Instituto Federal de Educação, Ciência e Tecnologia da Paraíba

Campus João Pessoa

Programa de Pós-Graduação em Tecnologia da Informação

Nível Mestrado Profissional

RAFAEL ANDERSON DE LIMA RAMOS

MONITORAMENTO DE MÉTRICAS DE QUALIDADE E PRODUTIVIDADE EM PROJETOS ÁGEIS DE SOFTWARE ATRAVÉS DA INTEGRAÇÃO DE DADOS EXTRAÍDOS DE FERRAMENTAS DE GESTÃO E TESTES

DISSERTAÇÃO DE MESTRADO

JOÃO PESSOA - PB

2022

Rafael Anderson de Lima Ramos

MONITORAMENTO DE MÉTRICAS DE QUALIDADE E PRODUTIVIDADE EM PROJETOS ÁGEIS DE SOFTWARE ATRAVÉS DA INTEGRAÇÃO DE DADOS EXTRAÍDOS DE FERRAMENTAS DE GESTÃO E TESTES

Dissertação de Mestrado apresentada como requisito parcial para obtenção do título de Mestre em Tecnologia da Informação pelo Programa de Pós-Graduação em Tecnologia da Informação do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba – IFPB.

Orientadora: Prof. Dra. Juliana Dantas Ribeiro Viana de Medeiros

Coorientadora: Prof. Dra. Heremita Brasileiro Lira

João Pessoa - PB

2022

Dados Internacionais de Catalogação na Publicação – CIP
Biblioteca Nilo Peçanha –IFPB, *Campus* João Pessoa

R175m Ramos, Rafael Anderson de Lima.
Monitoramento de métricas de qualidade e produtividade em projetos ágeis de software através da integração de dados extraídos de ferramentas de gestão e testes / Rafael Anderson de Lima Ramos. – 2022.
127 f. : il.

Dissertação (Mestrado em Tecnologia da Informação) – Instituto Federal da Paraíba – IFPB / Programa de Pós-Graduação em Tecnologia da Informação - PPGTI.
Orientadora: Profª. Dra. Juliana Dantas Ribeiro Viana de Medeiros.
Coorientador: Profª. Dra. Heremita Brasileiro Lira.

1. Qualidade de software. 2. Desenvolvimento ágil. 3. Métricas de produtividade. 4. Coleta automática de dados. 5. Open Project. 6. TestLink. I. Título.

CDU 004.414.2

Bibliotecário responsável Marx da Silva Medeiros – CRB15/470

MINISTÉRIO DA EDUCAÇÃO

SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

PROGRAMA DE PÓS-GRADUAÇÃO *STRICTO SENSU*

MESTRADO PROFISSIONAL EM TECNOLOGIA DA INFORMAÇÃO

RAFAEL ANDERSON DE LIMA RAMOS

MONITORAMENTO DE MÉTRICAS DE QUALIDADE E PRODUTIVIDADE EM PROJETOS ÁGEIS DE SOFTWARE ATRAVÉS DA INTEGRAÇÃO DE DADOS EXTRAÍDOS DE FERRAMENTAS DE GESTÃO E TESTES

Dissertação apresentada como requisito para obtenção do título de Mestre em Tecnologia da Informação, pelo Programa de Pós- Graduação em Tecnologia da Informação do InstitutoFederal de Educação, Ciência e Tecnologia da Paraíba – IFPB - Campus João Pessoa.

Aprovado em 23 de Dezembro de 2022

Membros da Banca Examinadora:

Dra. Juliana Dantas Ribeiro Viana de Medeiros

IFPB - PPGTI

Dra. Heremita Brasileiro Lira

IFPB - PPGTI

Dr. Francisco Petrônio Alencar de Medeiros

IFPB - PPGTI

Dr. Gustavo Henrique Matos Bezerra Motta

UFPB – MEMBRO EXTERNO

João Pessoa/2022

Documento assinado eletronicamente por:

- Juliana Dantas Ribeiro Viana de Medeiros, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 23/01/2023 10:57:27.
- Heremita Brasileiro Lira, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 23/01/2023 11:39:13.
- Francisco Petronio Alencar de Medeiros, PROFESSOR ENS BASICO TECN TECNOLOGICO, em 23/01/2023 14:54:03.
- Gustavo Henrique Matos Bezerra Motta, PROFESSOR DE ENSINO SUPERIOR NA ÁREA DE ORIENTAÇÃO EDUCACIONAL, em 23/01/2023 16:07:58.

Este documento foi emitido pelo SUAP em 05/12/2022. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código Verificador: 363371
Código de Autenticação: 53c6e727de

AGRADECIMENTOS

Gostaria de agradecer primeiramente a Deus, pelo dom da vida e pela saúde concedida, o que me proporcionou enfrentar os desafios encontrados ao longo da estrada da vida.

Aos meus pais, Joselma e Manoel, que doaram os seus esforços em pró do meu sucesso pessoal e profissional.

Aos meus avós maternos, Júlio e Ivonete, pois foram eles que moldaram o meu caráter, ensinando-me o caminho certo ao qual deveria trilhar para ser um homem íntegro e honesto.

Ao meu tio materno, Josenildo Francisco, que me acolheu em sua casa durante os anos de graduação, tendo para com a minha pessoa um verdadeiro cuidado paternal.

A minha esposa, Lidiane de Azevedo de Lima, pelo apoio concedido e companheirismo demonstrado.

A todos os meus familiares, que tanto apoio me concederam através das mais variadas demonstrações de carinho e afeto, fazendo com que a jornada acadêmica se tornasse menos árdua.

Agradeço também a minha orientadora e coorientadora, Juliana Dantas e Heremita Brasileiro respectivamente, pelo companheirismo demonstrado ao longo da execução deste projeto de pesquisa.

Aos colegas de estudo Joerverson Santos, Helder Rangel, Helena Miguel e Tales Bonfim, por sempre estarem ao meu lado auxiliando no desenvolvimento deste projeto através do impulsionamento de um grupo de pesquisa ativo e dedicado, deixando-me muito feliz em poder desfrutar da companhia de pessoas como tais.

Para finalizar, agradeço a todos que contribuíram de forma direta ou indireta para que este trabalho viesse a se concretizar.

RESUMO

O desenvolvimento de um *software* de qualidade é uma atividade que não requer apenas a codificação propriamente dita. Na prática, abrange um conjunto de processos, estando entre eles o levantamento de requisitos, análise, projeto, configuração, além de uma série de valores que atendam os objetivos de negócio dos clientes que utilizam um determinado sistema. Nesse contexto, muitas empresas buscam obter competitividade associando produtividade com qualidade, para garantir que os produtos de *software* sejam entregues com os requisitos contratados. Na gestão de projetos, a percepção do *status* de um projeto requer o acesso a informações sobre o andamento das atividades à medida que elas estão sendo realizadas pela equipe de desenvolvimento. Em um processo de tomada de decisão, em muitas ocasiões o gerente de projeto precisa coletar dados provenientes de fontes diferentes, como por exemplo, ferramentas de gestão de tarefas, registro de *bugs*, realização de testes e controle de versão. Isso geralmente demanda um tempo considerável por parte do gerente. Sendo assim, o objetivo desta pesquisa é investigar sobre métricas de qualidade e produtividade em projetos ágeis de *software* e propor mecanismos para monitorar essas métricas a partir da integração de dados extraídos de ferramentas de gestão e testes de *software*. Para isso, inicialmente foi realizada uma Revisão Sistemática da Literatura (RSL) visando identificar as principais estratégias de extração e integração dos dados contidos em distintas ferramentas de apoio utilizadas em projetos ágeis de *software*. A partir da análise de alguns projetos ágeis executados no Laboratório Assert do IFPB, foi realizado um mapeamento das métricas de qualidade e produtividade mais apropriadas para monitoramento de projetos no referido contexto. Em seguida, foi definida e implementada uma arquitetura para uma plataforma que se propõe a integrar e visualizar métricas de qualidade e produtividade extraídas das ferramentas *Open Project* (sistema para gerenciamento de projetos) e *Testlink* (sistema para gerenciamento de testes de *software*). Por fim, a primeira versão da plataforma desenvolvida foi validada em um dos projetos no Laboratório Assert. Resultados preliminares demonstraram que o conjunto de métricas coletadas e disponibilizadas através dos gráficos presentes no *dashboard* proporcionaram uma melhor gestão do progresso de um projeto. Identificou-se também que o nível de detalhes das métricas coletadas podem sofrer impactos mediante a forma como o *backlog* é detalhado na ferramenta *Open Project* e como os testes de um projeto são descriminados e referenciados na ferramenta *TestLink.*

Palavras-chaves: Métricas de Produtividade e Qualidade de Software; Desenvolvimento Ágil; Extração e Visualização de Métricas; Integração de Plataformas; Coleta Automática de Dados; Open Project; TestLink.

ABSTRACT

Developing quality software is an activity that requires more than just the coding itself. In practice, it encompasses a set of processes, among them are requirements gathering, analysis, design, configuration, and a series of values that meet the business objectives of the customers that use a given system. In this context, many companies seek to achieve competitiveness by associating productivity with quality, to ensure that software products are delivered with the contracted requirements. In project management, the perception of a project's status requires access to information about the progress of activities as they are being carried out by the development team. In a decision-making process, the project manager often needs to collect data from different sources, such as task management tools, bug tracking, testing, and version control. This often demands a considerable amount of time from the manager. Thus, the objective of this research is to investigate quality and productivity metrics in agile software projects and to propose mechanisms to monitor these metrics by integrating data extracted from software management and testing tools. To this end, initially a Systematic Literature Review (SLR) was performed aiming to identify the main data extraction and integration strategies contained in different support tools used in agile software projects. Based on the analysis of some agile projects executed in the Assert Lab of IFPB, a mapping of the most appropriate quality and productivity metrics for monitoring projects in this context was performed. Next, an architecture was defined and implemented for a platform that proposes to integrate and visualize quality and productivity metrics extracted from Open Project (project management system) and Testlink (software test management system). Finally, the first version of the developed platform was validated in one of the projects in the Assert Lab. Preliminary results showed that the set of metrics collected and made available through the graphics present in the dashboard provided a better management of a project's progress. It was also identified that the level of detail of the collected metrics can suffer impacts by the way the backlog is detailed in the Open Project tool and how the tests of a project are broken down and referenced in the TestLink tool.

Keywords: Productivity Metrics and Software Quality; Agile Development; Metrics Extraction and Visualization; Platform Integration; Automatic Data Collection; Open Project; TestLink.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

RSL	Revisão Sistemática da Literatura
TI	Tecnologia da Informação
ES	Engenharia de *Software*
ASD	*Agile Software Development*
XP	*Extreme Programming*
QA	*Quality Assurance*
IFPB	Instituto Federal da Paraíba
PD&I	Pesquisa, Desenvolvimento e Inovação
QP	Questão de Pesquisa
QPE	Questão de Pesquisa Específica
IC	*Inclusion Criteria*
EC	*Exclusion Criteria*
SMeS	Seleção de métricas de *software*
GQM	*Goal-Question-Metric*
AMP	*Assert Metrics Panel*
API	*Application Programming Interface*
ID	*Identity*
SQL	*Structured Query Language*
JSON	*JavaScript Object Notation*
CSS	*Cascading Style Sheets*
SPA	Aplicativos de Uma Só Página
MVC	Modelo, Visão e Controle
WBS	*Work Breakdown Structure*
PERT	*Program Evaluation and Review Technique*
VPN	*Virtual Private Network*
HTML	*HyperText Markup Language*

CAPES Coordenação de Aperfeiçoamento de Pessoal de Nível Superior

LISTA DE SÍMBOLOS

%	Por cento
#	Cerquilha

SUMÁRIO

1. INTRODUÇÃO

Métricas de *software* são parâmetros para a medição de um *software*. As métricas de *software* possibilitam a obtenção de uma variada gama de informações sobre a qualidade do produto, progresso de um projeto, custo e tamanho/complexidade de um determinado sistema. Segundo Cho et al. (2016), as métricas podem auxiliar efetivamente aos times de desenvolvimento ágil de *software* a lidarem com um dos grandes problemas que surgem no processo de desenvolvimento, que são as exigências corriqueiras por mudanças no escopo do produto. Pois, através das medições constantes e acompanhadas com periodicidade, é possível obter indicadores objetivos que representam a confiabilidade na capacidade do sistema atender às exigências atribuídas.

Devido ao processo iterativo e incremental, seguido em desenvolvimentos ágeis de *software*, muitas das formas de medições de sistemas utilizadas no desenvolvimento tradicional não são adequadas para o universo ágil, visto que não lidam com a realidade de constante mudança. Muitas métricas de *software* bem estabelecidas encontram-se disponíveis para o processo tradicional de desenvolvimento, mas apenas um pequeno nicho de métricas podem ser aplicados diretamente no desenvolvimento ágil e em alguns casos necessitando de adaptações, para que de fato essas métricas possam ser usadas para melhorar a próxima iteração (PADMINI et al., 2015).

Na gestão de projetos, a percepção do *status* requer acesso objetivo e imediato às informações sobre o andamento das atividades, à medida que elas estão sendo realizadas pela equipe de desenvolvimento. Também se faz necessário deter informações sobre a qualidade do produto, como por exemplo a quantidade e os tipos de *bugs* reportados pelo time de QA (*Quality Assurance*). Respaldando assim, a importância de uma integração das métricas contidas em diversos setores atuantes no ciclo de desenvolvimento de um *software* e visando explicitamente ajudar no acompanhamento do progresso do projeto, monitorando a qualidade do produto e permitindo uma melhor previsão e gestão de projetos.

Apesar da importância em deter tais dados citados, em diversas ocasiões os dados gerados em um projeto de *software* encontram-se disponíveis em distintas fontes, pois é demasiadamente corriqueiro o uso de variadas ferramentas de gestão de atividades e de qualidade no mesmo escopo de um projeto, sendo desta forma frequente os gestores demorarem ou até mesmo não conseguirem obter uma informação precisa por não terem a habilidade específica ou o tempo necessário para consolidar tal massa de informações. Ainda segundo Padmini et al. (2015), existe uma dificuldade em identificar quais métricas de *software* são aplicáveis no processo de desenvolvimento ágil e como deve-se realizar a correta adaptação de tais métricas ao contexto de atuação.

Mediante ao contexto apresentado é importante frisar que a gestão ágil de projetos se difere de uma gestão tradicional. Seu objetivo principal, busca a economia de tempo na realização de atividades, priorizando uma comunicação e atuação integrada (BUILDER, 2019).

Por muitas vezes o Gerente de Projetos necessita deter habilidades e dedicar um bom tempo do seu trabalho para que as informações geradas em um projeto sejam consolidadas e transformadas em dados relevantes, acessíveis e valiosos. A posse de tais dados possibilita a um gestor tomar decisões sempre baseado em informações consolidadas e atualizadas. Porém, a contradição desta conjuntura encontra-se justamente em se colocar diversas barreiras entre as informações relevantes e as pessoas que necessitam acessá-la, não ofertando uma análise e síntese automática das métricas necessárias e deixando de gerar predições oportunas e de valor para o fidedigno acompanhamento do projeto ágil de desenvolvimento (SHIU, 2020).

1.1. Justificativa e Definição do Problema

A competitividade global é considerada a chave no mercado entre os setores que buscam ofertar produtos de qualidade para manter-se no mercado. A busca pela atualização empresarial deve ser uma constante e a eficiência tanto nos produtos quanto nos serviços oferecidos deve ser elevada.

Em meio a esta busca constante pela satisfação do cliente e pela qualidade dos produtos, as empresas começam a rever e a reestruturar seus conceitos, tentando assim, detectar e até mesmo antecipar os possíveis problemas, na busca por soluções imediatas ou de longo prazo, porém sempre focadas em seus produtos ou em suas prestações de serviços (SANTOS JÚNIOR, 2015).

Métodos ágeis de desenvolvimento de *software* são hoje em dia amplamente difundidos e aceitos. De acordo com o VersionOne (2020), empresas que praticam o desenvolvimento ágil em mais da metade de suas equipes passaram de 26% no ano de 2019, para 33% no ano de 2020. Exemplificando assim, a ampla e rápida aceitação dos métodos ágeis pelo mercado e evidenciando uma notável necessidade pela identificação de conjuntos de métricas que mais se adequem ao processo ágil.

De acordo com Kunz et al. (2008) e Padmini et al. (2015), a familiaridade que muitos desenvolvedores e gerentes de projetos possuem para com os princípios do desenvolvimento tradicionais de *software*, acaba por tendenciar tais profissionais a utilizarem as mesmas métricas do contexto tradicional também no contexto ágil. Esse tipo de tentativa de medição, pode levar a uma interpretação equivocada do progresso do projeto, com impacto direto na qualidade dos resultados. Este cenário, de uso equivocado de métricas, também tende a gerar um impacto negativo entre os *stakeholders* envolvidos no empreendimento, acarretando frustrações em ambas as partes.

É importante frisar que, apesar de existirem muitas pesquisas que avaliam os diversos ângulos do desenvolvimento ágil de *software*, relativamente menos atenção é concedida à aplicação de métricas de *software* em um ambiente ágil de projetos. Como exemplo temos a pesquisa VersionOne (2020), concentrada no processo ágil de desenvolvimento de *software,* mas que não se refere em momento algum a quais métricas de *software* são aplicáveis ao desenvolvimento ágil e quais adequações fazem-se necessárias para a realização de uma objetiva e correta medição tendo por foco a extração e a análise dos dados de um projeto.

Além da problemática exposta no tocante da seleção de um conjunto ideal de métricas a serem aplicadas no contexto ágil, também é possível notar na literatura que praticamente não se abordam estratégias para coleta de dados junto ao ferramental comumente utilizado no ciclo de desenvolvimento de *software.* Segundo Padmini et al. (2015) não existe um senso comum na utilização pelas empresas atuantes no mercado de instrumentais especializados para acompanhamento de métricas em projetos, levando assim a empregabilidade de uma variedade de estratégias para coleta de métricas e impactando diretamente na qualidade da medição realizada, visto que tais medições podem ocorrer em um contexto que envolvam desde a coleta manual de dados até análise das informações coletadas em planilhas.

Esta problemática também foi observada na gestão dos projetos executados no Laboratório Assert, que desde a sua criação em 2016, tem se apresentado como a principal ferramenta de atuação do Polo EMBRAPII (Empresa Brasileira de Pesquisa e Inovação Industrial)[1] do IFPB (Instituto Federal da Paraíba) para execução de projetos de PD&I (Pesquisa, Desenvolvimento e Inovação) com empresas. O laboratório já executou mais de 40 projetos de PDI com mais de 20 empresas, envolvendo mais de 60 colaboradores internos e externos. Entretanto, o monitoramento do progresso de execução dos projetos de *software* requer um esforço considerável dos gestores que precisam coletar dados manualmente de duas ferramentas diferentes (*Open Project*[2] e *TestLink*[3]) e depois integrar os dados através de planilhas de *excel* para gerar métricas de qualidade e produtividade dos projetos. A ferramenta *Open Project* é utilizada para gerenciamento de projetos e a ferramenta *Testlink* para gerenciamento de testes de *software*.

Ao optar pela utilização dessas ferramentas, os gestores acabam encontrado uma barreira tecnológica e conceitual que os possibilitem unificar as informações em uma única visualização acessível de métricas desejadas. Posto isto, o acompanhamento em tempo real através de uma plataforma que extraia dados de diferentes ferramentas e gere métricas sobre a qualidade e a produtividade, permitindo a visualização atualizada do progresso dos projetos, torna-se justificado e necessário.

[1] EMBRAPII - https://embrapii.org.br/

[2] Open Project – https://www.openproject.org/

[3] Testlink – https://testlink.org/

1.2. Objetivos e Questão de Pesquisa

O objetivo geral desta pesquisa é propor uma plataforma para monitorar métricas de qualidade e produtividade de *software* a partir da integração de dados extraídos de ferramentas de gestão de projetos e testes.

Desta forma, os objetivos específicos apresentados para este trabalho são:

- Realizar revisão sistemática da literatura sobre as estratégias utilizadas para extração e disponibilização de métricas de qualidade e produtividade de projetos ágeis de *software*;
- Definir um conjunto de métricas de qualidade e produtividade a serem coletadas em projetos ágeis de *software*;
- Propor arquitetura para uma plataforma que se propõe a integrar e visualizar métricas de qualidade e produtividade através da extração de dados das ferramentas *Open Project* e *TestLink*;
- Implementar a versão inicial da plataforma;
- Validar a plataforma em um dos projetos executados no Laboratório Assert do IFPB.

