário muito esforço para adequar a equipe para a execução desta estratégia? Diante dessa questão, os membros entrevistados responderam:

- Não, achei simples. Acredito que não seria necessário muito trabalho para adequação, pois a equipe já teve contato, mesmo que introdutório, com testes.
- Não. O único e maior esforço seria preparar a equipe para trabalhar com testes, possivelmente recebendo ajuda de membros mais experientes.

Dessa forma, considerando as repostas dos membros da equipe, conclui-se que a estratégia proposta é adequada. Porém a existência de muitas práticas relacionadas a

testes de software, trará um esforço adicional de preparação dos membros em relação a este tema.

5.2.2 Completude

A completude, sub característica de semântica, diz respeito a presença, de forma correta, de todos os elementos relevantes do domínio em questão (KITCHENHAM et al., 2005).

A questão Q2 - Você sentiu falta da explicação de algum elemento presente na estratégia? diz respeito a esta característica. As respostas obtidas para essa questão foram:

- Sim. Senti falta de uma explicação sobre o modo pelo qual seriam feitas as capacitações da equipe, principalmente aquelas relacionadas às competências necessárias para criação de testes automatizados.
- Não.

Percebe-se que as respostas foram opostas uma da outra. Dessa forma, pode-se concluir, mais uma vez, que é necessário definir detalhes sobre como capacitar dos membros para trabalhar com testes na estratégia.

5.2.3 Acessibilidade

A acessibilidade, sub-característica de usabilidade é definida como a facilidade dos usuários, no caso deste trabalho os membros da equipe do Doarti, em encontrar informações sobre o processo proposto (KROEGER; DAVIDSON; COOK, 2014).

As questões que refletem essa subcracteristicas foram Q3, Q4 e Q5. Foram obtidas as seguintes respostas obtidas para a questão Q3 - Você conseguiu encontrar com facilidade explicações referentes à todos os elementos citados no modelo de processo apresentado ?

- Sim.
- Sim. A figura ilustrando o processo e as referências facilitaram. A leitura a partir do índice também ajudou muito.

Já para a questão Q4 - A disposição dos textos e documentos explicativos sobre a estratégia permitiram que o fluxo de leitura seguisse o fluxo cronológico das atividades?, as respostas foram:

- Sim. Deu para relacionar bem a ordem cronológica dos textos, principalmente com ajuda da figura ilustrando o processo.

- Sim. Porém a leitura do detalhamento de alguns documentos inerentes ao processo quebrou um pouco o fluxo de leitura.

Por fim, diante da última questão relacionada à acessibilidade: Q5 - A navegação entre os textos e documentos explicativos sobre a estratégia é fácil ?, os entrevistados responderam:

- Foi bem fácil navegar para os documentos referenciados, entretanto, o caminho de "volta" dessa navegação nem sempre foi simples.
- Não cheguei a navegar no documento por meio dos links.

A partir das respostas apresentadas, conclui-se que a acessibilidade do documento referente à estratégia de testes proposta foi relativamente bem avaliada, porém, ainda com pequenos defeitos de navegabilidade e disposição de textos a serem corrigidos.

É importante ressaltar que foi entregue aos participantes um documento no formato pdf composto pela Seção 5.1, em conjunto com os Apêndices B, C, D, E e F.

5.2.4 Validade

A validade, também sub característica de semântica, é definida como a apresentação correta de afirmações que sejam relevantes ao contexto do processo (KITCHENHAM et al., 2005).

Frente à questão Q6: Existe algum termo citado nos documentos referentes à apresentação da estratégia que você não conheça? Os entrevistados responderam:

- Sim. Não sei o que são testes de release.
- Não.

Apesar de divergentes, as repostas indicam novamente uma possível lacuna na estratégia desenvolvida, referente à capacitação da equipe relacionada a testes de software.

5.2.5 Adaptabilidade

A adaptabilidade, sub característica de usabilidade, representa a facilidade do processo em se adaptar a diversas situações possíveis no contexto do projeto para qual foi definido (KROEGER; DAVIDSON; COOK, 2014).

Diante da questão Q7: Você consegue imaginar alguma situação do contexto do Doarti não prevista pela estratégia proposta? As respostas dos entrevistados foram:

- Não, pois diversos tipos de testes foram explorados.
- Demandas que levam mais de uma sprint poderiam ser melhor explicadas no fluxo de atividades.

Apesar de uma das respostas demonstrar dúvida quanto ao cenário no qual uma demanda ocupa mais de uma *sprint*, tal cenário não deve ser tratado no processo de testes, mas no processo de construção da demanda que, seguindo princípios da metodologia ágil, não deve exigir mais de uma *sprint* para ser finalizada. Portanto, não são necessários ajustes na estratégia de testes em relação a este tema.

5.2.6 Compreensibilidade

A compreensibilidade, sub característica de usabilidade é definida como a capacidade dos usuários em entender o propósito e a relevância do processo (KROEGER; DAVIDSON; COOK, 2014).

Frente à questão Q8: Quais resultados você acredita que a aplicação dessa estratégia pode trazer?, os entrevistados responderam:

- Acredito que evitaria erros no futuro. Se houvesse cobertura de testes no código, um erro de acesso indevido de dados ocorrido recentemente provavelmente não teria acontecido.
- Acredito que proporcionaria alguns benefícios como: produção de códigos melhores; maior segurança em relação ao código produzido; e maior segurança em relação à rotatividade dos membros, visto que o código deixado por membros anteriores estaria já devidamente testado.