1.3. Metodologia

Considerando o contexto e os objetivos definidos, as seguintes questões de pesquisa foram estabelecidas:

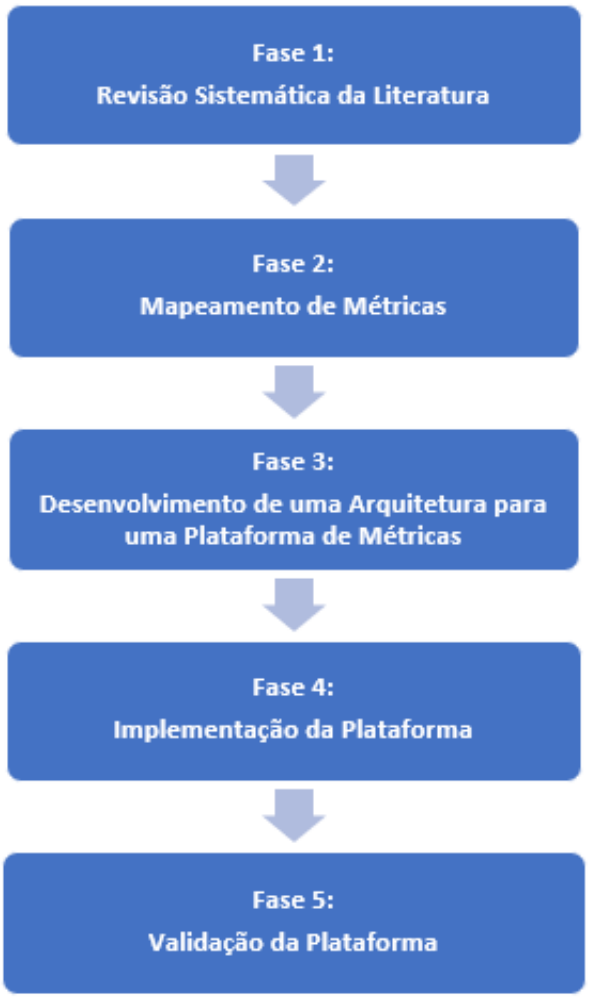
- QP1: Quais são as estratégias utilizadas para extração e disponibilização de métricas de qualidade e produtividade em equipes que desenvolvem projetos ágeis de *software*?
- QP2: Como extrair e integrar dados do *Open Project* e o *TestLink* para disponibilizar métricas de qualidade e produtividade de projetos ágeis de *software*?

O objetivo geral desta pesquisa é buscar respostas para perguntas que são, em essência, exploratórias. As questões exploratórias são projetadas para obter um conhecimento mais profundo sobre algum fenômeno e discutir questões úteis que ajudam a esclarecer a compreensão desse fenômeno (EASTERBROOK et al., 2008). O fenômeno em questão é o monitoramento de métricas de *software* relacionadas a qualidade e produtividade em projetos ágeis de *software*.

Uma estratégia mista de pesquisa foi utilizada para investigar o fenômeno na literatura e na prática industrial, objetivando construir uma descrição rica sobre o mesmo. Inicialmente foi realizada uma Revisão Sistemática da Literatura (RSL) visando identificar as principais estratégias de extração e integração dos dados contidos em distintas ferramentas de apoio utilizadas em projetos ágeis de *software*. A partir da análise de alguns projetos ágeis executados no Laboratório Assert do

IFPB, foi realizado um mapeamento das métricas de qualidade e produtividade mais apropriadas para monitoramento de projetos no referido contexto. Em seguida, foi definida e implementada uma arquitetura para uma plataforma que se propõe a integrar e visualizar métricas de qualidade e produtividade extraídas das ferramentas *Open Project* e *Testlink*. Por fim, a primeira versão da plataforma desenvolvida foi validada em um dos projetos no Laboratório Assert. A Figura 1 ilustra as fases adotadas para execução da pesquisa:

Figura 1 – Etapas da metodologia realizadas neste trabalho de pesquisa.



Fonte: Produzido pelo autor.

- Fase 1: Revisão Sistemática da Literatura. O propósito da fase consistiu na realização de uma Revisão Sistemática da Literatura (RSL) cujo foco esteve centralizado na busca pelo domínio acerca das estratégias utilizadas para extração e disponibilização de métricas de qualidade e produtividade em projetos ágeis de *software*, tido como embasamento teórico fundamental para o transcorrer desta pesquisa.

- Fase 2: Mapeamento de Métricas. Esta fase teve por objetivo realizar o mapeamento das métricas de qualidade e produtividade mais apropriadas para o monitoramento de projetos ágeis de *software* que utilizam as ferramentas *Open Project* e *Testlink*. O mapeamento foi realizado a partir da análise de projetos ágeis executados no Laboratório Assert que utilizavam as referidas ferramentas.
- Fase 3: Desenvolvimento de uma Arquitetura para uma Plataforma de Métricas. A partir dos resultados das fases anteriores, implementou-se uma arquitetura base para uma plataforma que se propõe a integrar e visualizar métricas de qualidade e produtividade através da extração de dados das ferramentas *Open Project* e *TestLink*. Tal arquitetura foi subdivida em dois importantes módulos: O Módulo Extrator, responsável pela coleta dos dados e o Módulo *Dashboard*, responsável pelo tratamento dos dados e por sua respectiva visualização.
- Fase 4: Implementação da Plataforma. Nesta fase foi implementada a versão inicial da plataforma que se constitui de um dashboard com gráficos que exibem métricas ágeis de produtividade e qualidade extraídas das ferramentas *Open Project* e o *TestLink*.
- Fase 5: Validação da Plataforma. Na quinta e última fase, foi realizada a validação da plataforma em um projeto real executado no Laboratório Assert. Para isso, foi feita a configuração de extração dos dados para consumir de informações exclusivas de um dado projeto do Laboratório Assert, possibilitando aos gerentes de projeto utilizar e validar tal solução.

1.4. Aplicabilidade

A solução proposta neste trabalho busca realizar a extração e facilitar a visualização de métricas de produtividade e qualidade em ambientes ágeis de desenvolvimento de *software,* tendo por direcionamento integrar-se a diferentes fontes de dados utilizadas no ciclo de desenvolvimento para que exista uma padronização das informações coletadas e uma disseminação ainda maior dos dados cruciais para a gestão de um projeto.

Neste sentido, optou-se por realizar a aplicação no contexto do Polo de Inovação – IFPB da plataforma proposta nesta pesquisa, pois tal ecossistema apresenta alguns critérios de alta relevância para o trabalho como:

- Espaço dedicado a atender demandas das cadeias produtivas no tocante a projetos de PD&I;
- Alto volume de dados referentes a produtividade e qualidade, provenientes de projetos de desenvolvimento ágeis de *software* ativos ou finalizados;

- Notória dificuldade na geração automática de métricas de produtividade e qualidade, devido ao fato de se utilizar diferentes ferramentas de gestão que não possuem relacionamentos;
- Processos sólidos e com um conjunto de métricas, coletadas manualmente, pré-estabelecidas;
- Anseio dos líderes e gestores por uma plataforma de controle automático de métricas.

Tal escolha pelo uso do Polo de Inovação – IFPB como ambiente propício para esta pesquisa, deve-se primordialmente ao fato do mesmo ser um espaço dedicado a atender demandas das cadeias produtivas no tocante a projetos de PD&I, sendo considerada uma unidade de importância ímpar para a instituição no que diz respeito a prestação de serviços tecnológicos. Neste sentido, o Polo de Inovação - IFPB mostra-se como um ecossistema relevante para aplicação da pesquisa sugerida, visto que um de seus objetivos é agregar aos seus projetos nova iniciativas de inovação e soluções disruptivas, advindas preferencialmente, de pesquisas propostas pela comunidade acadêmica do IFPB.

Também é importante ressaltar que, o Polo é uma unidade do IFPB constituída por uma total de sete laboratórios descentralizados e que atuam em propósitos específicos, sendo o Assert um destes laboratórios. Desta forma, neste contexto de inovação apresentado, o Laboratório Assert destaca-se como a principal ferramenta de atuação do Polo EMBRAPII do IFPB-JP para execução de projetos e também se destaca como centro operacional do Polo de Inovação – IFPB.

Tamanha experiência e comprometimento na execução de projetos juntamente a empresas parceiras, possibilitaram o Laboratório Assert aprimorar os seus processos de desenvolvimento ao mesmo tempo que se mantém aberto para experimentação de novos processos ágeis que lhe permitam um maior desempenho e qualidade na entrega das demandas.

É importante frisar que, a plataforma obtida como resultado desta pesquisa é muito mais que um simples *dashboard* expositor de dados, informações ou métricas. O *software* aqui apresentado possibilita que ecossistemas como o Polo de Inovação – IFPB, tenham uma solução de unificar automaticamente as informações de um mesmo projeto. Informações estas, que em muitas oportunidades encontram-se destrinchadas em diferentes ferramentas de gestão, permitindo assim que *softwares* responsáveis pelo monitoramento exclusivo da qualidade de um projeto passem a interagir com os *softwares* exclusivos para gestão de *backlogs*, dando ainda mais liberdade para os centros de pesquisa escolherem suas ferramentas de gestão dos projetos ágeis, visto que o monitoramento e controle é garantido através de uma plataforma de gestão de métricas completamente integrada.

Em suma, o emprego da proposta aqui exposta busca a otimização da produção e a diminuição dos custos. Pois, a visualização de *dashboards* integrados e consolidados promovem a melhor interpretação das métricas apresentadas, levando os gestores a entenderem por exemplo, se

índices de qualidade alcançados foram de fato obtidos dentro de um custo satisfatório de tempo e esforço.

1.5. Estrutura do Documento

O restante deste documento está organizado da seguinte forma:

- Capítulo 2 aborda a fundamentação teórica do projeto de pesquisa, de um ponto de vista mais conceitual. Traz um enfoque nas descrições, conceitos, recomendações, entre outros aspectos técnicos descritos. Também apresenta as tecnologias que foram utilizadas para desenvolvimento da Plataforma de Métricas.
- Capítulo 3 faz uma exposição detalhada do protocolo utilizado para realização da revisão sistemática com foco na busca por trabalhos de relevância para esta pesquisa. Buscando assim, descrever e categorizar os trabalhos selecionados, relacionando-os com as questões de pesquisa elencadas.
- Capítulo 4 descreve como foi realizado o mapeamento das métricas de qualidade e produtividade para monitoramento de projetos ágeis de *software*. E apresenta uma descrição detalhada da plataforma desenvolvida para monitorar as referidas métricas a partir da integração de dados extraídos das ferramentas *Open Project* e *TestLink*. A arquitetura da plataforma é detalhada, assim como as estratégias empregadas para a implementação, buscando explicitar todos os processamentos necessários desde a coleta dos dados até a criação dos *dashboards* personalizados.
- Capítulo 5 apresenta os resultados da validação da plataforma em um dos projetos executados no Laboratório Assert. Sendo detalhado o ambiente de experimentação no qual a plataforma foi implantada e validada.
- Capítulo 6 traz a apresentação das considerações finais do trabalho.

2. FUNDAMENTAÇÃO TEÓRICA

Pode-se afirmar que a Tecnologia da Informação (TI) se mostra cada vez mais como uma área de grande importância para inúmeros setores da sociedade, impactando diretamente nas atividades acadêmicas, comerciais e servindo de molde para a construção de um novo escopo social. Já na década de 90, Chu Shao Yong (1992) considerava a TI como um dos principais fatores para alcançar-se o sucesso em meio ao mundo dos negócios, seja a nível de sobrevivência (incluindo alongamento do ciclo de vida dos produtos e revitalização das organizações), como na obtenção de maior competitividade

A tecnologia trouxe a possibilidade de automatizar e solucionar problemas do cotidiano, levando a um ganho exponencial de produtividade, praticidade, segurança, entre outras vantagens. Um exemplo prático destas vantagens citadas anteriormente pode ser descrito com o uso do *software*, uma ferramenta fundamental para atender as necessidades das pessoas (RAMOS, 2015). Porém, é importante salientar neste contexto, que quando se investe bastante em TI e o ganho em produtividade obtido não é efetivamente expressivo, então se observa uma contradição. A essa contradição, o economista norte-americano Stephen S. Roach deu o nome de Paradoxo da Produtividade (ROACH, 1988).

O processo de desenvolvimento de *software* é complexo e necessita de grande prudência. Desta forma, a Engenharia de *Software* propõe soluções de suporte ao desenvolvimento baseados em paradigmas, que segundo Pressman (2016), enfatizam a tecnologia de camadas visando uma melhor qualidade organizacional.

Segundo Sommerville (2019), a Engenharia de *Software* adota uma abordagem sistemática buscando solucionar problemas em um âmbito que engloba as atividades de desenvolvimento e gerenciamento do ciclo de vida do *software*. Desta maneira, a Engenharia de *Software* gera artefatos que buscam um alto nível de qualidade, segundo cita Pressman (2016). Qualidade esta que pode ser alcançada através da definição de um processo contendo atividades específicas, que incluem desde revisões técnicas até a realização de testes multicamadas (PRESSMAN, 2016).

Ao longo do tempo, o desenvolvimento de soluções de *software* vêm sendo aprimoradas, e a gestão destes produtos também vêm acompanhando tais aprimoramentos. Sendo assim, para que um projeto tenha sucesso torna-se indispensável a utilização de uma metodologia de gestão ágil, sendo recomendável recorrer a novas opções de gestão além das tradicionais já consolidadas (BUILDER, 2019).

Desta forma, esta seção traz uma abordagem dos conceitos, terminologias, características, objetivos e limitações que compõem o cenário proposto pela pesquisa. Ressaltando que ao fim da seção, são exibidas de maneira resumida, as pesquisas relacionadas previamente identificadas no processo de investigação do problema de pesquisa.

2.1. Metodologias Ágeis

Segundo Sommerville (2019), uma das diferenças notáveis entre os métodos tradicionais de desenvolvimento de sistemas de *software* e os métodos ágeis, encontra-se na capacidade em atender determinadas demandas de desenvolvimento. Pois, para a concepção de sistemas críticos, os métodos tradicionais (nos quais existe uma preocupação na formalização da especificação do projeto), mostraram-se eficazes. Porém, na criação de sistemas de pequeno e médio porte, tais métodos tradicionais são tidos como ineficazes.

Tal ineficiência mencionada e relacionada aos métodos tradicionais, está associada ao tempo gasto com a necessidade em se documentar e especificar o sistema, o que acaba reduzindo o tempo destinado às atividades de desenvolvimento e testes, diminuindo também a capacidade das equipes de gerenciar mudanças.

Desta forma, em contraponto a este arcabouço burocrático de desenvolvimento de *software*, são propostas as metodologias ágeis que idealizam um desenvolvimento mais voltado e focado na construção do produto propriamente dito, reduzindo a responsabilidade na criação de documentação (RAMOS, 2015). O objetivo geral dos métodos ágeis é realizar entregas contínuas ao cliente, em um curto período de tempo, e desta forma possuir respostas mais efetivas as solicitações de mudanças que venham a surgir bem como o incremento de novos requisitos que venham a ser solicitados (SOMMERVILLE, 2019).

O conceito ágil foi desenvolvido buscando respaldar principalmente o desenvolvimento de sistemas de negócios, nos quais existem uma constante mudanças nos requisitos durante todo o ciclo de desenvolvimento. Sendo assim, para atender com êxito tais objetivos, os métodos ágeis buscam possuir uma abordagem iterativa nas fases de especificação, desenvolvimento e entrega do *software*.

Dentre as metodologias ágeis mais conhecidas e utilizadas no mercado, na atualidade, temos o *Scrum* (SCHWABER, 2007), *Extreme Programming* (BECK, 1999), FDD (HEPTAGON, 2015), dentre outras.

2.2. Métricas de Software

Os *softwares* mostram-se cada vez mais como um dos maiores componentes orçamentários das organizações atuantes no mercado. Desta forma, é de vital importância que os custos com *software* sejam controlados e que sua performance seja analisada. Para tal, as métricas apresentam-se como necessárias para cumprir a finalidade de fornecer medidas objetivas, coesas e atualizadas.

De acordo com Pressman (2016), as métricas de *software*, para finalidade de medição, podem ser classificadas em duas categorias, que são: medidas diretas e indiretas. As medidas diretas são aquelas relacionadas ao custo e ao esforço aplicado tanto no desenvolvimento quanto na manutenção do *software*. Também pode-se enxergar como medidas diretas a quantidade de linhas de código produzidas, velocidade de execução e o total de defeitos registrados.

Por sua vez, as medidas indiretas de um produto podem incluir funcionalidade, qualidade, complexidade, eficiência, confiabilidade, dentre outros aspectos do produto. Vale frisar que, as medidas indiretas são reconhecidas como mais difíceis de serem medidas e avaliadas (PRESSMAN, 2016).

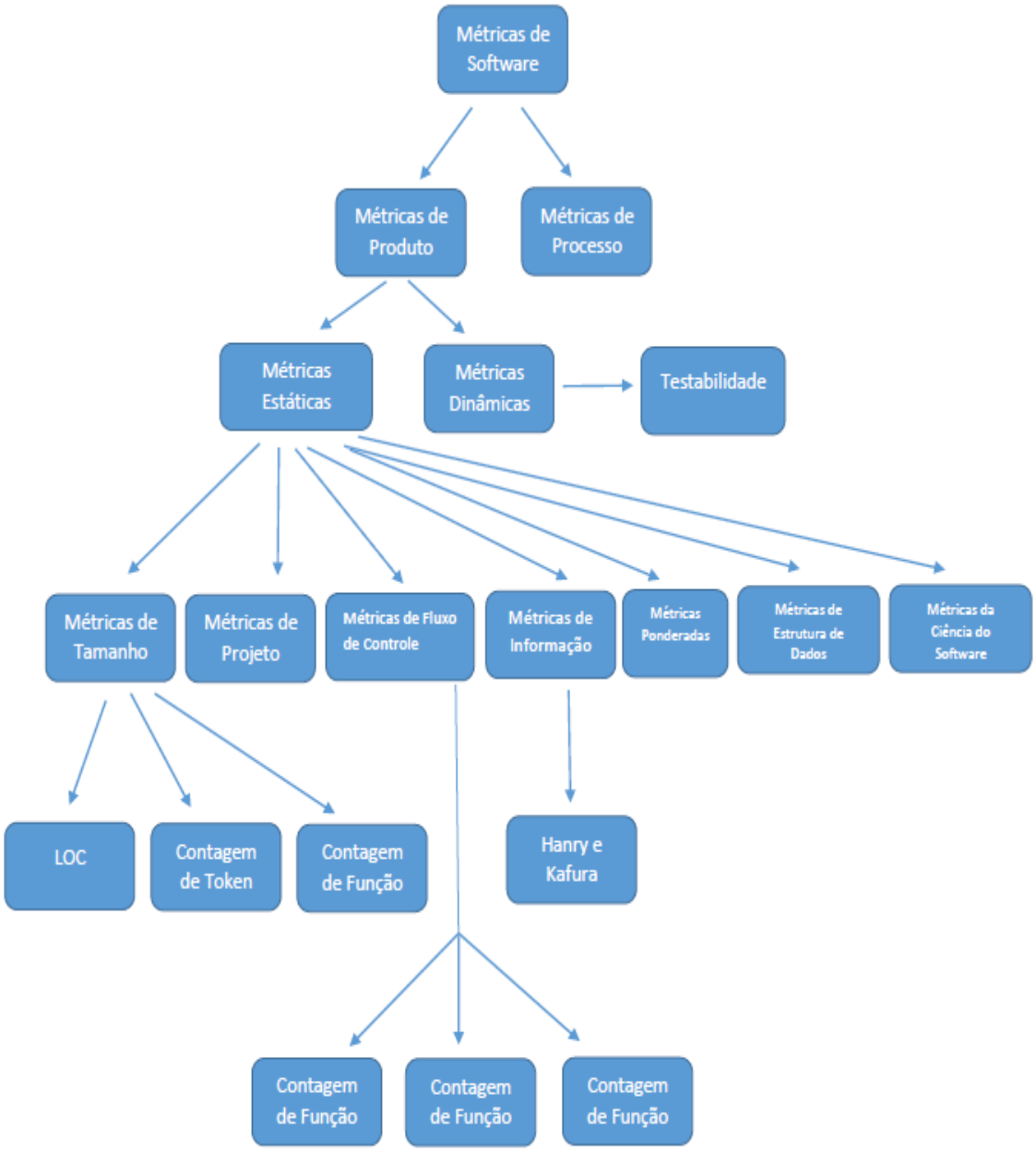
Também é possível dividir as métricas de *software* de acordo com suas respectivas aplicações. Sendo assim, no cenário de aplicabilidade de métricas encontram-se três categorias: métricas de produtividade, métricas de qualidade e métricas técnicas. Segundo Cordeiro (2000), as métricas de produtividade estão focadas na saída do processo de engenharia de *software*, as métricas de qualidade possuem seu foco no quanto o *software* atende aos requisitos especificados e as métricas técnicas centralizam-se nas características do produto, como por exemplo os níveis de complexidade lógica e modularidade.