Diante dessas respostas, percebe-se que a estratégia desenvolvida apresentou uma boa compreensibilidade para os membros entrevistados. Ambos reconheceram benefícios possíveis e alinhados com realização de testes de software no projeto.

5.3 Conclusão da validação da estratégia de testes

A partir da análise das respostas referentes a cada uma das características de qualidade envolvidas na validação, é possível identificar alguns pontos de melhoria. Dentre as oportunidades de melhoria identificadas, encontram-se melhorias na navegabilidade do documento, glossário de termos técnicos relacionados a testes, e principalmente, o planejamento de atividades de capacitação da equipe em relação a testes de software.

É importante ressaltar que os membros participantes das entrevistas se encontram no início do curso de Engenharia de software da Universidade de Brasília, o que justifica a preocupação com termos técnicos e capacitações relacionadas a testes de software. Tais preocupações não tem reflexo direto na estratégia proposta neste trabalho, porém mostram a necessidade de se prever e planejar ações de treinamento com os membros da equipe em um momento prévio a aplicação da estratégia proposta.

6 Considerações finais

A elaboração de uma estratégia de testes para a equipe do Doarti, reconhecida como uma equipe composta por voluntários geograficamente distribuídos, é iniciada com a conclusão das duas fases iniciais de preparação, apresentadas no Capítulo 2. Essas fases preparam o ambiente para a execução das duas últimas fases, também detalhadas no Capítulo 2, cujo objetivo foi propor uma estratégia de testes e validá-la com seus usuários e principais interessados: os desenvolvedores membros do Doarti.

É encontrado, no primeiro momento, o desafio de conectar os fundamentos relacionados à qualidade e a processos de software, com a teoria pesquisada, referente a equipes com características tão impactantes, como a distribuição geográfica, assincronia de trabalho e o regime de serviço voluntariado.

Como enfrentamento a esse desafio, foram pesquisados trabalhos que tratassem de tais perfis de equipes, bem como os padrões e normas estabelecidos para modelos de qualidade de software e de processos. Em seguida, foram relacionados conceitos ligados ao perfil da equipe, aos conceitos de qualidade de produto de software, mais especificamente, a manutenibilidade, presente no modelo de qualidade de produto de software proposto na série de normas SQuaRE.

Inicialmente foi realizada uma avaliação, seguindo os modelos do MPS.BR, para julgar o estado atual do processo de V&V do Doarti. Conforme descrito no Capítulo 4, tal processo no Doarti pode ser considerado como não-gerenciado, por não atender aos requisitos do nível G de capacidade de processos, o mais básico do MPS.BR. Por meio desta avaliação foi possível atingir o OE1, listado na Tabela 1, pois foram explorados e descobertos artefatos, práticas e fraquezas do processo de V&V do Doarti.

Já na reta final do trabalho, o principal desafio foi adaptar o referencial teórico encontrado sobre estratégias e boas práticas relacionadas a testes de software, ao contexto específico do Doarti.

Como enfrentamento a esse desafio, foi necessário olhar para a realidade da equipe e para o próprio código do aplicativo do Doarti, de modo a elaborar uma estratégia o mais adaptada possível ao contexto do Doarti, alcançando assim o OE2, listado na Tabela 1.

Foram estudadas também, como forma de auxílio e inspiração na elaboração da estratégia proposta neste trabalho, outras abordagens de teste existentes. Foram apresentadas apenas duas, dentre outras encontradas, sendo elas: Cow_Suite e Algorítimo de ordem de teste, descritos na Seção 3.7, por apresentarem pontos interessantes para o contexto deste trabalho. O Cow_Suite e o Algorítimo de ordem de teste apresentam

abordagens criteriosas para duas atividades específicas, que são a seleção dos casos de teste e a reutilização de casos de teste já existentes. Inspiradas nessas duas abordagens, essas atividades também foram tratadas na estratégia proposta neste trabalho e estão detalhadas no Capítulo 5.1.

Como parte da estratégia, foram elaboradas e selecionadas métricas, descritas no Apêndice B, planejadas para auxiliar na condução de uma análise, em caso de aplicação, da estratégia de testes proposta, medindo atributos da qualidade interna e externa relacionados à manutenibilidade do produto de software desenvolvido no Doarti. Essas métricas, juntamente à avaliação inicial de processo realizada, constituem uma base para tornar possível eventuais avaliações de efeitos positivos e negativos da estratégia de testes, em caso de sua aplicação.

Também como parte da estratégia, foram desenvolvidos documentos de auxílio, descritos na Seção 5.1.2. Cada um desses documentos foi elaborado com propósitos específicos: auxiliar na seleção dos casos de teste; apresentar boas práticas de escrita de código de modo a melhorar a testabilidade do código do Doarti; fornecer modelos para guiar os envolvidos na produção de documentos previstos na estratégia.

Esses documentos foram elaborados por conta do perfil da equipe do Doarti. Alguns membros, incluindo os que participaram da fase de validação, encontram-se no início do curso de Engenharia de Software da Universidade de Brasília; portanto, foi importante fornecer documentos que tornassem algumas etapas da estratégia mais objetivas, guiando de forma mais detalhada as ações dos envolvidos.

Por fim, alcançando o OE3, também listado na Tabela 1, como forma de validar se a estratégia desenvolvida foi realmente bem adaptada ao contexto do Doarti, foram realizadas entrevistas com os desenvolvedores membros da equipe. Tal validação concluiu, a partir de características de qualidade de processos, como a usabilidade e funcionalidade, que a estratégia desenvolvida se mostrou de fácil entendimento, porém com falhas pequenas referentes à acessibilidade. Também foram reconhecidas falhas consideráveis relacionadas a previsão de capacitações da equipe em relação a testes de software.

6.1 Trabalhos futuros

Com todas as fases do trabalho devidamente finalizadas, é necessário olhar para as falhas encontradas e possíveis oportunidades de trabalhos futuros.

Em relação à estratégia de testes proposta, existe uma grande oportunidade de exploração de mais exemplos de escrita de código testável, assim como os que foram expostos no Apêndice C. Também poderia ser avaliada a possibilidade do uso de ferramentas de análise estática pra verificação da aplicação correta das boas práticas especificadas no

Apêndice C. Durante a fase de validação da estratégia, este foi um documento que despertou bastante interesse dos entrevistados por seu nível de detalhe.

Em relação as abordagens e conceitos de testes de software explorados na estratégia, é interessante avaliar a possibilidade da inclusão de testes não funcionais e utilização de TDD, em um cenário onde a equipe se encaixe num perfil de maior maturidade.

Como consequência da fase de validação, foram expostos pontos de falha da estratégia proposta, um deles em especial, refere-se à necessidade de se elaborar um planejamento de capacitação para equipe na área de testes de software.

Também relacionado a validação da estratégia proposta, é reconhecida a importância da abordagem de mais características de qualidade de processos durante a validação. Silva (2014) apresenta, em seu trabalho de graduação, outras três características de qualidade que poderiam, caso houvesse tempo hábil, revelar muito mais oportunidades de melhoria acerca da estratégia validada.

Referências

ANDRADE, A. W.; AGRA, R.; MALHEIROS, V. Estudos de caso de aplicativos móveis no governo brasileiro. In: SBC. *Anais do IX Simpósio Brasileiro de Sistemas de Informação*. [S.l.], 2013. p. 780–791. Citado na página 21.

BASANIERI, F. et al. An automated test strategy based on uml diagrams. In: *Ericsson Rational User Conference. Upplands Vasby Sweden*. [S.l.: s.n.], 2001. Citado 2 vezes nas páginas 44 e 45.

BECK, K. et al. *Manifesto for Agile Software Development*. 2001. Disponível em: <http://www.agilemanifesto.org/>. Citado na página 35.

BRASIL. Lei nº 9.608, de 18 de fevereiro de 1998. 1998. Disponível em: <http://www.planalto.gov.br/ccivil_03/leis/l9608.htm>. Citado na página 29.

BRASIL, C. G. da Internet no. *Pesquisa sobre o uso das tecnologias de informação e comunicação nos domicílios brasileiros: TIC Domicílios 2019*. [s.n.], 2020. Disponível em: <https://cetic.br/media/docs/publicacoes/2/20201123121817/tic_dom_2019_livro_eletronico.pdf>. Citado na página 21.

BRASíLIA, U. de. Projeto doarti: Desenvolvimento de solução tecnológica para potencializar o processo de doações no df. 2020. Disponível em: <https://tinyurl.com/Doarti>. Citado 2 vezes nas páginas 21 e 22.

CALDIERA, V. R. B. G.; ROMBACH, H. D. The goal question metric approach. *Encyclopedia of software engineering*, p. 528–532, 1994. Citado 3 vezes nas páginas 11, 37 e 38.

ESPINOSA, A. et al. Shared mental models, familiarity, and coordination: A multi-method study of distributed software teams. *ICIS 2002 Proceedings*, p. 39, 2002. Citado na página 30.

GPS. *Manual de Produção de Software do Cercomp*. 2010. Disponível em: <https://files.cercomp.ufg.br/weby/up/18/o/manual.pdf>. Citado na página 89.

HARTMANN, G.; STEAD, G.; DEGANI, A. Cross-platform mobile development. *Mobile Learning Environment, Cambridge*, v. 16, n. 9, p. 158–171, 2011. Citado na página 21.

HOMÈS, B. *Fundamentals of software testing*. [S.l.]: John Wiley & Sons, 2013. Citado na página 42.

ISO/IEC. *ISO/IEC 25000. Software engineering - Software product Quality Requirements and Evaluation (SQuaRE) – Guide to SQuaRE*. [S.l.]: ISO/IEC, 2004. Citado 2 vezes nas páginas 31 e 32.

ISO/IEC. *ISO/IEC 25020. Software engineering - Software product quality requirements and evaluation (SQuaRE) - Measurement reference model and guide*. [S.l.]: ISO/IEC, 2006. Citado na página 34.

ISO/IEC. *ISO/IEC 25010. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) - System and software quality models.* [S.l.]: ISO/IEC, 2010. Citado 3 vezes nas páginas 11, 32 e 33.

ISO/IEC. *ISO/IEC 25023. Systems and software engineering - Systems and software Quality Requirements and Evaluation (SQuaRE) – Measurement of system and software product quality.* [S.l.]: ISO/IEC, 2011. Citado na página 34.

KITCHENHAM, B. A. et al. A framework for evaluating a software bidding model. *Information and Software Technology*, Elsevier, v. 47, n. 11, p. 747–760, 2005. Citado 2 vezes nas páginas 63 e 64.

KROEGER, T. A.; DAVIDSON, N. J.; COOK, S. C. Understanding the characteristics of quality for software engineering processes: A grounded theory investigation. *Information and Software Technology*, Elsevier, v. 56, n. 2, p. 252–271, 2014. Citado 4 vezes nas páginas 62, 63, 64 e 65.