Por fim, ainda é possível observar mais uma divisão de domínio das métricas defendida por Pressman (2016):

- ***Métricas orientadas ao tamanho:*** medidas diretas do *software* e do processo pelo qual é desenvolvido. Tais medidas derivam de atributos como linhas de código, custo, quantidade de documentação;
- ***Métricas orientadas à função:*** medida indiretas do *software* e do processo pelo qual é desenvolvido. Tais medidas concentram-se na funcionalidade ou na qualidade do *software*;
- ***Métricas orientadas às pessoas:*** utilizam informações sobre a maneira pela qual as pessoas desenvolvem *software* e percepções humanas sobre a efetividade das ferramentas e métodos.

A fim de auxiliar no entendimento do texto apresentado, bem como expor outras divisões presentes na literatura no que diz respeito ao tema domínio das métricas de *software*, a Figura 2, exibe de maneira hierárquica a classificação estipulada por Singh et al. (2011) onde o autor contempla em sua classificação diversas métricas e inclui exemplos associados de cada nível hierárquico.

Figura 2 – Classificação das métricas de *software*.



Fonte: Adaptação de (SINGH et al., 2011).

Sendo assim, dentre todas as divisões de domínio existentes para métricas de *software* apresentadas, esta pesquisa adota e foca na utilização das métricas de acordo com suas respectivas aplicações, onde as categorias de métricas por produtividade e por qualidade serão amplamente abordadas e revisitadas.

2.3. Importância das Métricas de Software

É notório afirmar que para deter o controle sobre determinado processo ou produto faz-se necessário medi-lo. Desta forma, Fernandes (1995) afirma que, a gestão de projetos e produtos de *software* apenas conseguem um determinado grau de eficiência e eficácia quando existem métricas e medidas que viabilizem a gestão baseada em fatos. Ressaltando a importância em frisar que, o processo é medido para sua constante melhoria e o produto é medido buscando um aumento exponencial em sua qualidade (PRESSMAN, 2016).

As métricas são necessárias para analisar a qualidade e produtividade dos processos de desenvolvimento/manutenção, bem como dos produtos de *software* desenvolvidos (CORDEIRO, 2000). Segundo Seibt (2001), as métricas técnicas são necessárias para mensurar a performance técnica dos produtos pelo ponto de vista do desenvolvedor. Já as métricas funcionais fazem o trabalho de mensurar a performance do produto pelo ponto de vista do usuário. Sendo assim, as medidas funcionais quando utilizadas de maneira independente das decisões do desenvolvimento técnico, podem ser empregadas para analisar a produtividade de diferentes tecnologias implementadas.

Para Pressman (2016), existem alguns motivos que justificam a necessidade por realizações de medições no desenvolvimento de *software*. Dentre estas razões podemos citar a facilidade em indicar o nível de qualidade do produto, a possibilidade em avaliar a produtividade das pessoas envolvidas nas atividades de desenvolvimento, a avaliação constante dos benefícios em termos de produtividade e qualidade advindos da implementação de novos métodos ou ferramentas, a formação de uma linha de estimativas e auxílio na elaboração de justificativas bem embasadas para aquisição de novas ferramentas ou para realização de novos treinamentos.

2.4. Utilização das Métricas de Software

O uso das métricas de *software* possibilitam a realização de uma das atividades mais importantes do processo de gestão, que é o planejamento baseado em evidências concretas. Desta forma, com a utilização de tais dados, é possível complementar a forma de fazer planejamento mais subjetivo baseado apenas nas experiências dos membros de uma equipe de desenvolvimento, passando a ter um planejamento mais concreto e robusto que busca mitigar riscos.

Segundo Brayner et al. (2008), a utilização das métricas de *software* podem auxiliar na busca por: entender melhor o comportamento e funcionamento dos produtos de *software*; determinar padrões, metas e critérios de aceitação; estabelecer o controle de processos, produtos e serviços de *software*; e prever valores para determinados atributos e cenários.

A utilidade das métricas é um fator a ser definido desde o momento em que há a decisão de implantá-las na avaliação do desenvolvimento do *software* (FUCK, 1995). Sendo assim, Fernandes (1995) afirma que a escolha pela utilização de métricas específicas, deve levar em consideração alguns pré-requisito básicos, que são:

a) O objetivo a ser alcançado com a utilização das métricas;

b) As métricas devem ser de fácil compreensão e utilização, visando assim subsidiar efetivamente os processos de tomadas de decisão;

c) As métricas devem ser objetivas, minimizando espaços para julgamentos pessoais na análise dos resultados;

d) O valor da informação obtida com o resultado das medições deve exceder o custo atrelado à coleta, armazenamento e cálculo das métricas. Sendo assim, as métricas devem ser efetivas no custo;

e) As métricas utilizadas devem propiciar informações que possibilitem a avaliação de acertos ou erros nas decisões e ações realizadas, bem como sejam capazes de prever a possibilidade de eventos futuros.

Ainda segundo Fernandes (1995), ao optar pela utilização das métricas de *software* é importante observar profundamente a sua utilidade no contexto do projeto ou no ambiente como um todo, além de atenuar-se em quais os tipos e categorias de métricas serão empregadas, os usuários de tais métricas, as pessoas para quais os resultados das métricas serão destinados e os níveis de aplicação.

É importante frisar que o processo de medição e avaliação necessita de um mecanismo para determinar quais dados devem ser coletados e como tais dados devem ser interpretados. Sendo assim, fica claro que o processo necessita de um mecanismo organizado para a determinação do objetivo da medição (FUCK, 1995). O fato de definir tal objetivo possibilita que algumas indagações sejam levantadas visando a definição de qual conjunto específico de dados possuirão o foco da coleta.

Desta forma, o objetivo da medição e análise de dados são consequências diretas da necessidade de uma determinada empresa, que segundo Seibt (2001), tais necessidades podem ser:

- Necessidade em avaliar determinada tecnologia;
- Necessidade em entender melhor a utilização dos recursos disponíveis visando melhorar a estimativa de custos;
- Necessidade atrelada a avaliação da qualidade do produto;
- Necessidade de avaliar as vantagens e desvantagens de um projeto.

Portanto, o objetivo principal na realização de medições em meio ao ciclo de desenvolvimento de *software* é a obtenção de maiores níveis de qualidade, considerando tanto o processo quanto o produto, e visando a satisfação dos clientes ou usuários a um custo economicamente viável.

2.5. Medição

A medição de *software* possui um papel importante ao fornecer para os responsáveis de determinado projeto as informações que permitem realizar um planejamento adequado, visando controlar o trabalho com exatidão e melhorando conceitos relacionados a um sistema mais seguro e confiável ao ponto de vista do cliente (SEIBT, 2001).

Segundo Fernandes (1995), as medições podem ser vistas como necessidades ou soluções para os seguintes cenários problemáticos observados no cotidiano das grandes corporações:

- Estimativas de prazos, esforços e custos realizadas com base no julgamento pessoal;
- Estimativa do tamanho do *software* não realizada;
- As métricas de produtividade no desenvolvimento não são mensuradas;
- A qualidade dos produtos finais e intermediários não são avaliadas;
- Os fatores de impacto na qualidade e produtividade não são levantados;
- A qualidade do planejamento não é avaliada;
- A capacidade de detecção e tratamento dos defeitos introduzidos durante o processo não é acompanhada;
- Não existem ações que busquem o aperfeiçoamento contínuo do processo de desenvolvimento ou de gestão de *software*;
- Não existe uma mensuração quanto ao nível de satisfação dos usuários.

Ainda segundo Fernandes (1995), para que seja possível medir os índices de qualidade e produtividade de projetos e produtos, comparando-os com metas preestabelecidas e verificando as tendências visando um aperfeiçoamento contínuo, faz-se necessário a adoção de alguns conceitos da engenharia de *software*, que são:

- Deter um processo de *software* bem definido quanto às políticas de desenvolvimento, procedimentos, técnicas, métodos e ferramentas;
- Realizar medições relativas a atributos de projeto (prazos, recursos, custo e esforço);
- Realizar medições relativas a atributos de processo (cobertura de testes, nível de complexidade do projeto e produtividade);
- Realizar medições relativas a atributos de produto (confiabilidade, nível de detecção de defeitos e tamanho do *software*);
- Realizar medições relativas ao nível de satisfação do cliente;
- Deter um processo estruturado de controle e garantia da qualidade, incluindo planos de qualidade, técnicas de teste de sistema e gestão de configuração.

Sendo assim, baseado nos conceitos expostos, fica notório que para conseguir aplicar de forma exitosa uma medição de *software* faz-se necessário antes de tudo identificar o objetivo da medição. Identificando assim, as características importantes a serem mensuradas e que gerem mais valor à base de conhecimento da organização.

É importante frisar que a identificação das características relevantes a um produto e *software* não é algo bem definido, como ocorre por exemplo em outros ramos da engenharia. Pois, a engenharia de *software* é uma área da engenharia relativamente mais recente que as demais, acarretando assim uma base de conhecimento histórico menor e que também sofre influência direta da forte peculiaridade da própria natureza inovadora da atividade que exerce (BRAYNER et al., 2008).

2.5.1 Seleção de Métricas

A seleção das métricas de *software*, deve ter por foco garantir a utilização de tais dados da maneira mais eficiente possível. As métricas devem ser facilmente coletadas, entendidas, calculadas e testadas, gerando assim uma sugestão de melhoria contínua nas estratégias definidas. Vale frisar, que é importante obter as métricas desejadas o mais cedo possível no processo de desenvolvimento de *software*, daí a importância em ter celeridade na definição de quais métricas utilizar no contexto de projeto a ser avaliado (COSTA et al., 2013).

Inicialmente em um projeto de desenvolvimento de *software*, as primeiras métricas selecionadas necessitam ser simples, objetivas, diretas, facilmente obtidas e devem deter a capacidade de servir como base para futuras avaliações do processo, bem como para futuras seleções de novas métricas. Sendo assim, tal evolução no tocante a escolha de novas métricas, deve seguir os mesmos princípios utilizados na seleção inicial, podendo a partir de então, oferecer novas referências. Como também, métricas anteriormente selecionadas, podem e devem ser removidas caso seja identificado uma ineficiência ou imprecisão de avaliação dentro do processo na qual foi aplicada (BRAYNER et al., 2008).

É importante salientar, que a seleção das métricas de produtividade e qualidade de *software* deve também ter por foco promover um ambiente produtivo e de tranquilidade para a equipe de desenvolvimento, afastando eventuais desestabilizações promovidas em decorrência dos procedimentos avaliativos. Portanto, o emprego destes dados coletados e analisados, devem acarretar uma estimulação na busca pelos objetivos de desempenho pré-estabelecidos.

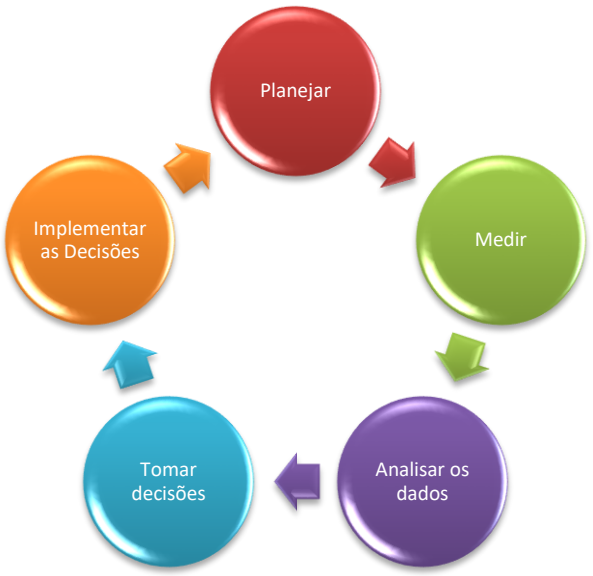
Segundo Kunz, Dumke e Zenker (2008), não existe um conjunto de métricas específico e universal para todos os projetos. Desta maneira, cabe o uso das métricas adequadas para cada realidade e a correta interpretação dos resultados (CARDOZO, 2020).

2.5.2 O Processo de Medição

Segundo Sommerville (2019), o processo de medição de *software* pode ser considerado como parte do processo de controle de qualidade, onde cada componente do sistema é analisado, e as divergências entre as métricas devem ser comparadas entre si.

Brayner et al. (2008) define um processo de medição baseado nos princípios da medição, composto por cinco atividades distintas, que são: Planejar, Medir, Analisar os Dados, Tomar decisões e Implementar as decisões. Em termos gerais, o processo consiste em planejar como as coletas e análises das métricas serão efetuadas e em seguida realizar a medição propriamente dita. Como resultado da medição obtêm-se os dados que servem de parâmetro para a realização das análises, e que por ventura norteiam as equipes a tomar eventuais decisões visando melhorias a serem posteriormente implementadas. Por fim, o processo é novamente retomado, de forma cíclica conforme observa-se detalhadamente na Figura 3, garantindo uma melhoria contínua.

Figura 3 – Processo de medição de *software*.



Fonte: Adaptação de Brayner et al. (2008).

É importante expor, que além das metodologias até então expostas, um outro processo de medição foi criado baseado nos princípios da medição propostos por Humphrey (1989), possuindo 4 papéis. Os papéis da mediação são:

- ***Entender*** – Métricas vistas como um facilitador no entendimento do comportamento e funcionamento de determinado processo ou produto;
- ***Avaliar*** – Métricas utilizadas na tomada de decisões e determinado o estabelecimento de padrões;
- ***Controlar*** – Métricas utilizadas no controle de processos e produtos;
- ***Prever*** – Métricas utilizadas na previsão dos valores de atributos.

2.6. Métricas no Processo de Desenvolvimento Ágil de Software

O correto e eficiente emprego de métricas junto ao processo ágil de desenvolvimento de *software* pode ter por consequência o aprimoramento na atuação da equipe. Desta forma, a utilização destes dados permite uma gestão que vise o aprimoramento do trabalho e que não deve ter por consequência acarretar um fardo nas atividades atribuídas aos membros do time de desenvolvimento (PADMINI et al., 2015).

Segundo Timóteo et al. (2008), um programa de métricas que gere relatórios robustos a respeito das metrificações analisadas, é uma abordagem que têm efeito direto na melhoria da qualidade dos produtos acarretando uma exponencial redução dos custos, promovendo assim uma evolução contínua nos processos de *software* metrificados.

Desta forma, Downey e Sutherland (2013) afirmam que, a implementação de métricas no processo ágil deve levar em consideração a facilidade e a simplicidade na manutenção de tais dados, onde faz-se necessário compreender as práticas adotadas pelas empresas e a forma como o monitoramento e controle de seus projetos são realizados, evitando *overhead* na adoção de métricas e buscando atender os pontos deficitários observados dentro de cada realidade.

Downey e Sutherland (2013) apontam dez métricas essenciais para serem utilizadas no processo ASD (*Agile Software Development*) e que possibilitam uma melhor tomada de decisão por parte das gestões. Sendo assim, as dez métricas sugeridas, são: Velocidade, Capacidade de Trabalho, Fator de Foco, Porcentagem de Trabalho Adotado, Porcentagem de Trabalho Realizado, Acurácia da Previsão, Aumento de Valor Alvo (TVI +), Escala de Sucesso e Registro de Vitórias/Derrotas.

Por sua vez, Pegoraro (2014) especifica um conjunto detalhado de métricas focadas no processo de ASD, que têm por foco não apenas explorar o objetivo atendido pela métrica propriamente dita, mas como também explorar o comportamento que tal medição desencadearia na equipe ou interessados. Sendo assim, torna-se possível prever os impactos positivos e negativos sobre um projeto de *software* específico.

As métricas propostas por Pegoraro (2014), e categorizadas pelos seus respectivos impactos são as seguintes:

- ***Métricas cujos resultados podem desencadear aceleração da equipe:*** Taxa de acerto na estimativa das tarefas, *Brump* da release/projetos, *Cycle time*, Número de integrações por dia, Número de tarefas não concluídas na iteração e *Throughput*;
- ***Métricas cujos resultados podem desencadear desconforto na equipe:*** Número de horas efetivamente trabalhadas em atividades de desenvolvimento e Total de horas trabalhadas no projeto;
- ***Métricas cujos resultados podem desencadear ações de melhoria do fluxo da tarefa:*** Diagrama de fluxo cumulativo das tarefas e *Work in Progress*;

- ***Métricas cujos resultados podem desencadear ações de minimização do surgimento de defeitos:*** Defeitos em aberto/Defeitos não solucionados e Defeitos encontrados pelo cliente.
- ***Métricas cujos resultados podem desencadear ações de aprendizado referente ao custo-benefício do projeto:*** Tempo investido em tarefas não planejadas, Variação do número de defeitos entre iterações e Variação dos custos (estimados vs. realizados);
- ***Métricas cujos resultados podem desencadear ações de desenvolver funcionalidades que atendam melhor às necessidades do cliente:*** Funcionalidades testadas, Funcionalidades Entregues e Grau de atendimento aos requisitos funcionais.

É importante observar que alguns conjuntos de métricas, que não são exclusivamente dedicados a medir métodos ágeis, podem e são utilizados pelas empresas em seus cotidianos dando inclusive um aspecto mais humano e social para as medições. Tais métricas de *software* contribuem com as equipes que utilizam o ASD na obtenção de um melhor controle sobre as atividades, tais métricas observadas são: Acurácia das estimativas, Funcionalidades testadas e entregues, índice de defeitos encontrados na fase de testes, Nível de satisfação do cliente e Índice de motivação dos membros da equipe do projeto (PEGORARO, 2014).

É possível notar que outras métricas em questão, também não exclusivas ao processo ASD, já divergem quanto ao alinhamento com os princípios e práticas ágeis, entre tais métricas pode-se citar o Total de horas trabalhadas no projeto e o Número de horas efetivamente trabalhadas no projeto. Sendo assim, a aplicação de tais métricas pode acarretar impactos negativos na equipe, e para mitigar tais impactos é sugerível que ao aplicar tais medições não sejam avaliados os indivíduos, mas sim um acumulado da equipe (PADMINI et al., 2015).

De forma geral, o número de métricas que podem atender ao contexto ASD é amplo e abrange diferentes necessidades por informação, bem como por controle. Sendo assim, cabe às instituições realizarem uma seleção eficaz por quais métricas de fato melhor lhes atendem, baseando-se sempre em suas necessidades de projeto e processo. Vale ressaltar que, algumas métricas de fato possuem objetivos similares e que são intercambiáveis, podendo ser escolhidas entre aquelas que mais se enquadram na necessidade de medição da organização.

2.6.1 Dificuldades para Aplicação de Métricas em Ambiente ASD

Conforme pode ser observado ao longo deste capítulo, as metodologias ágeis vêm sendo adotadas pelos times de desenvolvimento em uma rápida velocidade nos últimos anos. Segundo Aktunc (2012), existe um progresso razoável na profundidade dos estudos elaborados que abordam a área das métricas de *software* (apesar da pouca quantidade de títulos publicados na área), porém muitos destes progressos esbarram na dificuldade em obter-se *insights* através de indicadores de qualidade e produtividade advindos de um ambiente no qual dá-se ênfase às interações individuais

sobre processo e no qual se prioriza ter-se artefatos de *software* em detrimento de uma documentação robusta.

É preferível que em um processo ASD sejam utilizadas métricas breves, focadas e de curto prazo, porém em um ambiente de constantes mudanças o maior desafio é fazer com que estas métricas reflitam a natureza empírica dos projetos ágeis. Segundo Heimann et al. (2007), desafios como o custo para implementação de um sistema de métricas e a escolha por métricas que realmente forneçam dados necessários para a correta tomada de decisão devem ser levados em conta, pois no contexto ASD os desafios mencionados são fortemente impactados pelo fato de não ser possível planejar os esforços de desenvolvimento com antecedência, visto que as metas e planos são constantemente ajustados com base nos *feedbacks* do cliente e em outros fatores.