KUNG, D. et al. A test strategy for object-oriented programs. In: IEEE. *Proceedings Nineteenth Annual International Computer Software and Applications Conference (COMPSAC'95).* [S.l.], 1995. p. 239–244. Citado na página 45.

MYERS, G. J.; SANDLER, C.; BADGETT, T. *The art of software testing.* [S.l.]: John Wiley & Sons, 2011. Citado na página 44.

NAIK, K.; TRIPATHY, P. *Software testing and quality assurance: theory and practice.* [S.l.]: John Wiley & Sons, 2011. Citado 4 vezes nas páginas 41, 42, 44 e 85.

OLIVEIRA, M. L. S. d. *O estudo de equipes de desenvolvimento de software na indústria: um mapeamento sistemático da literatura.* Dissertação (Mestrado) — Universidade Federal de Pernambuco, 2019. Citado na página 29.

PANTELI, N.; YALABIK, Z. Y.; RAPTI, A. Fostering work engagement in geographically-dispersed and asynchronous virtual teams. *Information Technology & People*, Emerald Publishing Limited, 2019. Citado 2 vezes nas páginas 30 e 31.

PIMENTEL, S. B. d. O. F. F. Mps.br – uma ferramenta brasileira na gestão por processos. p. 15, 2013. Disponível em: <https://softex.br/download/mps-br-uma-ferramenta-brasileira-na-gestao-por-processos/?wpdmdl=106293&masterkey=5f32bbabacb6a>. Citado 2 vezes nas páginas 11 e 40.

PRESSMAN, R. S. *Engenharia de software: uma abordagem profissional.* [s.n.], 2011. OCLC: 868913398. ISBN 978-85-8055-044-3. Disponível em: <http://site.ebrary.com/id/10765474>. Citado 2 vezes nas páginas 37 e 38.

RAMOS, L. *Scrum: O que é? Aprenda o conceito e como funciona.* 2019. Disponível em: <https://auditeste.com.br/scrum-o-que-e-aprenda-o-conceito-e-como-funciona/>. Citado 2 vezes nas páginas 11 e 37.

SILVA, R. M. da. *Qualidade na Modelagem de Processos de Software.* 2014. Citado 3 vezes nas páginas 27, 28 e 69.

SKOGLUND, A. G. Do Not Forget about Your Volunteers: A Qualitative Analysis of Factors Influencing Volunteer Turnover. *Health & Social Work*, v. 31, n. 3, p. 217–220, ago. 2006. ISSN 0360-7283, 1545-6854. Disponível em: <https://academic.oup.com/hsw/article-lookup/doi/10.1093/hsw/31.3.217>. Citado na página 30.

SOARES, M. D. S. Comparação entre Metodologias Ágeis e Tradicionais para o Desenvolvimento de Software. p. 6, 2004. Disponível em: <http://infocomp.dcc.ufla.br/index.php/infocomp/article/view/68>. Citado 3 vezes nas páginas 11, 34 e 35.

SOFTEX. *Guia Geral MPS de Software*. SOFTEX, 2020. Disponível em: <https://softex.br/download/mps-br-guia-geral-software-2020/>. Citado 5 vezes nas páginas 11, 38, 39, 40 e 41.

SOFTEX. *Guia de Avaliação: Processo e Método de Avaliação MA-MPS*. SOFTEX, 2021. Disponível em: <https://softex.br/download/guia-de-avaliacao-2021/>. Citado 4 vezes nas páginas 40, 41, 51 e 52.

SOFTEX. *Guia Geral MPS de Software*. SOFTEX, 2021. Disponível em: <https://softex.br/download/guia-geral-de-software-2021/>. Citado 5 vezes nas páginas 41, 47, 48, 49 e 50.

SOMMERVILLE, I. *Engenharia de software, 10ª edição*. São Paulo: Editora Pearson, 2019. ISBN 9788543024974. Citado 6 vezes nas páginas 21, 39, 42, 43, 55 e 85.

STARNES, B.; WYMER, W. Conceptual foundations and practical guidelines for retaining volunteers to serve in local nonprofit organizations. *Journal of Nonprofit & Public Sector Marketing*, v. 9, p. 97–118, 01 2001. Citado na página 30.

SUTHERLAND, J.; SCHWABER, K. Guia do scrum. *Um guia definitivo para o Scrum: As regras do jogo*, 2013. Citado na página 36.

WEBER, K. C. et al. Modelo de Referência para Melhoria de Processo de Software: uma abordagem brasileira. p. 16, 2015. Citado 4 vezes nas páginas 13, 38, 40 e 41.

WOLTER, J.; RUFFER, R.; HEVERY, M. *Guide: Writing testable code*. 2009. Citado na página 61.

WONG, S.-S.; BURTON, R. M. Virtual teams: what are their characteristics, and impact on team performance? *Computational & Mathematical organization theory*, Springer, v. 6, n. 4, p. 339–360, 2000. Citado na página 21.

Apêndices

APÊNDICE A – Cronograma

Esta seção apresenta o cronograma de atividades executadas, incluindo todas as fases do trabalho. As atividades, e suas respectivas datas de execução são apresentadas na Tabela 15.