Além dos desafios mencionadas, os autores Dingsoyr et al. (2012) afirmam que um dos maiores desafios para aplicação de métricas em projetos ágeis de desenvolvimento de *software* é na verdade o pouco volume de estudos nesta área, ressaltando que nos primeiros anos após o advento do Manifesto Ágil[4] não foi possível identificar sequer dez estudos empíricos que abordam o tema. Sendo assim, reafirma-se que dentre os tópicos associados aos métodos ágeis que ainda precisam ser evoluídos, está o gerenciamento de projetos sobre a perspectiva de monitoramento e controle

2.7. Tecnologias Utilizadas

As tecnologias utilizadas no desenvolvimento da plataforma proposta para monitoramento das métricas de qualidade e produtividade em projetos ágeis de *software* através da integração com ferramentas de gestão e testes, foram:

- ***JavaScript –*** é a linguagem de *script* interpretada por navegadores e que possui um vasto número de responsabilidades, que incluem desde requisições assíncronas para obtenção de dados a alterações dinâmicas de CSS (AMBLER, 2015).
- ***Node.js*** – é uma forma de executar a linguagem *JavaScript* fora de um navegador *web*. Um de seus principais objetivos é permitir a criação de aplicações escaláveis, de altíssima concorrência e que são orientadas a eventos, não bloqueando assim as infra estruturas que utilizam (NODE.JS, 2022).
- ***ReactJS*** – é uma biblioteca *JavaScript* de código aberto, voltada para aplicações *web*, tendo por objetivo auxiliar no desenvolvimento de aplicativos de uma só página (SPA), com códigos flexíveis e eficientes (VIPUL; SONPATKI, 2016). O *ReactJS* é orientado a componentes, possibilitando que estes possam ser encapsulados e gerenciem seu próprio estado. O *ReactJS* também pode ser definido com o V do MVC (Modelo, Visão e Controle).

[4] Manifesto Ágil - http://agilemanifesto.org/

- ***Next.JS*** [5] *– framework* para o *ReactJS* de código aberto e hospedado no *github* sob a licença MIT. Tem por foco aumentar a eficiência da produção ao possibilitar que várias funcionalidades sejam reunidas e disponibilizadas, como: renderização estática e híbrida de conteúdos, suporte ao *TypeScript*, sistema de rotas, *pre-fetching* e outros *plugins* que visam acelerar o desenvolvimento fornecendo uma estrutura completa já pré-configurada.
- ***Core.UI*** – biblioteca de componentes de interface do usuário para o *ReactJS*, de código aberto e hospedado no *github* sob a licença MIT. É escrita em *TypeScript* e facilita a utilização de componentes estilizados da biblioteca CSS (COREUI, 2022).
- ***JSON –*** "é um mecanismo de codificação/decodificação de valores para intercâmbio de dados. Possui uma sintaxe de alto nível, fácil de ser entendida, facilitando o trabalho dos programadores e dos computadores. Ele é nativo da linguagem *JavaScript*" (PULUCENO, 2012). O JSON também pode ser considerado como um formato de texto completamente independente das linguagens de programação.
- ***Visual Studio Code*** [6] – é um editor de códigos desenvolvido pela companhia *Microsoft*. É customizável e inclui suporte para depuração de códigos, controle de versionamento, complemento inteligente de código, dentre outros benefícios.
- ***GitHub*** – site de codificação social que possibilita o controle de revisão distribuído e o gerenciamento do código fonte. Disponibiliza uma rede entre desenvolvedores que estimula a transmissão de atividades entre os respectivos interessados (THUNG, 2013).
- ***Netlify*** [7] **–** serviço de hospedagem de sistemas baseado em tecnologias *front-end*, que permite automatizar o processo de aplicações estáticas e possibilita o processo de integração contínua.
- ***Open Project – "****software* de código aberto, têm por foco a gestão de projetos, permitindo definir as tarefas e os custos do projeto planejado e realizado respeitando as respectivas precedências, permite nomear os recursos (pessoas, materiais, máquinas) possui recursos básicos, como Gráficos PERT, *Gantt*, *Earned value management*, WBS" (PARDELINHA, 2013).

[5] Next.JS - https://nextjs.org/

[6] Visual Studio Code - https://code.visualstudio.com/

[7] Netlify - https://www.netlify.com/

- ***TestLink* – "**é uma ferramenta *open source* para o gerenciamento de testes. Ela permite os cadastros de planos e casos de testes, bem como permite o controle e a execução dos testes. Com o *TestLink* é possível que equipes de testes trabalhem de forma sincronizada mesmo que em locais diferentes. Por ter uma interface *Web* e permitir níveis de acesso diferenciados, analistas de testes podem gerar as especificações de testes para que outras equipes realizem a execução. Outra característica interessante é o controle de execuções, gerando uma base histórica dos testes aos quais a aplicação foi submetida" (CAETANO, 2007).

2.8. Trabalhos Relacionados

É possível observar em um primeiro momento e através de uma investigação prévia, que os estudos realizados nesta linha de pesquisa focam na apresentação das histórias de sucesso ou lições aprendidas por organizações que adotaram metodologias ágeis em seus projetos, porém havendo uma notória falta de pesquisas e avaliações quanto à adequação de tais metodologias aplicadas, estando assim a adoção de métricas de *software* entre os pontos mais deficitários neste ramo da engenharia. Vale salientar que, Mikulenas e Butleris (2010) apud Pegoraro (2014), já alertavam em sua obra que a busca por tais informações é realmente de difícil acesso, escassa e pouco aprofundada, sendo muitas vezes necessário o próprio pesquisador sugerir uma solução própria para a utilização de métricas de *software* a partir da teoria encontrada.

Apesar das dificuldades apontadas e mensuradas, levantou-se um conjunto de trabalhos acadêmicos que tivessem por foco o emprego de métricas de *software* em projetos ágeis e cujo conteúdo em algum momento abordasse a utilização de métricas de produtividade no processo de desenvolvimento de *software*, métricas de qualidade de *software* e ferramentas utilizadas no processo de medição do processo ou produto. No capítulo 3 será descrito de forma mais detalhada como tais estudos foram de fato selecionados, a partir da realização de uma revisão sistemática da literatura.

O trabalho *Use of Software Metrics in Agile Software Development Process* (PADMINI et al., 2015), identifica um conjunto de dez métricas utilizadas na gestão de projeto ágeis, dados estes obtidos a partir da realização de um *survey* aplicado em mais de vinte e quatro empresas, onde julga-se que as métricas de *software* apontadas possuem benefícios que sobrepõem o *overhead*. A partir dos resultados obtidos, Padmini et al. (2015) realizam uma subcategorização na qual apontam quais métricas estão ligadas a qualidade, quais referenciam a produtividade e quais referenciam a previsibilidade do projeto. Neste mesmo trabalho, ainda se destaca a utilização de ferramentas como *Jira* e *Greenhopper* para a extração das métricas selecionadas sem muito esforço. Apesar de ser um estudo bem completo e elaborado, no tocante a identificação e sugestão de métricas que possam vir a ser utilizadas em projetos ágeis com o intuito de agregar mais valor à gestão, os autores não colocam em prática em um projeto real o processo de medição utilizando as dez métricas sugeridas operando de forma simultânea, como também não informam que as ferramentas para extração de

métricas sugeridas não são *open-source* e também não são amplamente disseminadas pela comunidade, muitas vezes devido ao seu alto preço de aquisição.

Por sua vez, Boerman et al. (2015) propõem um total de dezessete métricas de *software* a serem utilizadas em projetos ágeis, ressaltando a importância no uso de tais dados na facilitação da comunicação com os *stakeholders.* Boerman et al. (2015) realiza a mensuração das métricas que propõem, aplicando de fato a medição em projeto e utilizando a ferramenta *Jira* para coleta, análise e síntese dos dados. Porém, este trabalho possui como foco claro estabelecer métricas que estejam alinhadas com os objetivos do *product owner,* não havendo um foco em utilizar métricas que sejam categorizadas como diretamente ligadas a qualidade ou a produtividade de um projeto ou produto.

Quanto a utilização de aparatos que visem prestar suporte ao uso de métricas de *software,* Linke et al. (2008) exibem um levantamento contendo algumas ferramentas que detém tal propósito, dentre elas as principais são: *Anlyst4j, Chidamber & Kemmerers, Java Metrics, Dependency Finder, Eclipse Metrics 3.4, OOMeter, Understand for Java e VizzAnalyzer.* No entanto, a maioria destes programas tem por foco a extração e visualização de métricas orientadas a objetos, dando suporte ao uso de métricas como Linhas de Código, Número de Métodos, Número de Filhos, Métricas de Acoplamento entre classes e objetos, dentre outras. Linke et al. (2008) não especifica claramente a utilização destes instrumentos associados às métricas de produtividade ou qualidade, porém deixa claro a existência de diversas outras ferramentas que servem de suporte à criação e elaboração das métricas de *software.* Evidenciando assim, o grande leque de oportunidades a serem desbravadas neste campo de pesquisa referenciado.

Observou-se um claro consenso no que diz respeito ao aumento na utilização dos métodos ágeis, que trazem consigo modos de avaliação incompatíveis com os valores e princípios ágeis. Pois, muitas empresas adotam processos ASD sem de fato saber quais fatores devem ser medidos ou controlados. Sendo assim, o trabalho de Pegoraro (2014) respalda tal afirmação, apoiando-se na possibilidade de desenvolver um processo de medição alinhado com os princípios do desenvolvimento ágil de *software*. Para tal, a autora define um conjunto de métricas adequadas às necessidades de monitoramento e propõe o processo de medição, com foco na compatibilidade com a abordagem ágil. A pesquisa apresenta uma grande completude naquilo em que se propõe, exibindo detalhadamente como aplicar as métricas, calculá-las corretamente e analisar os resultados obtidos, porém assim como nas demais pesquisas analisadas, a aplicação prática daquilo que foi proposto não ocorreu. Vale ressaltar que Pegoraro (2014), não sugere ferramentas específicas para apoio na utilização de métricas de *software*, visto que a peculiaridade das métricas sugeridas requisitaria uma plataforma personalizada para atender tamanha necessidade.

Faz-se necessário também expor os resultados obtidos por Cardozo (2020), pois apesar da premissa de sua pesquisa ser bem semelhante ao observado em Pegoraro (2014), o autor propõe como resultado final mais do que um conjunto de métricas de *software* alinhadas com os princípios ágeis, na realidade é proposto um *guideline* acessível por meio de ferramenta web, que visa apoiar

a seleção de métricas a serem utilizadas em projetos que adotem métodos ágeis. O trabalho de Cardozo (2020) de fato propôs uma plataforma *web* customizada e acessível, mas é importante frisar que a ferramenta web auxilia na escolha da métrica a ser utilizada mediante ao cenário mas não auxilia na utilização da mesma.

Outro trabalho de extrema relevância e de similaridade para com esta pesquisa é o *Effective Monitoring of Progress of Agile Software Development Teams in Modern Software Companies - An Industrial Case Study* (MEDING, 2017) no qual através de um *survey* aplicado em um time ágil de desenvolvimento, os autores elencam as principais métricas a serem coletadas e desenvolvem uma plataforma *web* que têm por foco disponibilizar *dashboards* personalizados contendo tais métricas pré-definidas. Esse projeto sem dúvidas, apresenta *insights* extremamente relevantes para esta pesquisa, porém o seu foco encontra-se em avaliar como os times responderiam em seus cotidianos ao uso das métricas que eles mesmo pré-selecionaram para medição, fazendo com que o protótipo de plataforma de métricas não estivesse no foco do estudo. Sendo assim, a plataforma proposta não possui integração com outros repositórios de dados, depende de inserção de dados através de planilhas e não se discute a fundo como o sistema foi desenvolvido ou como o mesmo pode ser evoluído.

Posto isto, os trabalhos mostrados nesta seção possuem uma grande contribuição dentro de seus contextos específicos de aplicação, detendo inúmeros méritos e respaldos. Porém, na realização desta pesquisa, não foram identificados cenários que aplicassem em projetos reais as métricas de *software* propostas para mensurar a qualidade e a produtividade com foco em projetos ágeis de *software*, a fim de realizar uma análise precisa e que buscasse promover uma melhor visão tanto do processo quanto do produto constituinte do panorama avaliado.

Ou seja, as pesquisas levantadas não exploraram amplamente a aplicação de um suporte ferramental que buscasse auxiliar a gestão e acompanhamento dos projetos através do agrupamento de informações produzidas em diversas ferramentas utilizadas no ciclo de desenvolvimento, objetivando consolidar este fluxo de dados em métricas de valor. Também pouco se discutiu sobre quais estratégias deveriam ser empregadas na extração dos dados dos mais diversos *softwares* de gestão de projetos utilizados no mercado, fazendo com que não exista uma proposta arquitetural consolidada para tal propósito e acarretando assim um efeito colateral ocasionado pela ausência de referencial, pois ao buscar-se unificar informações contidas em ferramentas como o *Open Project* e o *TestLink*, não é possível encontrar um guia de referência que conduza em tal processo de consolidação dos dados.

Sendo assim, em observância com as lacunas de pesquisa observadas na área, a proposta inicial deste projeto é investigar como as métricas de qualidade e produtividade podem ser extraídas das bases de dados dos *softwares* utilizados no processo de desenvolvimento para gerenciar os projetos, e como desenvolver uma plataforma que possa aplicar um processo de medição das métricas de *software* no contexto ágil, levando também em consideração as métricas definidas nos

trabalhos relacionados, promovendo como saída a visualização dos dados em formato de *dashboards* personalizados.

3. REVISÃO SISTEMÁTICA DA LITERATURA

Este capítulo apresenta inicialmente uma exposição detalhada do protocolo utilizado para realização da Revisão Sistemática da Literatura (RSL) com foco na busca por trabalhos de relevância para esta pesquisa. Também apresenta os resultados da RSL, buscando descrever e categorizar os trabalhos selecionados, relacionando-os com as questões de pesquisa elencadas.

3.1. O Protocolo

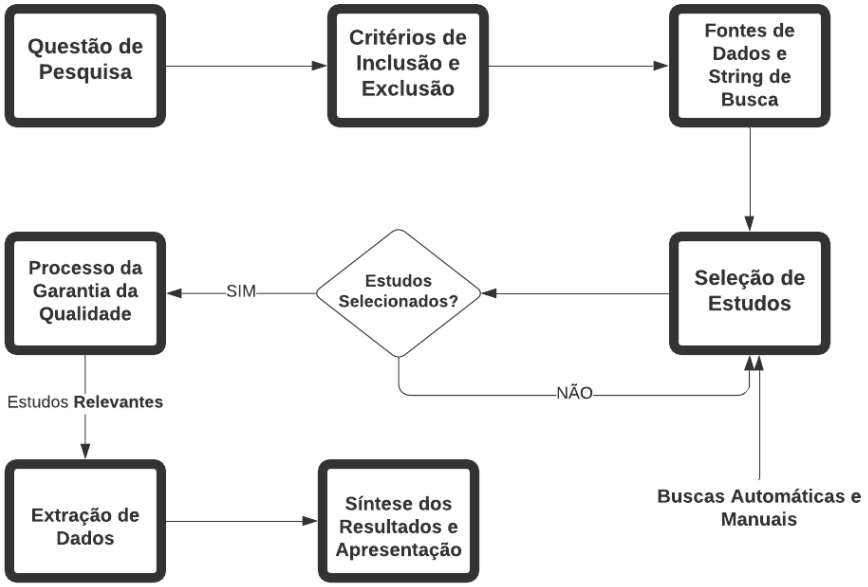
A Revisão Sistemática da Literatura (RSL) é um método que tem por foco a busca pelo domínio de temas específicos, visando a obtenção de um embasamento teórico sólido e pleno. Ainda segundo Medeiros et al. (2015), a aplicação de uma RSL tem por vistas analisar, identificar, interpretar e reportar os estudos mais relevantes que se encontram disponíveis para resolução de determinadas questões de pesquisa.

Para realização da RSL, foi desenvolvido um protocolo de pesquisa seguindo o que determina os "Procedimentos de Execução de Revisões Sistemáticas em Engenharia de *Software*", de Kitchenham et al. (2007), buscando assim deixar os resultados mais confiáveis, auditáveis e possíveis de serem replicados.

Desta maneira, tal protocolo de pesquisa concebido visou guiar a execução de uma RSL e identificou o estado da arte em métricas de qualidade e produtividade de projetos ágeis de *software*, norteando as etapas de seleção dos estudos, extração e síntese dos dados, além de definir as bases de dados a serem utilizadas, bem como a *string* de busca empregada nas consultas automáticas de tais bases previamente selecionadas.

Esta pesquisa foi realizada em sete fases, conforme pode ser melhor observado na Figura 4, e para sua condução fez-se necessário uma equipe contendo cinco integrantes, sendo dois estudantes de graduação, um estudante de pós-graduação e dois professores orientadores que nortearam o processo.

Figura 4 –Fases da RSL.



Fonte: Produzido pelo autor.

As fases da pesquisa destacadas anteriormente, constituem atividades que segundo Kitchenham (2007) podem ser divididas em três fases: planejamento, condução e relatório. Tais fases podem ser melhor observadas na Figura 5.

Figura 5 – Atividades do processo de execução da Revisão Sistemática da Literatura.

Fonte: Kitchenham (2007).

Na fase de planejamento, o protocolo de pesquisa foi construído, explicitando as questões e os objetivos da RSL. Na fase de Condução, deu-se a seleção dos estudos e a síntese dos dados, para

tal conjuntura foi definida uma *string* de busca a ser executada nos principais engenhos de busca de artigos científicos e utilizando-se de critérios de exclusão pré-estabelecidos, realizou-se a filtragem dos resultados obtidos. Os artigos foram indexados e armazenados em um *software* online dedicada à gestão de RSL chamada *Parsif.al*[8], onde também nessa ferramenta realizou-se uma avaliação de qualidade dos artigos pré-selecionados. Na última fase, foram classificadas e tabuladas as incidências de cada questão de pesquisa sobre os artigos escolhidos, utilizando a ferramenta de análise dos dados qualitativos e métodos mistos conhecida por *MAXQDA*[9].

3.1.1 Objetivo e Questões da RSL

O principal objetivo deste trabalho foi compreender as estratégias utilizadas para extração e disponibilização de métricas de qualidade e produtividade em projetos ágeis de *software*. Desta forma, foi definida a seguinte questão geral de pesquisa (QP1, conforme descrito anteriormente na seção 1.3): quais são as estratégias utilizadas para extração e disponibilização de métricas de qualidade e produtividade em equipes que desenvolvem projetos ágeis de *software*?

A fim de buscar facilitar as etapas de condução da RSL, bem como a síntese dos resultados, a QP1 foi detalhada nas seguintes sub-questões específicas:

- ***QPE1.1:*** Quais plataformas estão sendo utilizadas para facilitar a extração e visualização de métricas de qualidade e produtividade durante a execução das atividades realizadas em projetos ágeis de *software?*
- ***QPE1.2:*** Quais estratégias estão sendo utilizadas para integração com ferramentas de gestão de projetos, controle de versão, testes, dentre outras, visando a coleta automática de dados sobre produtividade e qualidade?
- ***QPE1.3:*** Quais estratégias não automatizadas de extração e visualização das métricas de qualidade e produtividade, são aplicadas durante a execução das atividades em projetos ágeis de *software*?
- ***QPE1.4:*** Quais os impactos (positivos/negativos) são observados na gestão e na tomada de decisão, a partir do uso das plataformas de extração e visualização das métricas de qualidade e produtividade?

[8] https://parsif.al/

[9] https://www.maxqda.com/pt

3.1.2 Estratégias de Buscas Automáticas

Objetivando uma busca efetiva e abrangente, que garanta o máximo de cobertura possível para a pesquisa realizada, definiu-se que as fontes de pesquisas a serem utilizadas deveriam estar disponíveis via *web* e preferencialmente em bases de dados científicas da área da Engenharia de *Software*. Sendo assim, foram escolhidas as fontes de busca automáticas e manuais com acesso institucional permitida para o Instituto Federal da Paraíba - IFPB via Portal de Periódicos da CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior).

Sendo assim, foi definida uma *string* de busca para realização da procura automática, contendo os constructos integrantes da pesquisa mais seus respectivos sinônimos ou derivações, concatenados com operadores booleanos "*OR*" e "*AND*".

As palavras-chaves extraídas a partir da questão de pesquisa principal, bem como das questões específicas foram: Qualidade de *Software*; Métricas; Indicadores; Desenvolvimento de *Software*; Produtividade no Desenvolvimento de *Software*; Gerenciamento de Projetos de *Software*; Projetos Ágeis de *Software*; Painel de Gerenciamento em Projetos Ágeis; Tomada de Decisão; Extração e Visualização de Métricas; Integração de Plataformas e Coleta Automática de Dados. Desta forma, com o objetivo de obter o maior número possível de artigos relacionados ao tema, utilizou-se para a construção da *string* termos abrangentes que compuseram a seguinte *string* derivada:

(("software" OR "information system engineering" OR "information system development") AND ("agile" OR "agility") AND ("project management" OR "decision making" OR "software quality" OR "quality of software" OR "dashboard" OR "management panel" OR "software development productivity" OR "software team productivity" OR "developer productivity") AND ("metrics" OR "indicators"))

As fontes de busca automática utilizadas nesta pesquisa estão expostas na Tabela 1. Tais fontes foram escolhidas, pois são citadas por Kitchenham (2007) como de extrema relevância para a área investigada.

Tabela 1 – Fontes destinadas a buscas automáticas.

Engenho de Busca
Biblioteca Digital do IEEE (http://ieeexplore.ieee.org/Xplore/)
Biblioteca Digital da ACM (http://portal.acm.org/)
SCOPUS (http://www.scopus.com/home.url)
SpringerLink (http://springerlink.com)

Fonte: Produzido pelo autor.

3.1.3 Seleção dos Estudos

Os estudos obtidos como resultados do processo de busca, passaram por análises baseadas nos critérios de inclusão e de exclusão expostos na Tabela 2. Em geral, o artigo seria incluído se estivesse alinhado para com os critérios de inclusão mencionados, e excluído se atendesse pelo menos um dos critérios de exclusão apontados.

Tabela 2 – Critérios elencados para seleção dos estudos.

Critérios de Inclusão
IC1. Estudos aplicados na indústria ou academia, que tratem sobre uso de métricas de qualidade e produtividade na gestão de projetos de ágeis de *software;*
IC2. Pesquisas qualitativas ou quantitativas;
IC3. Estudos primários.
Critérios de Exclusão
EC1. Escrito em um idioma que não seja o inglês;
EC2. Estudos duplicados ou repetidos;
EC3. Estudos que não tratem de métricas de qualidade de *software* ou produtividade de *software*;
EC4. Estudos realizados a mais de 12 anos;
EC5. Estudos incompletos, rascunhos, slides ou resumos;
EC6. Estudos secundários, terciários e meta-análises;
EC7. Estudos que não abordam pelo menos uma métrica de qualidade e produtividade em projetos ágeis de *software*;
EC8. Artigos que não estejam disponíveis gratuitamente para *download* nos ambientes institucionais do IFPB;
EC9. Estudos teóricos que não apresentem nenhum tipo de validação.

Fonte: Produzido pelo autor.

Na etapa de seleção dos artigos, este projeto de pesquisa buscou seguir a proposta de trabalho apresentada por Medeiros et al. (2015), na qual é sugerido que inicialmente os pesquisadores realizem a leitura dos títulos e resumos dos artigos, aplicando os critérios de inclusão e exclusão pré-estabelecidos. Na fase seguinte, os critérios devem ser aplicados na leitura da introdução e da conclusão dos estudos resultantes da fase anterior.

Vale frisar que, objetivando reduzir o viés de pesquisa, a seleção dos estudos foi dividida entre duas duplas de pesquisadores. Sempre que conflitos e divergências eram identificados em uma dupla, o impasse era resolvido compartilhando a discussão com os membros da outra dupla. Se mesmo assim o impasse persistisse, o professor orientador do projeto era convidado a mediar tal impasse, buscando resolver a situação da melhor forma possível.

3.1.4 Avaliação de Qualidade

Utilizando-se de um questionário adaptado de Dybå e Dingsøyr (2008), os pesquisadores realizaram uma análise de qualidade dos artigos resultantes da fase anterior, que porventura passaram pelo crivo da análise dos critérios de exclusão. Sendo assim, para que a qualidade dos trabalhos pudesse ser efetivamente aferida, foi realizada a leitura completa das obras selecionadas.

É importante frisar que, os artigos que durante esta etapa vieram a apresentar baixa aderência aos critérios (Inclusão e Exclusão), tiveram de ser excluídos. Pois, apesar de não terem sido excluídos na fase anterior, a leitura completa da pesquisa demonstrou a necessidade de não levar adiante a avaliação de qualidade de alguns títulos anteriormente selecionados.

Os artigos foram analisados por quatro pesquisadores, que tiveram acesso ao questionário de qualidade dos estudos apresentado na Tabela 3. Sendo assim, para cada questão aplicada foi utilizada a escala de três pontos de Likert (1932), visando obter uma pontuação do artefato validado:

- ***0***: Não existe nada no artigo que atenda ao critério avaliado;
- ***0.5***: O artigo não deixa claro se atende ou não ao critério;
- ***1***: O artigo atende ao critério avaliado.

Tabela 3 – Questionário de qualidade aplicado aos estudos selecionados.

Critérios de Qualidade
1. Trata-se de um artigo de pesquisa?
2. Os objetivos da pesquisa são claros?
3. O contexto do estudo encontra-se claro?
4. O método da pesquisa foi justificado?
5. A amostragem foi representativa?
6. Os dados foram coletados de maneira a atender adequadamente a questão de pesquisa? (Está claro como os dados foram coletados?)
7. Os dados coletados justificam os resultados obtidos na pesquisa? (O processo de análise dos dados foi descrito?)

8. Vieses foram considerados?
9. Os resultados foram claramente descritos?
10. O estudo apresenta uma discussão a respeito da contribuição para prática ou para literatura?

Fonte: Produzido pelo autor.

O *Parsif.al*, ferramenta online utilizada para apoiar os pesquisadores na realização de RSL no contexto da Engenharia de *Software*, concede suporte à execução do processo de avaliação de qualidade dos artigos selecionados. Porém, foi necessário efetuar uma adaptação no ferramental buscando realizar a equivalência entre a escala de três pontos de Likert (1932) e a escala padrão utilizada pelo sistema. Posto isto, a pontuação 0 da escala foi colocada em equivalência ao parâmetro 'Não' da ferramenta, a pontuação 0,5 da escala foi colocada em equivalência ao parâmetro 'Parcialmente' da ferramenta e a pontuação 1 da escala foi colocada em equivalência ao parâmetro 'Sim' da ferramenta.

Desta maneira, quando o pesquisador selecionava os parâmetros do *software Parsif.al* dedicados a análise de qualidade (Não, Sim e Parcialmente), o sistema realizava a equiparação com seu valor respectivo na escala de Likert (1932). A partir do somatório das notas de todos os critérios, os artigos foram classificados em quatro faixas de qualidade conforme apresentadas na Tabela 4. Vale salientar, que os estudos cujos somatórios de notas lhe possibilitaram ser classificado nas faixas Alta e Muito Alta, tiveram como destino o encaminhamento para fase de extração. Já as demais pesquisas que não alcançaram tais faixas de qualidade, foram descartadas nesta etapa.

Tabela 4 – Pontuações para cada faixa de qualidade.

Baixa	Média	Alta	Muito Alta
$0 \leq N \leq 2,5$	$3 \leq N \leq 5,5$	$6 \leq N \leq 8,5$	$9 \leq N \leq 10$

Fonte: Produzido pelo autor.

3.1.5 Extração e Síntese dos Dados

Após as fases de seleção dos estudos e avaliação da qualidade, deu-se início a extração e síntese dos dados dos artigos resultantes da fase anterior. O processo teve por objetivo a realização da leitura completa de cada artigo, leitura na qual possibilitou aos pesquisadores extraírem dos conteúdos analisados as possíveis relações para cada questão de pesquisa elencada.

Esta etapa foi realizada por três pesquisadores. Os dados foram extraídos de forma independente por um pesquisador e revisados por outro, objetivando validar a eficácia da atividade e reforçar o resultado obtido. É importante frisar que, os conflitos foram resolvidos com a realização de reuniões que sempre obtiveram o consenso quanto à decisão final.

A ferramenta utilizada para execução da síntese e análise qualitativa das questões de pesquisa foi o *MAXQDA.* Ferramenta tal, que facilitou o controle dos dados extraídos graças a facilidade de se realizar robusta gestão dos constructos e de manter atualizado o trabalho desempenhado entre todos os pesquisadores via processo de junção (*merge*).

3.2. Resultados e Discussões da RSL

O foco desta seção está na exposição dos resultados obtidos com a realização da Revisão Sistemática da Literatura, sendo assim possível responder às questões de pesquisa elencadas para este trabalho.

Os resultados alcançados serão expostos em duas partes específicas:

- ***Visão geral dos estudos:*** métricas relacionadas a quantidade de estudos retornados e a quantidade de estudos excluídos por fase da pesquisa específica;
- ***Mapeamentos das evidências:*** respostas obtidas para cada questão de pesquisa.

3.2.1 Visão Geral dos Estudos

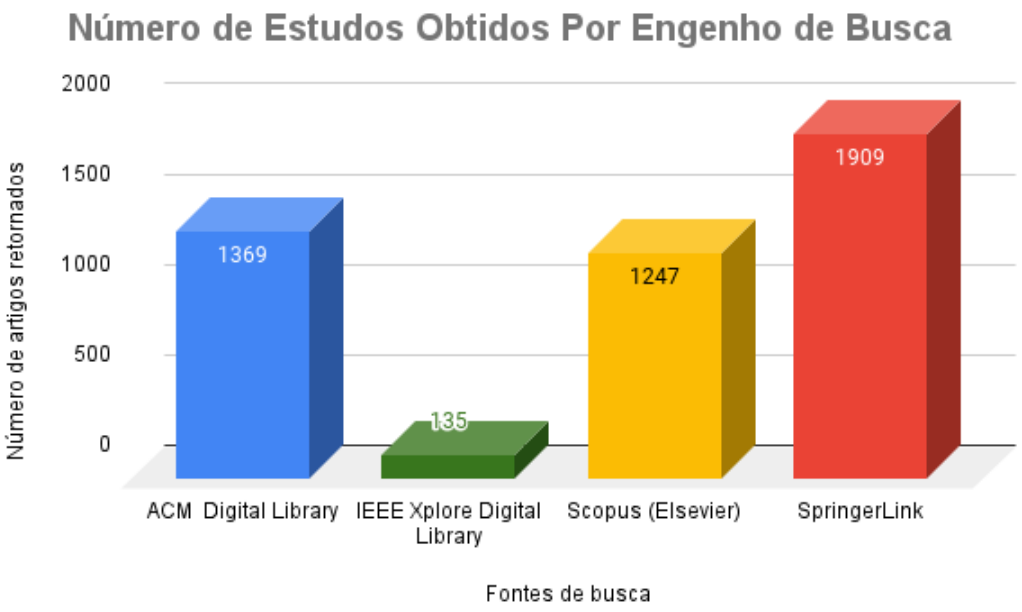
A atividade de coleta dos trabalhos ocorreu em três fases, que foram:

- Realização de buscas primárias, entre os anos de 2008 a 2020, nos principais engenhos;
- Seleção dos trabalhos, buscando respeitar os critérios de inclusão e exclusão;
- Avaliação de qualidade dos trabalhos selecionados, alocando-os em faixas de qualidade de acordo com a pontuação obtida.

Inicialmente, os pesquisadores pretendiam apenas considerar os intervalos de 2010 a 2020, para realização das buscas nas fontes de dados. Porém, ao utilizar a *string* de busca para realização de pesquisas *ad-hoc* no *IEEE Xplore Digital Library* e no *Google Scholar*, observou-se de forma evidente uma série de publicações relevantes datadas entre os anos de 2008 e 2009. Baseando-se nesta prerrogativa, tornou-se plausível e justificável a ampliação das buscas por um período de doze anos.

A execução das buscas primárias retornou como resultado um total de 4660 estudos, dos quais todos foram provenientes de buscas automáticas. Na Figura 6, é possível ter um panorama detalhado do número de trabalhos obtidos em cada biblioteca digital pesquisada.

Figura 6 – Número de estudos obtidos por engenho de busca.



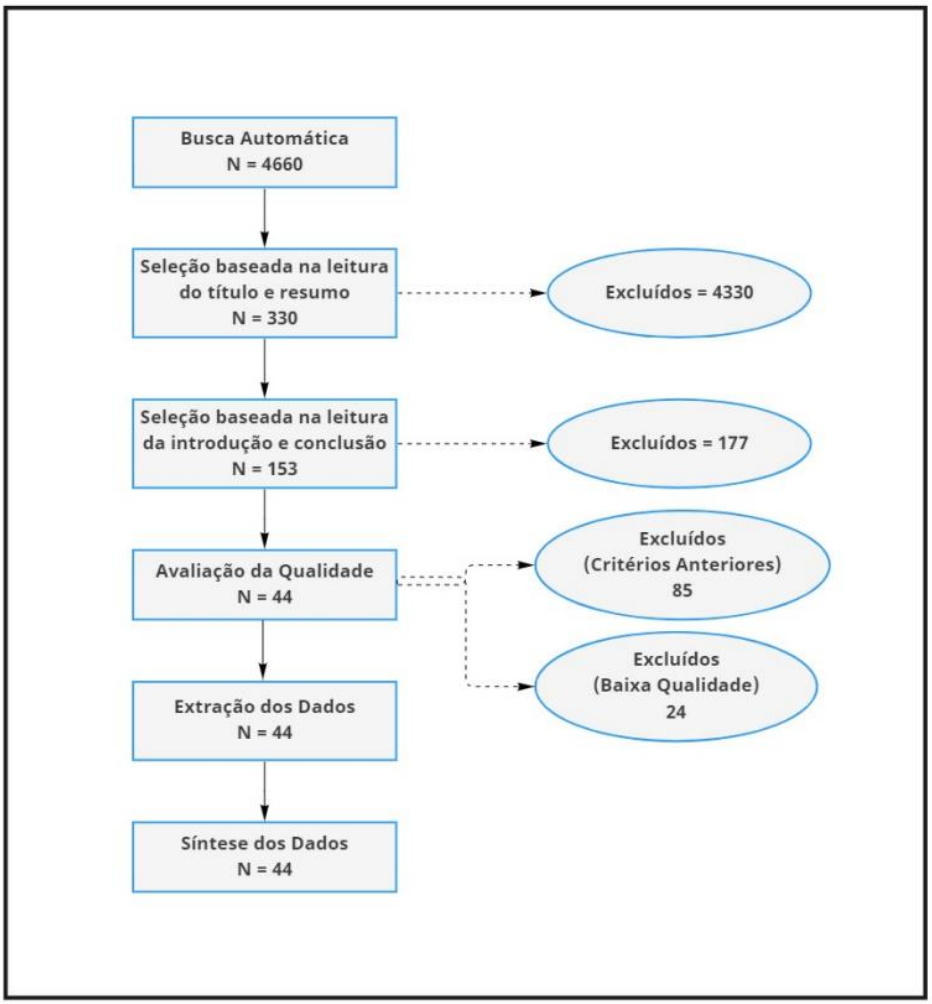
Fonte: Produzido pelo autor.

Dos 4660 estudos, 4330 foram excluídos na primeira etapa de seleção, na qual os pesquisadores realizaram a leitura do título e do resumo de cada material, restando assim um total de 330 estudos com potenciais reconhecidos. Na fase de seleção seguinte, a leitura da introdução e da conclusão possibilitou que fossem identificadas 177 obras aptas para exclusão, restando 153 estudos.

Após os estudos terem passado pelas duas primeiras etapas de seleção, foi então realizada a leitura completa das obras resultantes e consequentemente aplicada a etapa de avaliação da qualidade. Sendo assim, realizou-se a exclusão de 85 artigos, pois foi percebido que os mesmos deveriam ter sido excluídos em etapas anteriores por não estarem dentro dos critérios de inclusão pré-estabelecidos. Ainda na etapa de avaliação da qualidade, outros 24 artigos foram devidamente excluídos devido a sua baixa qualidade.

Por fim, restaram 44 estudos para extração dos dados, etapa esta que não registrou a exclusão de mais nenhum outro artigo. A Figura 7 exemplifica melhor como realizou-se a exclusão dos estudos em cada fase da RSL.

Figura 7 – Estudos por fase.



Fonte: Produzido pelo autor com base em Medeiros et al. (2015).

A Tabela 5 apresenta um melhor panorama no tocante a quantidade de artigos selecionados por biblioteca digital específica. Sendo assim, é possível afirmar que a maior parte dos estudos selecionados foram provenientes da *ACM Digital Library*, contabilizando 19 trabalhos ao todo. Já os engenhos de busca *IEEE Xplore Digital Library* e *Scopus* (*Elsevier*), contabilizaram respectivamente onze e quatorze estudos selecionados para cada um.

Faz-se importante salientar que nenhum dos artigos selecionados foram provenientes da base *SpringerLink.*

Tabela 5 – Quantitativo de estudos incluídos e excluídos, por engenho de busca.

Fontes de busca	Seleção Inicial	Excluídos			Total de Estudos Selecionados
		Critérios de Inclusão e Exclusão	Baixa Qualidade	Total de Exclusão	
ACM Digital Library	1369	1339	11	1350	19
IEEE Xplore Digital Library	135	118	6	124	11
Scopus (Elsevier)	1247	1227	6	1233	14
SpringerLink	1909	1908	1	1909	0

Fonte: Produzido pelo autor com base em Medeiros et al. (2015).

Vinte e quatro estudos foram excluídos por apresentarem a qualidade indesejada para os parâmetros estipulados nesta pesquisa. De acordo com a Figura 8, é possível observar as classificações dos 68 estudos que passaram pela etapa de avaliação da qualidade, dentre eles quatorze foram classificados como possuindo uma qualidade muito alta (20%), trinta como detentores de uma qualidade alta (44%), vinte e quatro estudos possuindo uma qualidade média (35%) e nenhum (0%) foi classificado como detentor de qualidade baixa.

Figura 8 – Quantitativo de estudos incluídos e excluídos, por faixa de qualidade.

Fonte: Produzido pelo autor.

3.2.2 Mapeamento das Evidências

Nesta seção serão apresentados os mapeamentos das evidências encontradas em cada estudo analisado, expondo assim os resultados da revisão sistemática por questão de pesquisa específica.

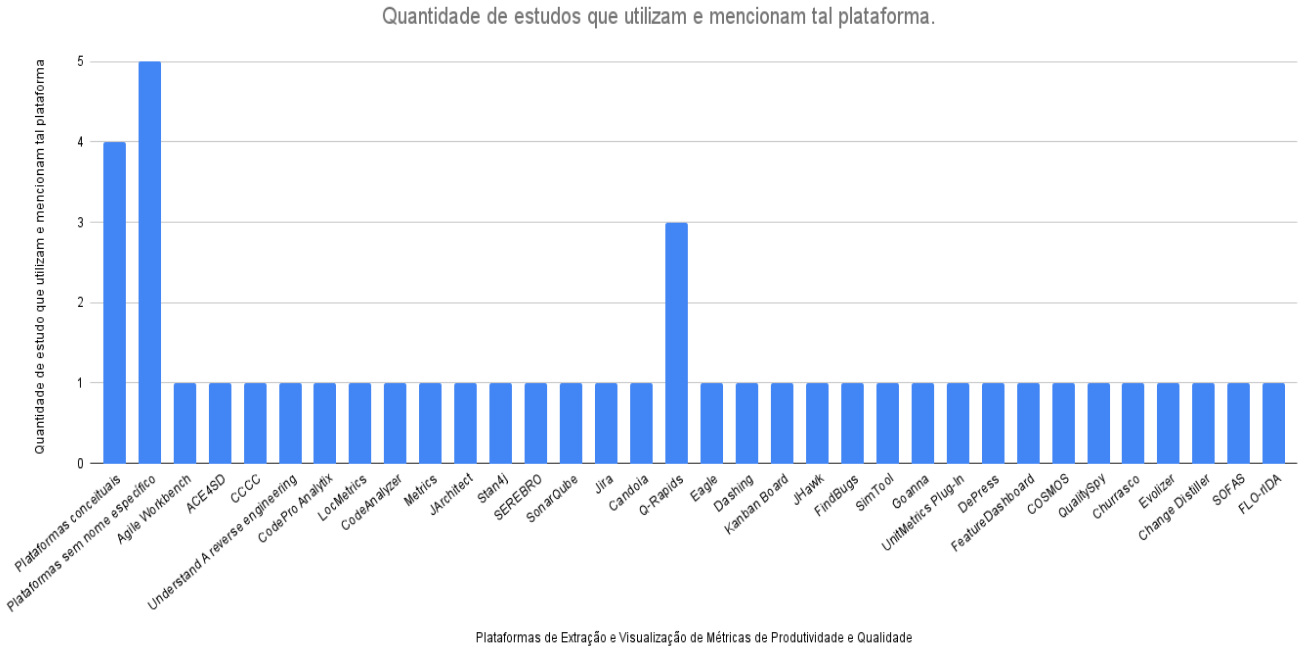
3.2.2.1 QPE1.1: Quais plataformas estão sendo utilizadas para facilitar a extração e visualização de métricas de qualidade e produtividade durante a execução das atividades realizadas em projetos ágeis de software?