Tabela 15 – Cronograma executado

Fase	**Atividades realizadas**	**Período**
Contextualização	Revisão de literatura.	23/08/2020 a 12/12/2020
Contextualização	Definição da estrutura do trabalho.	30/08/2020 a 26/09/2020
Contextualização	Leitura da ISO 9126, e definição do objetivo e questões da revisão de literatura.	30/08/2020 a 05/09/2020
Contextualização	Pesquisa sobre métricas para o processo de testes de software, MPS.BR e SQuaRE.	06/09/2020 a 19/09/2020
Contextualização	Pesquisa sobre GQM.	20/09/2020 a 26/09/2020
Contextualização	Estudo sobre avaliação de processos com CMMI e MPS.BR.	27/09/2020 a 03/10/2020
Contextualização	Estudo sobre preenchimento da planilha de avaliação do MPS.BR, e elaboração do problema de pesquisa.	04/10/2020 a 17/10/2020
Contextualização	Escrita da introdução.	18/10/2020 a 31/10/2020
Contextualização	Início da escrita do referencial teórico.	01/11/2020 a 21/11/2020
Contextualização	Escrita da metodologia.	22/11/2020 a 05/12/2020
Contextualização	Finalização e revisão da escrita da metodologia de pesquisa e introdução.	01/03/2021 a 07/03/2021
Contextualização	Finalização e revisão do referencial teórico das seções: GQM, MPS.BR, Desenvolvimento ágil.	08/03/2021 a 14/03/2021
Contextualização	Finalização e revisão do referencial teórico das seções: Equipes de desenvolvimento de software.	15/03/2021 a 21/03/2021

Continua na próxima página

Tabela 15 – Cronograma executado (Continuação)

Fase	**Atividades realizadas**	**Período**
Contextualização	Início da escrita da seção: ISO 9126, SQwaRE 25000.	22/03/2021 a 28/03/2021
Coleta inicial de dados	Escrita dos capítulos sobre métricas e avaliação inicial, e Revisão da avaliação inicial do processo do Doarti.	29/03/2021 a 04/04/2021
Escrita	Escrita inicial do capítulo de considerações finais.	05/04/2021 a 11/04/2021
Escrita	Revisão da estrutura e outros aspectos do trabalho: título, informações, resumo, abstract, remoção de trechos inutilizados, figuras, tabelas e referências.	12/04/2021 a 25/04/2021
Escrita	Preparação para apresentação do TCC 1.	25/04/2021 a 25/05/2021
Elaboração da estratégia de testes	Pesquisa sobre referencial teórico relacionado a testes de software e estratégias de teste. Elaboração de uma estratégia de testes.	19/07/2021 a 25/07/2021
Elaboração da estratégia de testes	Escrita do referencial teórico relacionado a testes de software e estratégias de teste.	26/07/2021 a 15/08/2021
Validação da estratégia de testes	Pesquisa e escrita sobre qualidade de modelos de processos de software.	16/08/2021 a 19/09/2021
Validação da estratégia de testes	Elaboração e Realização das entrevistas com membros do Doarti.	20/09/2021 a 03/10/2021
Validação da estratégia de testes	Análise dos resultados da validação.	04/10/2021 a 10/10/2021
Escrita	Revisão do trabalho por completo.	11/10/2021 a 18/10/2021
Apresentação	Apresentação do trabalho.	19/10/2021 a 06/11/2021

APÊNDICE B – Métricas

B.1 Falhas válidas encontradas

Este conjunto de métricas foi selecionado para avaliar a testabilidade, considerando a qualidade externa do produto de software do Doarti. O foco dessas métricas é acompanhar a eficiência do processo de testes manuais do projeto, medindo, em conjunto, a porcentagem de falhas válidas encontradas por cada testador membro da equipe de testes do Doarti, bem como seus níveis de conhecimentos relacionados à estratégia de testes e à complexidade das implementações testadas. As especificações do conjunto de métricas se encontram na Tabela 16.

Na fase de elaboração de estratégia de testes, serão analisados os elementos mais importantes inerentes a uma estratégia de testes, para que sejam inclusos em um questionário que possa fornecer dados para a NCT (Nível de Conhecimento do Testador em relação à estratégia de testes aplicada no Doarti), descrita na Tabela 16.

B.2 Cobertura de testes

Este conjunto de métricas, detalhado na Tabela 17, foi selecionado para avaliar a testabilidade, considerando a qualidade interna do produto de software do Doarti. O foco dessas métricas é acompanhar a evolução da porcentagem de linhas de código relevantes, cobertas por testes automatizados, considerando o conhecimento técnico da equipe. Essa cobertura só considera a versão estável do código.

O questionário responsável por fornecer dados para a CETT (Conhecimento da Equipe relacionado a Testes e às Tecnologias utilizadas no projeto), descrita na Tabela 17, irá indagar os membros da equipe do Doarti em relação às experiências prévias em desenvolvimento de software e principais tecnologias utilizadas no desenvolvimento do aplicativo móvel, tais como: Firebase; Flutter; Dart; Testes automatizados entre outras.

Tabela 16 – Validade de Falhas Encontradas por Testador - VFET

Título	Validade de Falhas Encontradas por Testador - VFET
Objetivo, problema, objeto e perspectiva	Melhorar a eficiência do processo de testes manuais na perspectiva da equipe de desenvolvimento do Doarti.
Questões	Os testes manuais realizados ao final da Sprint detectam falhas válidas?
Métricas	Porcentagem de Falhas Aceitas - PFA. Falhas Validadas pela Equipe de Desenvolvimento - FVED. Total de Falhas Reportadas na Sprint por um Único Testador Específico - TFRS Coletada através da Fórmula: $PFA = \frac{FVED}{TFRS \cdot 100}$
	Nível de Conhecimento do Testador em relação à estratégia de testes aplicada no Doarti - NCT. Coletada através de questionários realizados no final de cada Sprint. Intervalo de 1 a 5, de menor para maior nível de conhecimento.
	Complexidade das Implementações Testadas - CIT. Coletada através de anotação já realizadas atualmente pela equipe nas especificações das demandas. Intervalo de 1 a 13, respeitando valores da sequência de Fibonacci, em que 1 representa menor complexidade e 13 maior complexidade.