No ecossistema ágil de desenvolvimento de projetos, é corriqueiro a utilização de diferentes ferramentas que por si só geram dados e métricas das mais variadas formas. Sendo assim, é cabível existir uma preocupação quanto a variação na coleta e exibição de métricas entre diferentes projetos, principalmente quando tais projetos precisam ser monitorados, medidos e comparados por um mesmo gestor ou vertical responsável.

Desta forma, para garantir consistência na extração e visualização de um dado conjunto de métricas, podem ser utilizadas plataformas que visem criar camadas de abstração, possibilitando assim uma plena integração com ferramentas populares utilizadas no desenvolvimento ágil de *software*. Objetivando evitar disparidades entre ferramentas, as plataformas de extração e visualização de métricas capturam os dados de forma independente, calculam automaticamente as métricas desejadas e criam espaços consistentes de visualizações (SHARMA e KAULGUD, 2016).

A síntese dos estudos apontou um total de 43 diferentes plataformas utilizadas para extrair e expor métricas em projetos de desenvolvimento ágeis de *software*, conforme pode ser melhor observado na Figura 9. A falta de uniformidade no tocante ao uso de tais plataformas é algo que chama atenção, pois em sua maioria cada estudo analisado emprega uma maneira diferente de coletar, analisar e expor as métricas desejadas. Neste contexto, ainda de forma mais específica, é possível observar um grande número de plataformas que sequer possuem um nome definido, circunstância esta observada em cinco diferentes estudos cujas plataformas foram desenvolvidas para atender única e exclusivamente a um contexto específico.

Figura 9 – Quantitativo de estudos que fazem referência a uma determinada plataforma de extração e visualização de métricas.



Fonte: Produzido pelo autor.

Além das plataformas que não possuem uma nomenclatura definida e que foram criadas para validação do contexto delineado de um dado escopo, também chamam a atenção as plataformas conceituais, que contabilizam um total de 4 plataformas das 43 totais contabilizadas. Estas plataformas conceituais não chegaram de fato a serem implementadas, apenas alguns pontos de suas arquiteturas foram efetivamente elaborados, como pode ser observado no estudo de Bukhari et al. (2018), no qual é proposto uma estrutura conceitual de seleção de métricas de *software* (SMeS), onde os principais elementos da estrutura conceitual proposta incluem o processo de seleção de métricas baseado em metas. Porém, o estudo mencionado não chega a implementar de fato a arquitetura da plataforma idealizada.

Dentre as plataformas identificadas, é importante salientar que 25 apresentaram um grande enfoque na obtenção de métricas a partir do código fonte desenvolvido para o projeto analisado, não importando em integrar-se a outras fontes de dados. Já dentro das demais 18 plataformas que consideram o processo de desenvolvimento ágil como um todo, e não focaram apenas no código desenvolvido, destaca-se a plataforma *Q-Rapids* cuja relevância foi citada em três estudos de pesquisa distintos.

O *Q-Rapids* oferece uma avaliação contínua, que pode ser em tempo real, dos indicadores estratégicos para tomada de decisão. A ferramenta é constituída dos Módulos de Coleta de Dados, Módulos de Modelagem/Análise de Dados e Módulos de Tomada de Decisões Estratégicas. Também detêm conectores *Apache Kafka* para coleta de dados heterogêneos provenientes de fontes

de dados externas, ressaltando que tais conectores são desenvolvidos em formas de *plugins* customizáveis, alguns já nativamente disponibilizados e outros que podem ser criados especificamente para serem acoplados e funcionarem juntamente com o *Q-Rapids.*

Os estudos dos autores Martínez-Fernández et al. (2019) e Choras et al. (2020), usaram a plataforma *Q-Rapids* com a finalidade de relatar uma experiência prática sobre o uso de métricas relacionadas ao processo de desenvolvimento de *software* como meio de apoio às equipes ágeis, evidenciando assim o grande potencial da plataforma em questão quanto a coleta em tempo real de vários tipos de dados relacionados ao desenvolvimento. Já o estudo realizado pelos pesquisadores Kozik et al. (2018) buscou aprofundar-se na arquitetura do *Q-Rapids*, expondo suas capacidades e atribuições, no que diz respeito a elaboração de uma estrutura que reúne dados básicos de ferramentas diversas (*GitLab*, *SonarQube* ou *JIRA*), processando tais dados e objetivando calcular métricas avançadas, fatores de produto, indicadores e correlações entre eles.

3.2.2.2 QPE1.2: Quais estratégias estão sendo utilizadas para integração com ferramentas de gestão de projetos, controle de versão, testes, dentre outras, visando a coleta automática de dados sobre produtividade e qualidade?

Com a popularização e a larga aceitação dos princípios ágeis na indústria de *software*, uma grande lista de ferramentas vêm sendo utilizadas para planejamento e gestão dos projetos, dentre elas podemos citar como exemplo: *GitHub*, *GitLab*, *Jira*, *Redmine*, *Jenkins*, *Sonar Qube*, *Mantis*, dentre outras.

Sendo assim, buscando agregar as informações fornecidas pelas mais diversas ferramentas disponíveis e vinculá-las a métricas específicas, faz-se necessário estabelecer estratégias de integração e capturas de dados nas mais diversas fontes, consolidando tais informações para que seja possível a realização de uma análise concreta do processo de desenvolvimento de *software*, preenchendo lacunas e fornecendo conexões para com as métricas avaliadas (MARTÍNEZ-FERNÁNDEZ et al., 2019).

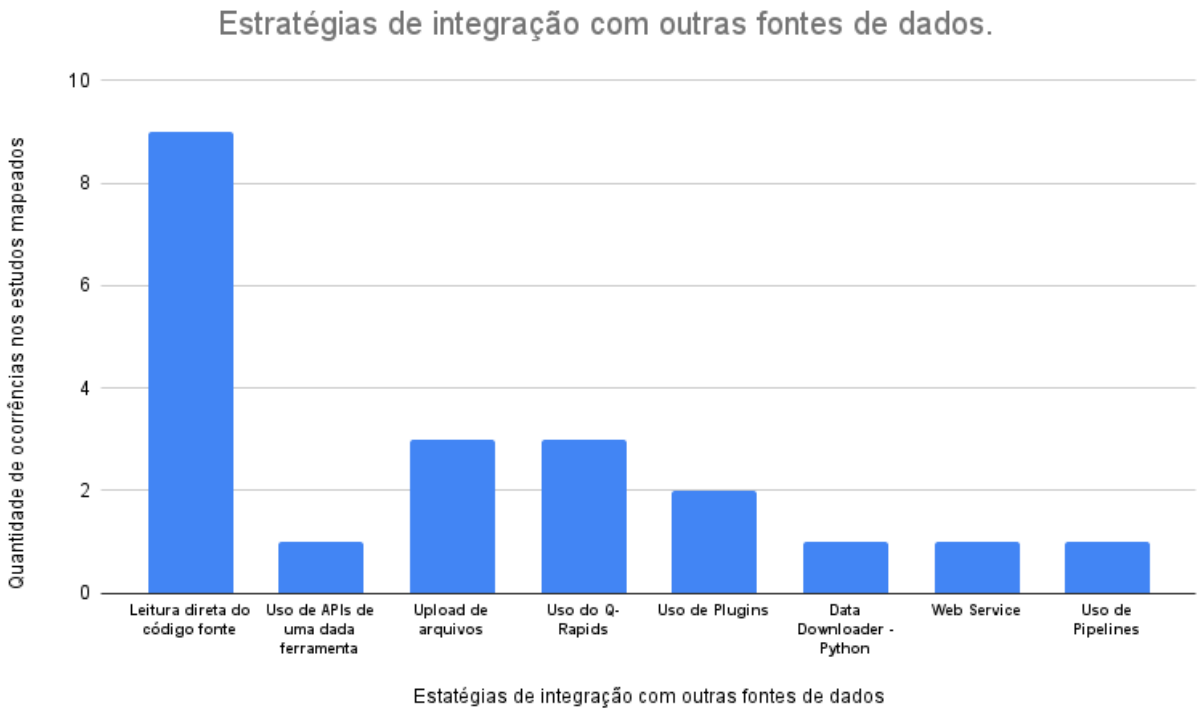
No contexto geral, uma estratégia de integração para coleta de informações pode ser entendida como, uma diminuição máxima de intrusão no momento da coleta, tendo por intenção economizar o tempo dos integrantes de um projeto para obtenção de dados, mantendo-os focados no ciclo de desenvolvimento do produto (SANTOS, 2013).

Após a realização da análise dos estudos, foram mapeados um total de 8 estratégias de integração com diferentes ferramentas de gestão de projetos, controle de versão, testes, dentre outras, visando a coleta automática de dados sobre produtividade e qualidade. Vale ressaltar que, estas 8 estratégias foram mencionadas em apenas 19 dos 44 estudos selecionados na RSL. E dentre os estudos que mencionam tais estratégias de integração, os artigos dos autores Dahab et al. (2018) bem como dos autores Deuter e Engels (2014), destacaram-se por apresentarem a utilização de 2

estratégias distintas para obtenção dos dados em ferramentas externas de gestão, cada pesquisa respectivamente.

Conforme pode ser verificado na Figura 10, as principais estratégias de integração com outras ferramentas de gestão de projeto, foram as seguintes: Leitura direta de código fonte do projeto, Uso de API's (*Application Programming Interface*) específicas de uma determinada ferramenta, *Upload* manual de arquivos (como *logs*, *xlsx* e outros), Uso do *software* de análises *Q-Rapids*, Uso de *plugins*, Aplicação e utilização da ferramenta *Data Downloader – Python*, Utilização de *Web Services* e Uso de *Pipelines* contendo várias ferramentas integradas.

Figura 10 – Quantitativo de estudos que fazem referência a uma determinada estratégia de integração com outra fonte de dados.



Fonte: Produzido pelo autor.

Definitivamente o maior número de ocorrências esteve associado à prática de captação dos dados diretamente do Código fonte do projeto, contabilizando um total de 9 ocorrências. Porém em sua maioria esta estratégia não possibilita ou não está associada com a possibilidade de integrar-se automaticamente a nenhum tipo de fonte de dados externa, levando a obtenção de métricas que estão mais alinhadas a qualidade do código e a produtividade em seu desenvolvimento.

De fato, existe uma variação na quantidade de estratégias abordadas, porém o uso do *Q-Rapids* e de *Upload* manual de arquivos são muito mencionados, possuindo ambas, 3 menções cada. Sendo assim, tendo por foco a integração e coleta automática de dados provenientes de outros

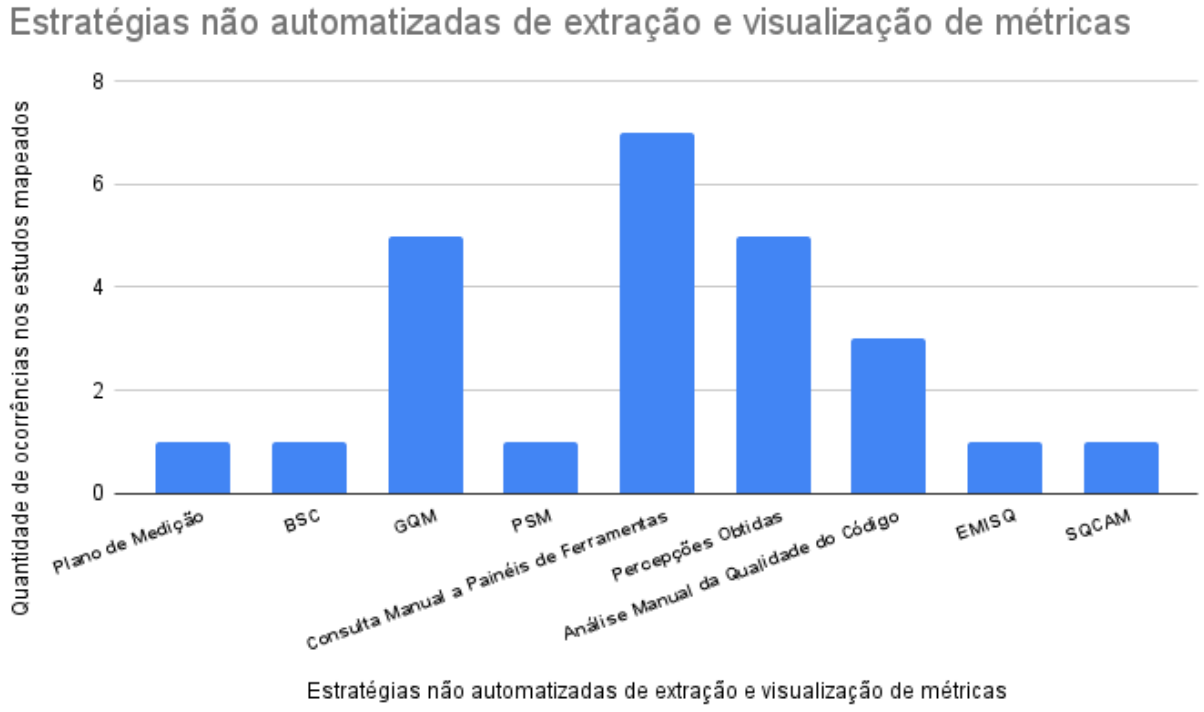
softwares de gestão de qualidade e produtividade em um projeto ágil, é importante realçar o modelo de negócio da plataforma *Q-Rapids*.

Relembrando que, como já mencionado, o *Q-Rapids* possui um módulo *Data Gathering* que é composto por diferentes Conectores *Apache Kafka* e que possuem por foco a coleta de dados heterogêneos provenientes de fontes externas. Dentre tais fontes podemos citar as seguintes: ferramentas focadas em análise de código estático (por exemplo, *Sonar-Qube*), ferramentas de integração contínua (por exemplo, *Jenkins*), repositórios de código (por exemplo, *SVN*, *Git*, *GitLab*), ferramentas de rastreamento de problemas (por exemplo, *Redmine*, *GitLab*, *JIRA*, *Mantis*) e uso de *logs* como principal fonte de dados (MARTÍNEZ-FERNÁNDEZ et al., 2019).

3.2.2.3 QPE1.3: Quais estratégias não automatizadas de extração e visualização das métricas de qualidade e produtividade, são aplicadas durante a execução das atividades em projetos ágeis de software?

Sabe-se que a automação dos processos de obtenção e disponibilização das métricas torna mais viável o controle de um dado processo monitorado. Entretanto, existem uma série de estratégias que pregam a obtenção destes valores manualmente e que podem ser melhor visualizadas na Figura 11.

Figura 11 – Quantitativo de estudos que fazem referência a uma determinada estratégia, não automatizada, de extração e visualização de métricas de qualidade e produtividade.



Fonte: Produzido pelo autor.

É possível observar, que a análise dos estudos resultou na contabilização de 9 diferentes estratégias não automatizadas com foco na extração e visualização de métricas de produtividade e qualidade em projetos ágeis de *software*. Dentre tais estratégias, a que mais apresentou destaque foi a Consulta Manual a Painéis das Ferramentas, com um total de 7 menções. Inclusive o artigo dos autores Liechti et al. (2017), deixa claro a utilização de tal estratégia quando expõe que ferramentas como o *SonarQube* e o *Jira* são dotados de painéis de interface que possibilitam a visualização direta de determinados indicadores e consequentemente também permitem uma melhor interpretação dos dados.

Outras duas estratégias, que também aparecem com um certo destaque são o GQM (*Goal-Question-Metric*) e a estratégia baseada em Percepções Obtidas, ambas com 5 menções cada. Sendo assim, é interessante ressaltar que tais estratégias possuem aplicações distintas e variadas. Enquanto os autores Liechti et al. (2017) e Oliveira et al. (2016) pregam que percepções sobre progressos e qualidade podem ser levantados por impressões (muitas vezes obtidas através de reuniões) e não por análises de dados propriamente ditas, outras estratégias como o GQM implementam um modelo conceitual mais complexo visando a medição de um determinado conjunto de questões e um conjunto de regras para a interpretação dos dados de tal medição.

Desta forma o GQM, abordado no artigo dos autores Bukhari et al. (2018), fornece uma base estrutural conceitual que visa a avaliação de métricas de *software* e a elaboração de um modelo de seleção. Esta estrutura informa aos profissionais as mais valiosas métricas e objetivos das organizações, para que desta maneira seja possível ajudá-los a tomar decisões precisas e provenientes de tais métricas. O GQM opera no conceito de vincular as necessidades de informações organizacionais, estratégias e objetivos dos tomadores de decisões para uma medição precisa (BUKHARI et al., 2018).

Vale ressaltar que, com tudo que foi abordado neste tópico, existe uma dificuldade para se metrificar manualmente a qualidade e a produtividade de sistemas e processos, principalmente à medida que os mesmos se tornam cada vez mais complexos. Booch (1994, *apud* DIAS, 1997) em sua obra já enfatizava que o ser humano possui uma limitação para lidar com altas complexidades de *software*: A complexidade dos sistemas de *software* que desenvolvemos só faz crescer, e ainda temos limitações básicas nas nossas habilidades humanas para lidar com tamanhas complexidades.

3.2.2.4 QPE1.4: Quais os impactos (positivos/negativos) são observados na gestão e na tomada de decisão, a partir do uso das plataformas de extração e visualização das métricas de qualidade e produtividade?

A análise desta questão de pesquisa específica, possibilitou a identificação de diversos impactos positivos no que diz respeito ao uso das plataformas de extração e visualização das métricas de produtividade e qualidade, conforme pode ser observado na Figura 12 que representa uma nuvem de palavras correspondentes aos impactos positivos mais mencionados. O aumento na interoperabilidade, escalabilidade, manutenção do processo de medição, diminuição da dependência

de especialistas, ressalto da qualidade, ressalto do custo e realce das entregas realizadas, estão dentre os principais fatores positivos mencionados que permitem a identificação de como estes tópicos evoluem, convergem e são previsíveis.

Figura 12 – Nuvem de palavras correspondentes aos impactos positivos mais mencionados.

Fonte: Produzido pelo autor.

Liechti et al. (2017) conduziram uma pesquisa na qual buscou-se argumentar que a aceitação de um processo de melhoria contínua orientado por dados sofre influência dos canais de divulgação utilizados para publicação de métricas. Sendo assim, o estudo aponta que as plataformas baseadas em painéis analíticos ajudam as pessoas a tomarem decisões baseadas em dados, visando melhorar seus processos ou produtos. Neste contexto, um *dashboard* pode ser aplicado a fim de apresentar métricas de *software* e consequentemente ajudar os profissionais a tomarem decisões de melhoria contínua, possibilitando uma melhor percepção de progresso a qual detém forte impacto na motivação pessoal de cada colaborador e bem como na equipe em um contexto geral.

Kozik et al. (2018) apresentam um estudo focado na análise de dados e que visam ajudar as empresas a avaliarem a qualidade do *software*. Para alcançar tamanha aplicabilidade de pesquisa, parte-se da premissa que plataformas de análise e visualização de métricas auxiliam a atender os requisitos de qualidade, planejar melhor tarefas e problemas, atribuir melhor tarefas a desenvolvedores específicos e, finalmente, atender aos indicadores estratégicos do produto e organização. Além dos pontos positivos já mencionados, o estudo ainda salienta que a utilização de tais sistemas avançados de métricas permitem um melhor entendimento para que seja possível tomar decisões com mais conhecimento e consciência situacional, mostrando também as correlações ocultas entre vários aspectos dos projetos.

Em contrapartida, não apenas os impactos positivos foram observados durante a análises dos estudos. Pois, também foi possível presenciar uma quantidade significativa de impactos negativos relatados, no que diz respeito ao uso das plataformas de extração e visualização das métricas de produtividade e qualidade, conforme pode ser visto na Figura 13 que representa uma nuvem de palavras correspondentes aos impactos negativos mais mencionados.

Figura 13 – Nuvem de palavras correspondentes aos impactos negativos mais mencionados.

Fonte: Produzido pelo autor.

A utilização de plataformas de extração e visualização das métricas complexas e caras, a dificuldade em se gerenciar enormes quantidades de dados, alta dependência de desenvolvedores e o tempo gasto para selecionar as medidas que de fato entreguem valor, estão entre os principais fatores negativos mencionados. Segundo Ktata e Lévesque (2010), tais impactos negativos mencionados podem criar um ambiente de trabalho hostil. Assim, qualquer medida introduzida pode ser vista principalmente como um malefício e não como uma economia.

Ainda segundo Ktata e Lévesque (2010), medições mal projetadas tendem a causar péssimos efeitos colaterais em um ambiente ágil de desenvolvimento de *software*. Visto que, tal conjectura cria um cenário de resistência à mudança, associado ao medo de ter-se aumentos na sobrecarga de atividades de apoio à coleta de dados. Levando as equipes a dependerem exclusivamente de métricas específicas para monitorar progresso e usarem a intuição como uma ferramenta de decisão primária.

Meding (2017) apresentou um estudo de caso, no qual implantou um *dashboard* visando aferir um conjunto de medidas de *software* que deveriam ser analisadas pelas equipes ágeis. Durante o desenvolvimento da pesquisa, o autor observou diversos impactos negativos que necessitavam de atenção, como por exemplo: falta de padronização nas medidas apresentadas via *dashboard*, pouca flexibilidade para se alterar marcos de tempo definidos, frequência com a qual os dados são

atualizados e tempo para desenvolver e implementar um painel. Desta forma, ficou evidente que monitorar o processo de desenvolvimento de *software* não é uma tarefa trivial, pois envolve uma série de parâmetros que necessitam ser levados em consideração e que possibilitam que as ações de resposta a problemas, típicas de gestão, não sejam baseadas em intuições.

3.3. Considerações Sobre a RSL

Durante o desenvolvimento deste estudo, foi possível observar na literatura uma demasiada carência de pesquisas que abordassem a utilização de plataformas para lidar com as principais métricas de um projeto ágil de desenvolvimento de *software*. Tanto é, que tal afirmação pode ser comprovada baseando-se nos números de artigos obtidos em detrimento daqueles realmente utilizados nas análises das questões de pesquisa. Desta forma, dos 4660 artigos obtidos como resultado da execução das buscas nos engenhos, apenas 44 foram efetivamente utilizados (aproximadamente 0,9%).

Vale ressaltar que, buscando mitigar a dificuldade em encontrar-se os artigos mais relevantes para o tema, foram utilizados os engenhos de busca mais relevantes para a área da Engenharia de *Software*, levando em consideração as recomendações citadas por Medeiros et al. (2015) e Dybå e Dingsøyr (2008).

Os pesquisadores preocuparam-se em manter um cronograma de atividades, bem como em realizar um acompanhamento constante da evolução do progresso individual e global. Para tanto, foram estipuladas reuniões semanais, nas quais as estratégias para seleção dos estudos eram repassadas, além dos *status* de cada integrante. E em caso de divergências de opiniões entre os pesquisadores, um determinado tempo da reunião era destinado para mediação do conflito, tendo sempre a participação e o aval do professor orientador.

Por fim, é importante acrescentar que esta RSL não considerou artigos publicados nos anos de 2021 em diante, pois neste período a pesquisa já estava em curso. Além disso, aproximadamente 1% dos artigos selecionados não puderam ser analisados, justamente por não estarem disponíveis gratuitamente para *download* nos ambientes institucionais do IFPB. Sendo plausível cogitar que nesta amostra existisse algum artigo de relevância para esta pesquisa, e que infelizmente não pode ser incluído pela limitação aqui já justificada.

4. ASSERT METRICS PANEL: UMA PLATAFORMA PARA MONITORAMENTO DE MÉTRICAS

Este capítulo inicialmente descreve como foi feito o mapeamento das métricas de produtividade e qualidade para projetos ágeis de software. Em seguida, descreve a plataforma desenvolvida, aprofundando-se na descrição arquitetural e nas estratégias empregadas para a implementação da solução de *software* que detém o foco de integrar as informações provenientes das ferramentas *Open Project* e *TestLink*.

4.1. Mapeamento das Métricas de Produtividade e Qualidade

As métricas selecionadas para constituir a plataforma desenvolvida nesta pesquisa tiveram por objetivo auxiliar nas ações de estimativa do projeto, no monitoramento da qualidade, na avaliação da produtividade e na concessão de um maior controle a nível gerencial do projeto analisado.

Desta forma, tais métricas aplicadas objetivaram atender as recomendações propostas por Sato (2007) e Cohn (2006), recomendações estas que são: endossar os princípios ágeis; olhar para tendências e não apenas para os números; tentar englobar toda a equipe envolvida no projeto analisado; pertencer a um conjunto pequeno de métricas e diagnósticos; ser facilmente coletada; conceder *feedbacks*; promover a melhoria do processo e ser principalmente compreensíveis fazendo sentido para o contexto aplicado.

Sendo assim, tendo como norte de orientação os princípios para seleção de métricas anteriormente apresentados, realizamos uma reunião para com gestores do Laboratório Assert com o objetivo de compreender melhor como métricas de qualidade e produtividades estavam sendo utilizadas pela instituição, as áreas mais importantes para monitoramento e os processos de medição já realizados no dia a dia dos projetos.

Como resultado do referido encontro e de uma observação atenta ao processo de desenvolvimento ágil empregado no cenário aferido, observou-se que o Laboratório já utilizava em seus projetos uma gama de métricas de produtividade e qualidade advindas de diferentes repositórios de dados. Tais métricas, analisadas exclusivamente no contexto do Laboratório Assert em um primeiro momento, foram catalogadas para melhor entendimento do ambiente de experimentação e podem ser analisadas conforme apresentadas na Tabela 6.

Tabela 6 – Métricas inicialmente utilizadas no Laboratório Assert.

Nome da Métrica	Tipo da Métrica	Ferramenta que Fornecerá os Dados para o Cálculo da Referida Métrica	
		Open Project	TestLink
Quantidade Total de *Bugs* Registrados no Projeto	Qualidade	X	
Quantidade Total de *Bugs* Registrados na *Sprint*	Qualidade	X	
Quantidade Total de Melhorias Registradas no Projeto	Qualidade	X	
Quantidade Total de Melhorias Registradas na *Sprint*	Qualidade	X	
Quantidade Total de *Bugs* Registrados na Semana	Qualidade	X	
Quantidade Total de Melhorias Registradas na Semana	Qualidade	X	
Quantidade de Casos de Teste Executados no Projeto	Qualidade e Produtividade		X
Quantidade de Planos de Teste Criados para Atender o Projeto	Qualidade e Produtividade		X
Quantidade de Casos de Teste Executados em um Plano de Testes	Qualidade e Produtividade		X
Número de Testes Aprovados em Todos os Planos de Teste	Qualidade		X
Número de Testes Aprovados em um Plano de Teste	Qualidade		X
Número de Testes Reprovados em Todos os Plano de Teste	Qualidade		X
Número de Testes Reprovados em um Plano de Teste	Qualidade		X
Número de Testes Bloqueados em Todos os Plano de Teste	Qualidade		X
Número de Testes Bloqueados em um Plano de Teste	Qualidade		X
Número de Testes Não Executados em um Plano de Teste	Qualidade		X
Média de Tempo Gasto Executando Testes	Produtividade		X
Quantidade Total de *Bugs* em *Status* de Aberto	Qualidade e Produtividade	X	
Quantidade Total de *Bugs* em *Status* de Resolvido	Qualidade e Produtividade	X	
Quantidade Total de Melhorias em *Status* de Aberto	Qualidade e Produtividade	X	
Quantidade Total de Melhorias em *Status* de Resolvido	Qualidade e Produtividade	X	
Quantidade Total de *Bugs* Registrados para Cada Área do Sistema (*Feature*)	Qualidade e Produtividade	X	
Quantidade Total de *Bugs* em *Status* de Aberto para Cada Área do Sistema (*Feature*)	Qualidade e Produtividade	X	

Quantidade de *Bugs* em *Status* de Aberto e Organizados por Severidade	Qualidade	X	
Quantidade de *Bugs* em *Status* de Aberto e Organizados por Áreas do Sistema (*Feature*)	Qualidade	X	
Quantidade Total de *Bugs* em *Status* de Resolvido para Cada Área do Sistema (*Feature*)	Qualidade e Produtividade	X	
Quantidade de *Bugs* em *Status* de Resolvido e Organizados por Severidade	Qualidade	X	
Quantidade de *Bugs* em *Status* de Resolvido e Organizados por Áreas do Sistema (*Feature*)	Qualidade	X	
Quantidade de Melhorias em *Status* de Aberto e Organizadas por Áreas do Sistema (*Feature*)	Qualidade	X	
Quantidade de Melhorias em *Status* de Resolvido e Organizadas por Áreas do Sistema (*Feature*)	Qualidade	X	
Número de Casos de Teste Implementados para Cada Áreas do Sistema (*Feature*)	Qualidade e Produtividade		X
Nome dos Colaboradores que mais Registraram *Bugs*	Produtividade	X	
Nome dos Colaboradores que mais Registraram Melhorias	Produtividade	X	
Nome dos Colaboradores que mais Resolveram *Bugs*	Produtividade	X	
Nome dos Colaboradores que mais Resolveram Melhorias	Produtividade	X	

Fonte: Produzido pelo autor.

Importante frisar que a classificação do tipo de cada métrica, apontada na tabela acima, entre sendo um indicador de qualidade, produtividade ou qualidade e produtividade, foi uma ação de categorização realizada conjuntamente com os Gestores de Projeto do Laboratório Assert e que buscou respeitar os princípios propostos por Guarizzo (2008) e Preesman (2016), no que diz respeito a correta definição entre métricas. Desta forma, foram identificadas como métricas de produtividade aquelas que se concentravam na saída do processo da engenharia de *software* (por exemplo: número de iterações, número de estórias do usuário), métricas de qualidade aquelas que ofereciam uma indicação de quanto o *software* se adequava às exigências implícitas e explícitas do cliente (por exemplo: erros, fases) e métricas de qualidade e produtividade aquelas que de alguma forma mesclavam estes dois conceitos apresentados.

Além de seguir os princípios permeados pela literatura, também buscou-se respeitar a classificação das métricas já em uso no Laboratório Assert e que já se encontram amplamente difundidas entre os seus colaboradores e clientes, através principalmente da divulgação dos relatórios de *sprints* desenvolvidos quinzenalmente. Reforçando assim, a necessidade do laboratório em manter a sua forma de tipificar cada métrica utilizada, pois também já é um padrão de tipificação de métricas conhecido e respeitado pelos clientes com os quais se realiza projetos de cooperação.

Afora o modelo de tipificação de métricas empregado na instituição analisada, outro fator que chama bastante atenção é o uso frequente de métricas que possuem associação direta a *Fetures* do Sistema. Segundo Turner et al. (1999), uma *Feature* pode ser conhecida como um agrupamento de requisitos individuais dentro de um tema específico, enfatizando sua origem dentro do domínio do problema. Ou seja, a *Feature* pode ser compreendida como uma característica visível para um *stakeholder*, permitindo assim a expressão de partes comuns e variáveis entre as instâncias, além de representar requisitos reusáveis e configuráveis.

Posto isto, a partir dos dados coletados sobre as métricas utilizadas no Laboratório Assert, é então possível afirmar que o centro de pesquisa possui uma cultura consolidada no uso de tais medições. No entanto a problemática está relacionada ao fato de tais informações sempre necessitarem ser consolidadas manualmente através de consultas e contagens diretas nas ferramentas *OpenProject* e *TestLink*, visto que para ter-se um relatório consistente é necessário obter os dados registrados em ambas as ferramentas e que por ventura não se comunicam, inviabilizando uma coleta automatizada dos dados de uma forma mais simplificada.

Tomando assim por base o conhecimento obtido sobre o contexto de aplicação da pesquisa, em especial no que diz respeito a abordagem de métricas de produtividade e qualidade utilizadas nos projetos de desenvolvimento ágil de *software*, foi possível notar que a seleção das métricas a serem empregadas nesta pesquisa também poderia dar-se por similaridade entre aquelas que já estavam em uso no contexto de aplicação *versus* aquelas métricas propostas na literatura. Pois, durante a realização da RSL foi possível observar que alguns autores chegaram a sugerir um conjunto de métricas ideais ao contexto ágil e totalmente focados na medição de produtividade e qualidade. Como foi o caso de Meding (2007), que sugeriu algumas métricas similares às já em uso no Laboratório Assert, dentre as quais pode-se citar: quantidade de defeitos abertos para o sistema, número de defeitos relatados por cada integrante do time, defeitos na *sprint* e *burn-down /burn-up*.

Desta forma, baseando-se nos princípios permeados pela utilização das métricas ágeis, pelas boas práticas observadas em trabalhos levantados via RSL e tendo a disposição o cenário atual das métricas utilizadas no Laboratório Assert, realizou-se mais um alinhamento entre pesquisadores e gestores no qual foi definido um conjunto de métricas viáveis a serem coletados, levando em conta o fator de dificuldade para obtenção de tais informações e o grau de valor das principais métricas elencadas. Posto isto, é possível analisar na Tabela 7 o entendimento consolidado entre os gestores do Laboratório Assert e os pesquisadores deste projeto, referente às métricas elegíveis para coleta e análise através da plataforma para gestão de métricas em sua primeira versão desenvolvida.

Tabela 7 – Métricas selecionadas para comporem a Plataforma de Métricas.

Nome da Métrica	Tipo da Métrica	Ferramenta que Fornecerá os Dados para o Cálculo da Referida Métrica	
		Open Project	TestLink
Quantidade Total de *Bugs* Registrados no Projeto	Qualidade	X	
Quantidade Total de *Bugs* Registrados na Semana	Qualidade	X	
Quantidade de Casos de Teste Executados no Projeto	Qualidade e Produtividade		X
Quantidade de Planos de Teste Criados para Atender o Projeto	Qualidade e Produtividade		X
Número de Testes Aprovados em Todos os Planos de Teste	Qualidade		X
Número de Testes Reprovados em Todos os Plano de Teste	Qualidade		X
Média de Tempo Gasto Executando Testes	Produtividade		X
Quantidade Total de *Bugs* Registrados para Cada Área do Sistema (*Feature*)	Qualidade e Produtividade	X	
Quantidade Total de *Bugs* em *Status* de Aberto para Cada Área do Sistema (*Feature*)	Qualidade e Produtividade	X	
Quantidade Total de *Bugs* em *Status* de Resolvido para Cada Área do Sistema (*Feature*)	Qualidade e Produtividade	X	
Número de Casos de Teste Implementados para Cada Áreas do Sistema (*Feature*)	Qualidade e Produtividade		X
Nome dos Colaboradores que mais Registraram *Bugs*	Produtividade	X	
Nome dos Colaboradores que mais Resolveram *Bugs*	Produtividade	X	

Fonte: Produzido pelo autor.

4.2. Arquitetura da Plataforma

Esta pesquisa busca como solução desenvolver uma plataforma de gerenciamento de métricas de *software* para projetos ágeis, utilizando-se de uma solução disruptiva que incorpora os componentes necessários para o funcionamento da proposta.

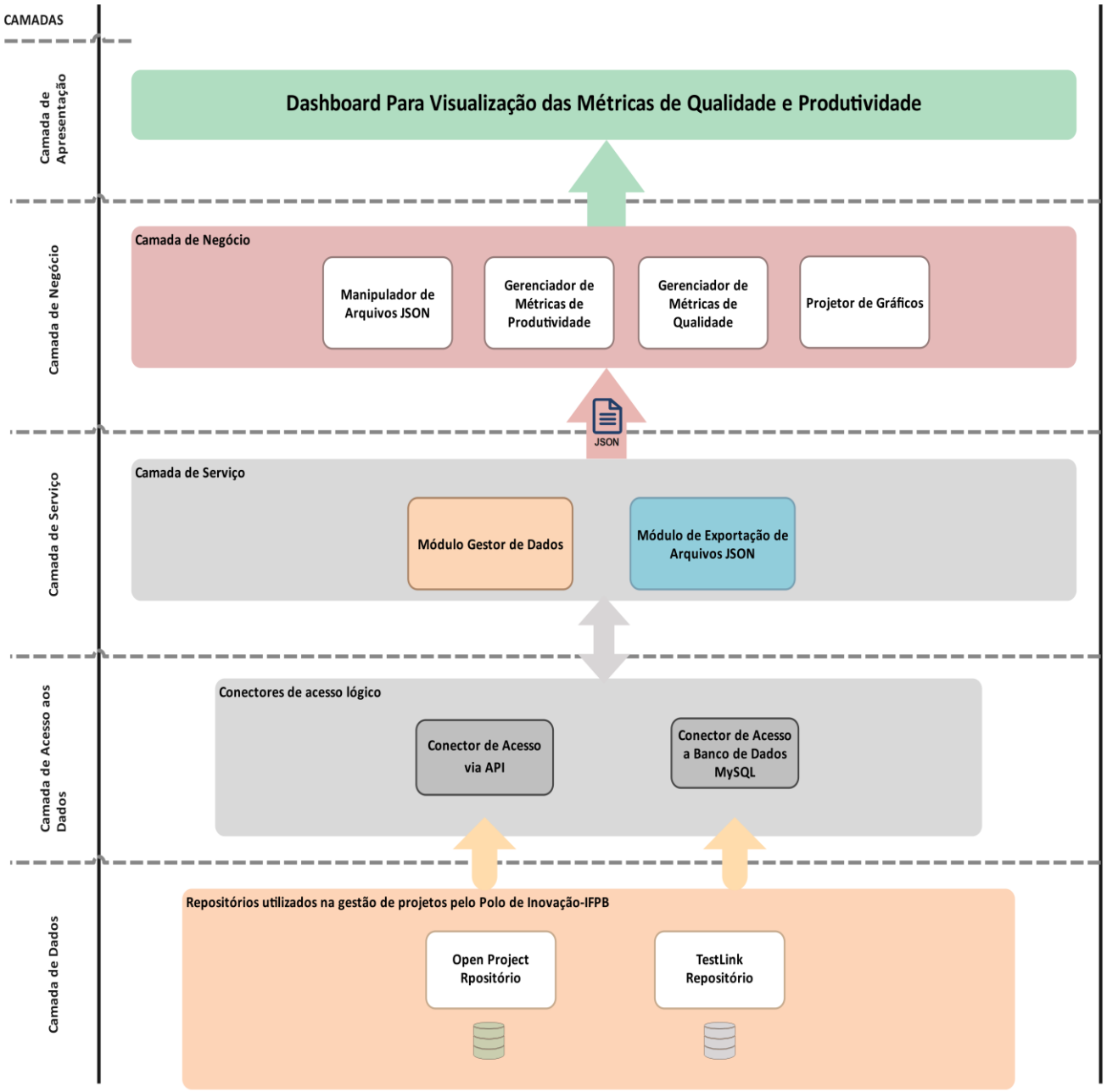
Com o intuito de facilitar a referenciação, deste ponto em diante a Plataforma de Métricas sugerida nesta pesquisa será nomeada de ***Assert Metrics Panel* (AMP).** O **AMP** pode ser entendido como uma solução integrada de *software* que visa disponibilizar métricas de produtividade e qualidade acerca de um dado projeto ágil de desenvolvimento, facilitando desta maneira o acesso às informações provenientes de diferentes fontes e dando uma conotação de padronização aos dados. Sendo assim, no contexto de aplicação do Polo de Inovação – IFPB, a arquitetura da plataforma foi pensada para que fosse possível realizar-se a integração com as ferramentas de gestão em uso pela

instituição (*Open Project* e *TestLink*), bem como para que se pudesse filtrar as informações obtidas manuseando-as e transformando-as nas métricas de produtividade e qualidade desejadas.