Tabela 17 – Porcentagem de Código Testado - PCT

Título	Porcentagem de Código Testado - PCT
Objetivo, problema, objeto e perspectiva	Aumentar a cobertura de testes do código do aplicativo do Doarti.
Questões	Qual é a porcentagem de código testado atualmente?
Métricas	Porcentagem de Código Testado - PCT Coletada automaticamente através de ferramentas utilizadas no projeto.
	Conhecimento da Equipe relacionado a Testes e às Tecnologias utilizadas no projeto - CETT. Coletada através de questionários realizados no final de cada Sprint. Intervalo de 1 a 5, de menor para maior nível de conhecimento.

APÊNDICE C – Especificação de boas práticas para escrita de código testável

C.1 Instanciar objetos diretamente na classe ou em seu construtor

Exemplo:

```
class Doacao {
  Doador doador = Doador();
  Entidade entidade;

  Doacao() : entidade = Entidade();
}
```

Solução:

```
class Doacao {
  Doador doador;
  Entidade entidade;

  Doacao(this.doador, this.entidade);
}
```

A instanciação de objetos, seja no construtor ou diretamente na declaração dos atributos da classe, prejudica a testabilidade do código, uma vez que estes objetos não podem ser alterados, durante os testes, para uma versão forjada, com dados criados especificamente para teste.

Com a passagem dos objetos por parâmetro, no construtor da classe, torna-se possível a criação de objetos customizados para o teste da classe que os engloba.

Teste:

```
test('Teste de Doacao', () {
  Doador doador = DoadorTeste();
  Entidade entidade = EntidadeTeste();

  Doacao doacaoTeste = Doacao(doador, entidade);
```

```
  //Restante do teste
});
```

C.2 Exploração de objetos de valor através de objetos de serviço

Exemplo:

```
class CalculadoraDeTaxaDeEntrega {
  num calculaTaxa(Entidade entidade, num valorPorKm) {
    Endereco endereco = entidade.getEndereco();
    num distancia = calculaDistancia(endereco);
    return distancia * valorPorKm;
  }
}
```

Solução:

```
class CalculadoraDeTaxaDeEntrega {
  num calculaTaxa(Endereco endereco, num valorPorKm) {
    Endereco endereco = endereco;
    num distancia = calculaDistancia(endereco);
    return distancia * valorPorKm;
  }
}
```

Neste caso, é ressaltada a importância de se atentar a utilização dos parâmetros escolhidos para um método. No exemplo, percebe-se que o método ´´calculaTaxa" utiliza o objeto ´´entidade" apenas para acessar o endereço. Com isso, a criação do teste desse método requer a criação de objetos forjados para entidade e endereço.

A solução deste problema é passar, diretamente, o objeto de interesse, no parâmetro do método. Em casos como este, não há necessidade de se percorrer um caminho mais longo para acessar um atributo de um objeto.

Teste:

```
test('Teste CalculaTaxa', () {
  Endereco endereco = Endereco(/*Atributos de endereco com distancia de 2Km*/);
  //Entidade entidade = Entidade(endereco, /* Demais atributos de entidade*/);
  num valorPorKm = 2.00;

  expect(calculaTaxa(endereco, valorPorKm), 4.00);
});
```

É notável que, caso a solução sugerida não fosse aplicada, também seria necessário criar o objeto "entidade" para fornecê-lo como parâmetro para o método "calculaTaxa", sendo que apenas o valor do endereço é realmente utilizado, dificultando a criação do teste.

C.3 Instanciar objetos de dependências externas no corpo de métodos

Exemplo:

```
Future<Usuario> loginRedeSocial() async {
  RedeSocialXLogin redeSocialXLogin = RedeSocialXLogin(/*Atributos do objeto*/);
  //Implementação
  return usuario;
}
```

Solução:

```
Future<Usuario> loginRedeSocial(RedeSocialXLogin redeSocialXLogin) async {
  //Implementação
  return usuario;
}
```

A instanciação de algum objeto de terceiros, no corpo de um método, impossibilita o teste deste método, uma vez que torna-se inviável a criação de uma versão forjada deste objeto de terceiros, para fins apenas de testes.

Passando este objeto de terceiros por parâmetro para o método, sua criação fica a cargo do desenvolvedor, que ganha a possibilidade de criar uma versão forjada, específica para o momento dos testes.

Teste:

```
test('Teste login com rede social X', () async {
  RedeSocialXLogin redeSocialXLoginTeste = RedeSocialXLoginFalso();

  loginRedeSocial(redeSocialXLogin);

  //Restante do teste
});
```

APÊNDICE D – Considerações acerca da seleção dos elementos alvos de teste

É importante selecionar o que testar e o que não testar, pois a escrita de testes automatizados é custosa, e demanda mais tempo dos desenvolvedores (SOMMERVILLE, 2019). Por isso, foram abordadas neste documento algumas considerações com o objetivo de auxiliar os desenvolvedores na tomada de decisão, sobre criar ou não, testes automatizados para os elementos desenvolvidos pro eles.

Sommerville (2019) cita os chamados "testes de experiência" como um dos tipos de casos de testes unitários onde, como o nome já sugere, a experiência do desenvolvedor é crucial para a criação do teste. Neste tipo de seleção de caso de teste, o desenvolvedor procura cobrir com testes elementos que são fontes comuns de problemas.

Outra forma de se pensar na escolha dos elementos a serem cobertos por testes, consiste em atentar-se ao objetivo do teste. Segundo Naik e Tripathy (2011), um caso de teste deve ser associado a um propósito claro, que por consequência, determina seu modo de criação. Portanto, cabe ao desenvolvedor estabelecer, de forma clara, o propósito dos testes a serem feitos, com base no contexto das implementações pelas quais estão responsáveis. Com o objetivo de teste estabelecido, se torna mais fácil identificar quais elementos compõem o cenário que precisa ser testado.

Também é importante selecionar casos de testes, e consequentemente, seus elementos alvo, atentando-se aos requisitos do sistema. De acordo com Naik e Tripathy (2011), as especificações de funcionamento do sistema, e necessidades do usuário, são a principal fonte de casos de teste.

Em suma, a seleção de casos de teste, e seus elementos alvo, pode ser orientada à experiência do desenvolvedor/testador, ao propósito do teste e aos requisitos do sistema. Cada uma dessas técnicas, apresentadas como forma de auxílio ao desenvolvedor, pode ser utilizada isoladamente ou em complemento, para que a atividade #A1 da estratégia de testes proposta neste trabalho possa ser concluída.

APÊNDICE E – Questionários CETT e NCT

Os questionários aqui apresentados servem como método de coleta das métricas CETT e NCT, especificadas no Apêndice B. Todas as questões tem como resposta um número no intervalo de 1 a 5, onde 1 representa o menor nível de conhecimento, e 5 o maior nível de conhecimento.

E.1 Nível de Conhecimento do Testador em relação à estratégia de testes aplicada no Doarti (NCT)

- Qual grau de conhecimento você considera ter em relação à estratégia de testes aplicada no Doarti ?

E.2 Conhecimento da Equipe relacionado a Testes e às Tecnologias utilizadas no projeto (CETT)

- Qual grau de conhecimento você considera ter em relação à testes automatizados ?
- Qual grau de conhecimento você considera ter em relação à testes manuais ?
- Qual grau de conhecimento você considera ter em relação à lógica de programação ?
- Qual grau de conhecimento você considera ter em relação à Dart ?
- Qual grau de conhecimento você considera ter em relação à Flutter ?
- Qual grau de conhecimento você considera ter em relação à Firebase ?

APÊNDICE F – Modelo de caso de teste

O modelo de caso de teste apresentado aqui, possui campos de informações relevantes para a estratégia de testes desenvolvida para o Doarti, e foi adaptado do Manual de Produção de Software do Cercomp GPS (2010).

Tabela 18 – Modelo de caso de teste

Nome do caso de teste	
Identificador	
Descrição	
Componentes testados	
Pré-condições	
Pós-condições	
Passos previstos	
Resultados esperados	
Executor	
Passos executados	
Resultados observados	

Anexos

ANEXO A – Fluxos de utilização do Doarti

Neste anexo são apresentados os fluxos de intenção de participação e de doação do Doarti, de acordo com a documentação interna do projeto.

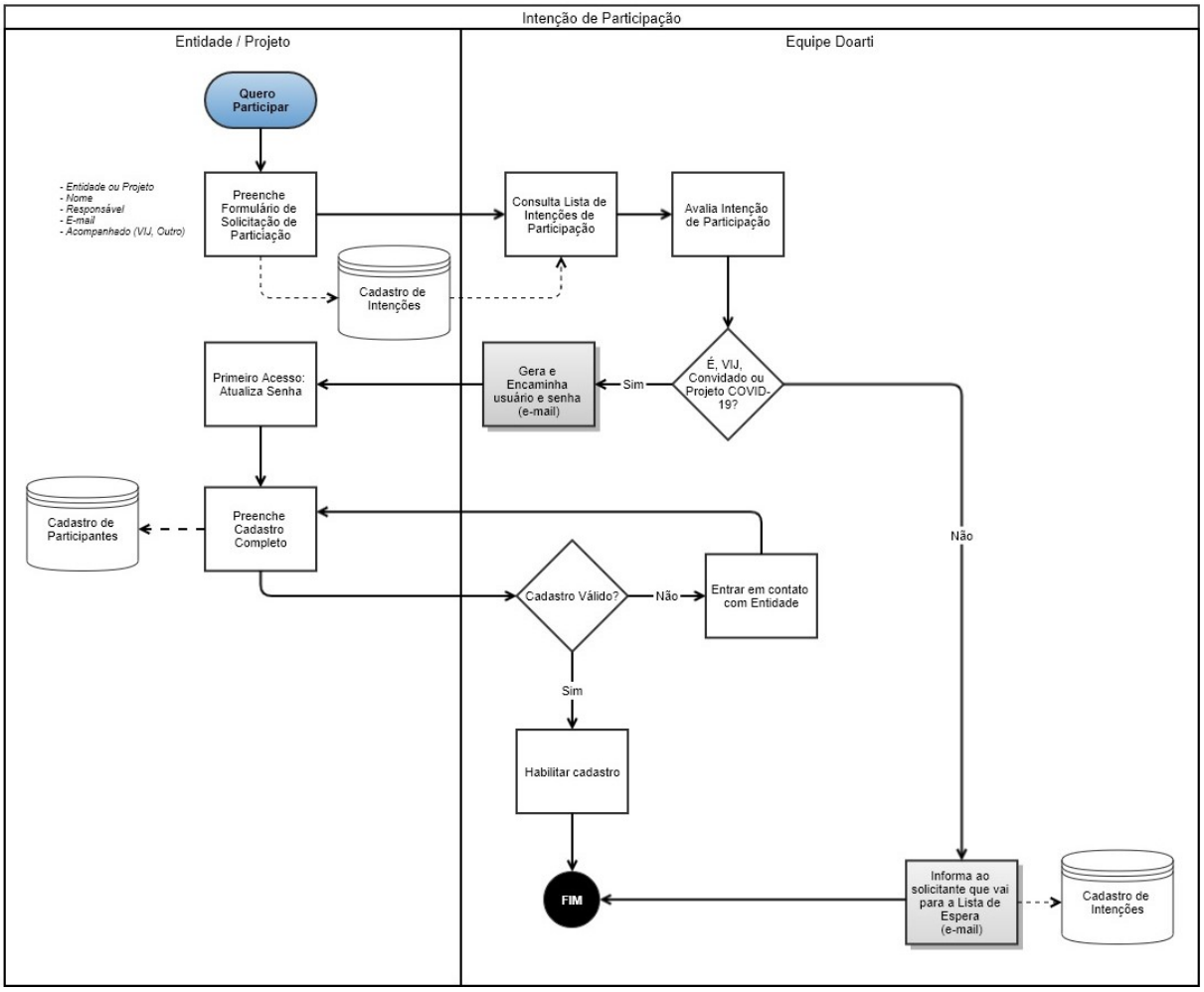

Figura 10 – Fluxo de intenção de participação. Fonte: Documentação interna do Doarti.

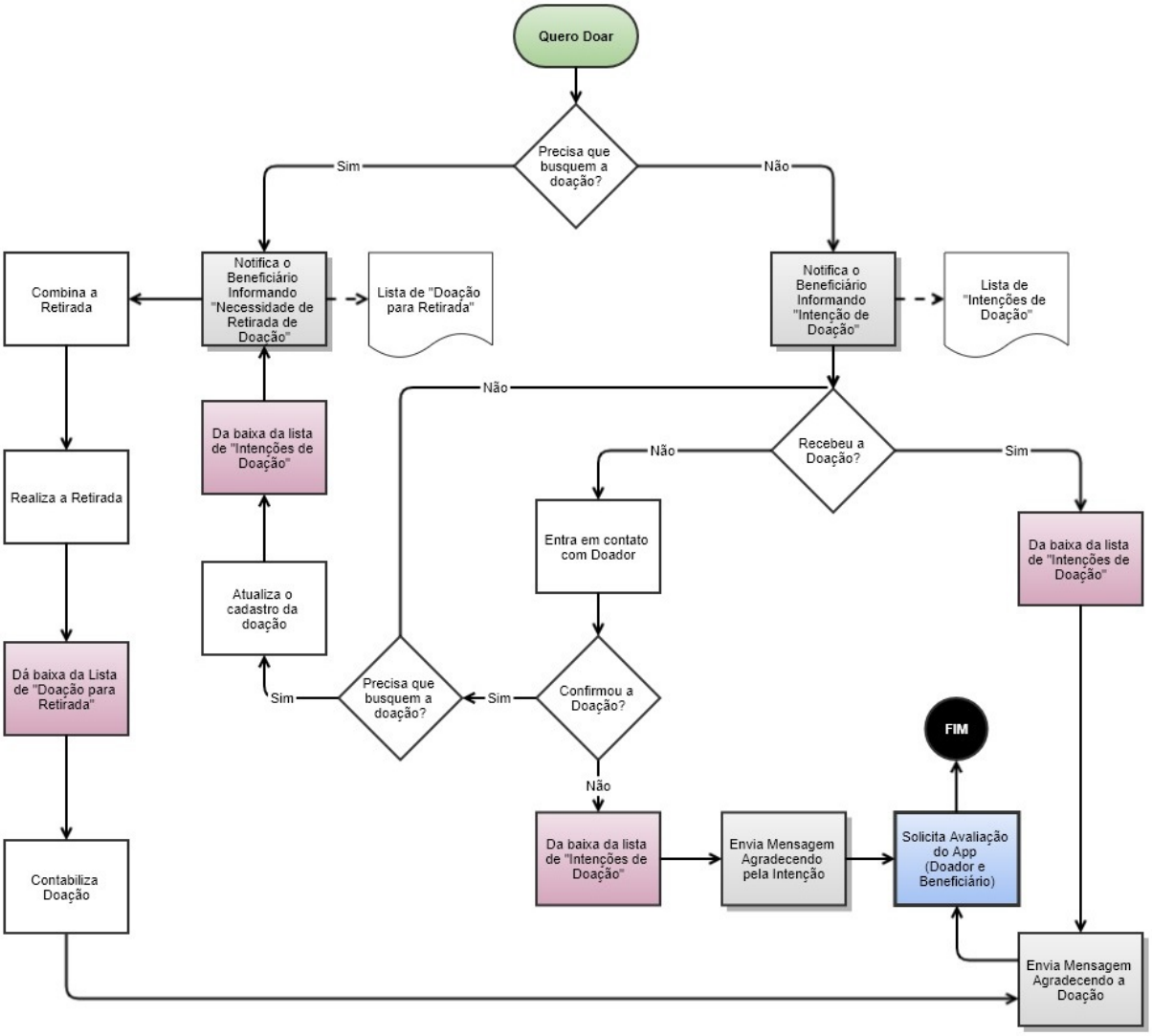

Figura 11 – Fluxo de doações. Fonte: Documentação interna do Doarti.