A Figura 14 representa a arquitetura proposta para a Plataforma **AMP**. Na referida arquitetura, são exibidos os domínios utilizados na estrutura do sistema desenvolvido. Ressaltando que, baseando-se em Boa Morte (2020), a definição de "domínio" empregada nesta pesquisa tem o significado de "área de competência" ou "blocos de atribuição da responsabilidade". A arquitetura da plataforma foi estruturada em 5 camadas:

- Dados: camada mais baixa da arquitetura, sendo responsável pela comunicação com as bases de armazenamento de dados utilizadas;
- Acesso: fornece acesso simplificado aos dados armazenados em um ou mais bancos de dados, constituindo-se como um elo entre as entidades de negócio e as camadas de armazenamento de dados;
- Serviços: atua como um limite transacional, contendo serviços dedicados à infraestrutura de dados, destacando-se os Módulos de Gestão de Dados e Exportação de Arquivos JSON;
- Negócio: responsável pela classificação dos dados entregues pela camada de serviço;
- Apresentação: camada de interação dos usuários para com a Plataforma.

Figura 14 – Arquitetura proposta para o desenvolvimento da AMP.



Fonte: Produzido pelo autor.

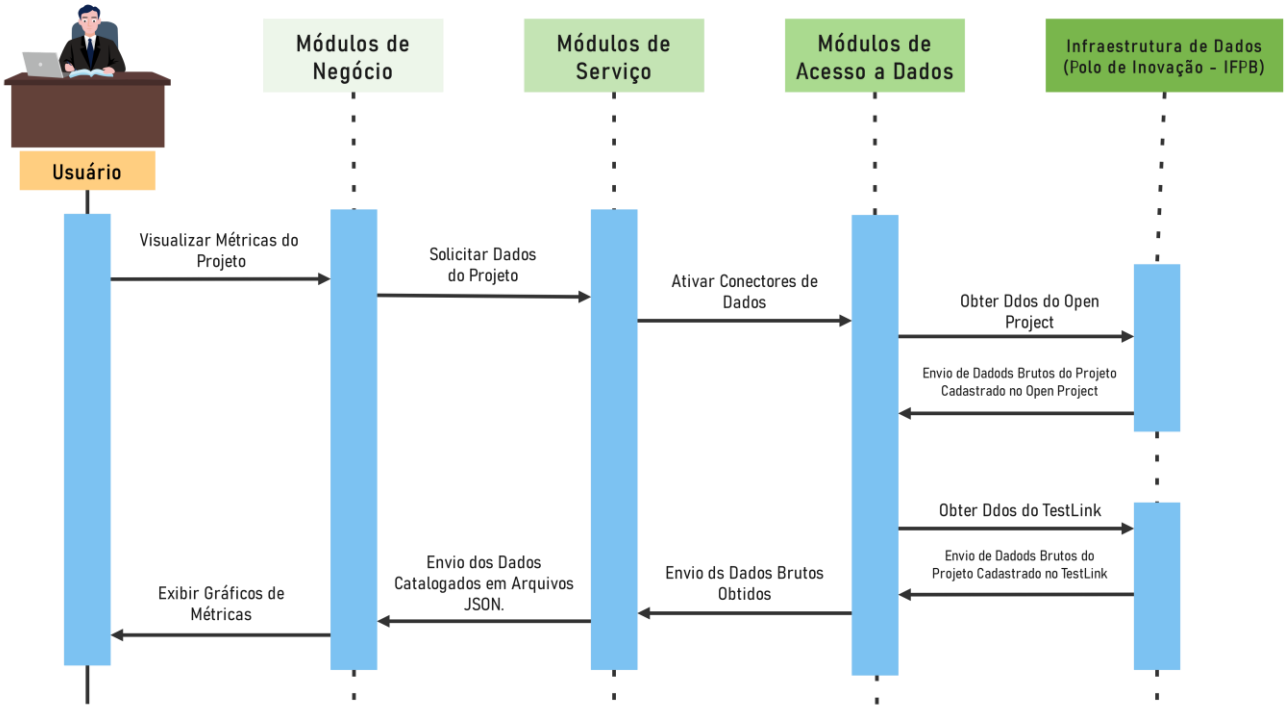
A Figura 15 apresenta o fluxo de atividades envolvendo a utilização da Plataforma **AMP** para monitorar métricas sobre produtividade e qualidade de um projeto que utilize as ferramentas *Open Project* e *TestLink*. Tais atividades explicitadas, são:

- O usuário tem acesso a Plataforma **AMP**, tendo o objetivo de visualizar métricas que retratem a produtividade e a qualidade de um dado projeto;
- A camada de apresentação gera uma requisição para a camada de negócio,

solicitando a construção dos gráficos contendo as métricas de produtividade e qualidade do projeto analisado;

- Para que a camada de negócio consiga realizar a consolidação das métricas ela solicita os dados necessários para camada de serviço, que por sua vez, ativa os conectores de dados responsáveis por buscarem diretamente nas ferramentas *Open Project* e *TestLink*;
- Os conectores de dados estabelecem relacionamentos com as ferramentas *Open Project* e *TestLink*. Esses dados são enviados em um *response* e sem uma filtragem prévia para a camada de serviço;
- A camada de serviços trata os dados recebidos criando arquivos do tipo .JSON, visando auxiliar a criação das métricas, e envia estes arquivos para a camada de negócio;
- A camada de negócio realiza a manipulação dos arquivos. JSON recebidos, extraindo os dados necessários para criação das métricas de produtividade e qualidade. Ao final, envia para a camada de apresentação os gráficos de cada métrica devidamente montados;
- Por fim, o usuário tem acesso ao *dashboard* contendo os gráficos pré-estabelecidos para representação das métricas de qualidade e produtividade em um projeto ágil de desenvolvimento de *software* específico.

Figura 15 – Fluxo de trabalho geral da Plataforma AMP.



Fonte: Produzido pelo autor.

Desta forma, visando apresentar uma descrição detalhada para cada camada de domínio as quais foram incisivamente abordadas nas Figura 14 e 15, são apresentadas as seções abaixo objetivando expandir as especificações de cada área de competência, bem como os seus relacionamentos e peculiaridades no contexto arquitetural no qual estão integrados.

4.2.1 Domínio de Dados e de Acesso

O Domínio de Dados relatado a seguir e especificado na Figura 14, refere-se a camada mais baixa da arquitetura da plataforma, sendo responsável pela comunicação com as bases de armazenamento de dados utilizadas.

É nesta camada que se especifica com quais bases a plataforma irá se comunicar para coleta de dados. Sendo necessário ter acesso aos repositórios utilizados pelas ferramentas *Open Project* e *TestLink*.

Na arquitetura elucubrada, a camada de Domínio de Dados também inclui a camada de acesso aos respectivos dados, que visa fornecer acesso simplificado aos dados contidos nos repositórios desejados e para alcançar tal objetivo usa em determinados momentos de interfaces do tipo API (*Application Programming Interface*). Segundo Oliveira (2018), a API é uma interface implementada via código e que obedece a um conjunto de padrões e rotinas, que é capaz de interligar diferentes tipos de aplicações e sistemas. Dessa forma, uma API possui um ou diversos *endpoints*,

que fornecem um serviço para ser consumido por outras aplicações clientes, tornando-se ideal para o contexto desta pesquisa.

A camada de extensão do Domínio de Dados, conhecida por Domínio de Acesso aos Dados, é constituída por dois tipos de conectores, que são:

- ***Conector de Acesso a Repositórios de Dados via API:*** Tem por foco conectar-se diretamente ao repositório de dados do *software Open Project*, via chamada de API e através de um *endpoint* específico, para que desta maneira se tenha acesso a todos os dados de um determinado projeto cadastrado na ferramenta. Importante ressaltar que, o *endpoint* utilizado neste projeto foi o de *Projects*, buscando-se pela propriedade *workPackages* e como pré-requisito para acesso aos dados faz-se necessário ter o Id do Projeto desejado e um *Token* de acesso à API.
- ***Conector de Acesso a Repositórios de Dados MySQL:*** Tem por foco conectar-se diretamente ao repositório de dados do *software TestLink*, estabelecendo-se uma mysql *connection*, com auxílio da tecnologia de desenvolvimento empregada na implementação da plataforma e realizando consultas SQL nas tabelas desejadas. Importante ressaltar que, o foco deste conector foi relacionar-se com a tabela *nodes_hierarchy* detentora das informações necessárias, e como pré-requisito para ter-se acesso aos dados se faz necessário uma autorização para o estabelecimento de conexão no banco de dados.

Importante realçar, que a utilização de dois diferentes tipos de conectores visa facilitar a modularidade e diminuir o acoplamento dos conectores.

4.2.2 Domínio de Serviço

O Domínio de Serviços atua como um limite transacional, contendo serviços dedicados à infraestrutura de dados. Neste referido domínio destacam-se os Módulos de Gestão de Dados e Exportação de Arquivos JSON.

O módulo de Gestão de Dados visa manter a referência aos conectores dos repositórios controlando os seus acionamentos, bem como fornecer um direcionamento sobre quais dados fazem-se necessários para a obtenção. Atua, em outras palavras, como uma interface consistente para acesso aos dados, possibilitando a aquisição, transformação das informações e dando subsídios para que o módulo de e Exportação de Arquivos JSON consolide o conjunto de dados em arquivos específicos que são tidos como artefatos de saída da camada de Serviço e consequentemente servirão como artefatos de entrada para a camada de Negócio, estabelecendo assim um elo entre domínios.

4.2.3 Domínio de Negócio

O Domínio de Negócio inclui os principais módulos de transações da Plataforma **AMP**, pois observa-se que as operações de catalogação dos dados em métricas e suas respectivas visualizações são realizadas neste domínio específico.

Nesta camada arquitetural, é realizada a classificação dos dados entregues pela camada de serviço e disponibilizados em arquivos do tipo JSON. A correta leitura e tratamento de tais arquivos é de responsabilidade do Manipulador de Arquivos JSON, que apenas interpreta a informação sem alterá-la.

A classificação dos dados obtidos, expandindo-os em métricas pré-estabelecidas é realizada por dois recursos do sistema, que são:

- ***Gerenciador de Métricas de Produtividade:*** Atua na obtenção de métricas de produtividade a partir dos dados obtidos da ferramenta *Open Project* sobre os pacotes de trabalho de um determinado projeto específico. Visando assim, quantificar as informações obtidas salvando-as em constantes que podem ser facilmente manuseadas.
- ***Gerenciador de Métricas de Qualidade:*** Atua na obtenção de métricas de qualidade a partir dos dados obtidos na ferramenta *Open Project* e na ferramenta *TestLink*. Dados referentes à qualidade do projeto encontram-se registrados em ambas as bases de informação. O principal objetivo deste recurso também é quantificar as informações obtidas salvando-as em constantes que podem ser facilmente manuseadas.

Desta forma, assume-se que tais operações desempenhadas pelos módulos arquiteturais de obtenção de métricas, fornecerão subsídios para a construção de gráficos adequados que darão maior visibilidade à informação consolidada. Consequentemente, o módulo Projetor de Gráficos consolidará as constantes obtidas e as manipularão de forma a exibir métricas diagramadas.

4.2.4 Domínio de Apresentação

Este domínio representa, de forma sucinta, a camada direta de interação dos usuários com a Plataforma **AMP** (***Assert Metrics Panel*)**. Neste domínio o que está em destaque é a exposição dos dados em formato de métricas num modelo de visualização contendo *dashboards*, o que possibilita o acesso à informação de maneira prática e de forma externalizada (externa aos bancos de dados pertencentes).

Sendo assim, a visualização das métricas é o processo de criar representações visuais dos dados analisados. Tal processo pode ser usado para apresentar informações às partes interessadas, para comunicar-se com o público interno e externo e para expandir a análise dos conjuntos de dados. Ou seja, a Plataforma **AMP** possibilita que as métricas da produtividade e qualidade de um dado

projeto sejam calculadas com uma certa periodicidade, habilitando a construção de uma análise histórica, bem como a construção de estimativas de pequeno, médio e longo prazo. Tudo isso proporciona uma avaliação objetiva da situação, promovendo decisões baseadas no resultado dos indicadores, considerando tendências e comparações.

Posto isto, é possível afirmar que o domínio de apresentação da Plataforma de Gestão de Métricas é a camada responsável por transformar dados brutos sobre registros de *bugs*, andamento do *backlog* ou cobertura dos testes de *software*, em uma representação visual gráfica que pode ser facilmente compreendida pelo olho humano.

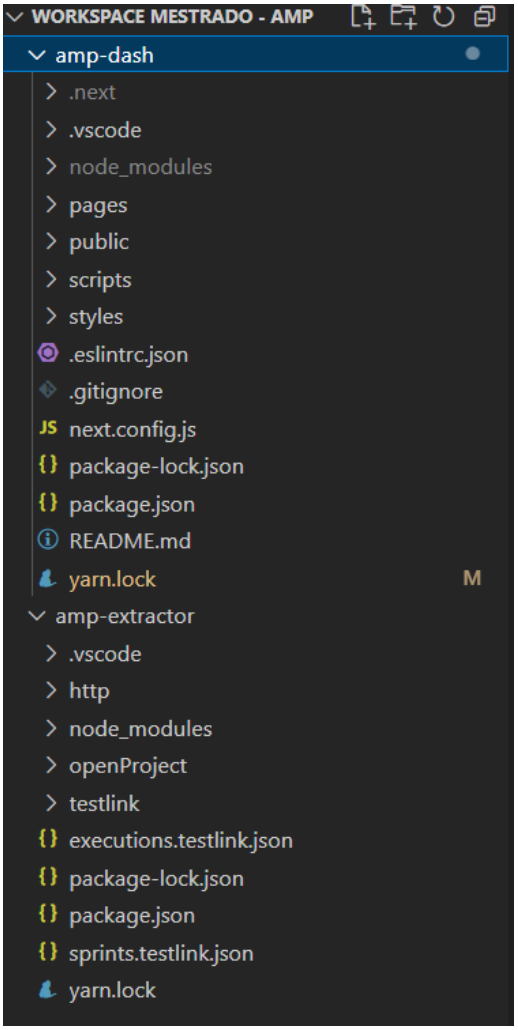
4.3. Implementação da Plataforma

Como forma de validar a proposta de solução arquitetural, foi desenvolvido um arcabouço de implementação para a plataforma. Primeiramente, é importante explicitar que para alcançar o propósito estabelecido com a implementação da Plataforma **AMP**, fez-se necessário definir um conjunto de tecnologias a serem empregadas nas atividades de codificação e suporte. Sendo assim, optou-se pelo uso das seguintes tecnologias para o desenvolvimento da plataforma: JavaScript, Node.js 14.19.1, ReactJS 17.0.2, Next.JS 12.1.11, Core.UI 4.1.3 e JSON. Já no que tange o uso das ferramentas de apoio à codificação, optou-se pela utilização dos seguintes artefatos tecnológicos de suporte: Visual Studio Code, GitHub e Netlify.

Posto isto, o sistema foi desenvolvido e disponibilizado em uma versão *web* responsiva conforme pode ser melhor vislumbrado e acessado através do seguinte endereço eletrônico: https://630a7621fbefa937f7c71069--bright-strudel-8e70ad.netlify.app/. A plataforma adequa-se muito bem a telas de alta resolução, permitindo que seja utilizada em monitores de acompanhamento posicionados junto a salas dedicadas a gestão e monitoramento de projetos.

Seguindo a lógica arquitetural e a proposta para fluxo de trabalho, as quais são exibidas respectivamente nas Figuras 14 e 15, buscou-se então estabelecer uma conexão direta e assíncrona para com os repositórios de dados das ferramentas *Open Project* e *TestLink*. Para tanto, optou-se em separar o *workspace* de codificação em dois projetos interconectados conforme pode-se observar na Figura 16, o *amp-extractor* (foco na obtenção de dados, atendendo assim as especificações das camadas de dados e acesso) o *amp-dash* (foco na análise e exposição dos dados, atendendo assim as especificações das camadas de serviço, negócio e apresentação), cada qual com um objetivo específico.

Figura 16 – *Workspace* de codificação contendo os projetos interconectados *amp-extractor* e *amp-dash*.



Fonte: Produzido pelo autor.

É importante frisar que as seções deste capítulo que especificam detalhadamente como se deram as implementações das camadas arquiteturais da Plataforma **AMP**, ficam restritas apenas aos detalhamentos técnicos da codificação. Os resultados gráficos obtidos, a partir dos códigos desenvolvidos, serão detalhados e expostos a longo de todo o capítulo 5.

4.3.1 Implementação das Camadas de Dados e Acesso

O projeto *amp-extractor* é a representação das especificações das camadas de dados e acesso, sendo o responsável por manter as implementações referentes às conexões com as bases de dados desejadas. Também, neste projeto mantém-se os códigos para as manipulações dos dados indispensáveis à criação das métricas pré-estabelecidas, fazendo com que tais informações sejam consolidadas em arquivos .JSON. Arquivos estes que possuem papéis indispensáveis do ponto de vista arquitetural, pois são compartilhados juntamente ao projeto *amp-dash* para que o mesmo possa realizar suas devidas funcionalidades.

Vale salientar que, não foram utilizados os mesmos métodos de conexão para obtenção dos dados nas ferramentas *Open Project* e *TestLink*. E isto deu-se pela necessidade de implementação encontrada e pela limitação tecnológica observada. Por exemplo, para obtenção das informações cadastradas no *Open Project* referentes aos pacotes de trabalho de um dado projeto, foi estabelecida uma conexão direta via API (*Application Programming Interface*), conectando-se diretamente ao *endpoint Projects*, buscando-se pela propriedade *workPackages* e informando na própria requisição o Id do Projeto desejado e um *Token* de acesso à API, para que assim fosse possível ter-se acesso aos dados.

Todos os dados retornados pela API do *Open Project* foram então transformados e salvos diretamente em um arquivo .JSON, conforme pode ser visto no trecho de código exposto na Figura 17.

Figura 17 – Código utilizado para estabelecer conexão via API com a ferramenta *Open Project*.

```
  const data = await
axios.get('http://openproject.assert.ifpb.edu.br/api/v3/projects/XX/work_pack
ages?filters=XX', { headers: { 'Content-Type': 'application/json',
Authorization: 'Basic ' + Buffer.from('apikey:XX ').toString('base64') } })
```

Fonte: Produzido pelo autor.

Ressalta-se que a requisição de acesso apresentada, foi desenvolvida seguindo as orientações da documentação da própria API do *Open Project*. Ao longo deste documento, algumas informações contidas no código foram omitidas pela representação **XX**, tendo por foco primordial respeitar as boas práticas de privacidade e segurança do projeto utilizado durante o desenvolvimento da plataforma.

No que diz respeito à integração entre a Plataforma **AMP** e a ferramenta *TestLink*, os pesquisadores optaram por descartar o uso da API como método de conexão. Pois, ao realizar alguns testes não se obteve êxito quanto ao uso da API nativa disponibilizada, bem como não foram encontradas soluções plausíveis na literatura ou na própria documentação da ferramenta que propusessem uma estratégia de conexão entre o *TestLink* e uma aplicação desenvolvida em *Node.js*.

Sendo assim, a extração dos dados proveniente da ferramenta *TestLink*, foi realizada através de uma conexão direta ao seu banco de dados MySQL. Conforme pode ser observado na Figura 18, na qual é exibido o trecho de código responsável por se estabelecer uma conexão assíncrona mediante as credenciais informadas.

Figura 18 – Código utilizado para estabelecer-se uma conexão MySQL junto ao banco de dados da ferramenta *TestLink*.

```
import mysql from 'mysql';
import fs from 'fs'

var connection = mysql.createConnection({
    host: 'testlink.XX.ifpb.edu.br',
    user: 'XX',
    password: 'XX',
    database: 'XX’
});
```

Fonte: Produzido pelo autor.

A partir da conexão estabelecida é possível a obtenção dos dados desejados, realizando-se consultas *SQL* por meio da função *query* do *Node.js*, que porventura usará o objeto *connection* criado e para ter-se acesso ao repositório solicitado.

4.3.2 Implementação das Camadas de Serviço, Negócio e Apresentação

A nível de codificação, a arquitetura lógica do código que faz referência a análise das métricas e suas respectivas exibições, se encontra registrada no projeto *amp-dash*, que por sua vez está contido no *workspace* geral do projeto e depende do projeto *amp-extractor* para que o mesmo lhe forneça os arquivos .JSON contendo os dados necessários ao processamento e obtenção das métricas.

Tais métricas, a depender se sua complexidade, podem ser exibidas ao usuário de duas maneiras específicas, que são: *Widgets* Numéricos e *Widgets* Gráficos.

Widgets numéricos, são definidos para este trabalho de pesquisa como sendo as áreas de interação da interface gráfica que exibem as métricas de produtividade e qualidade do projeto em simples cards HTML.

O termo *Widgets* Gráficos, é definido neste este trabalho de pesquisa como sendo áreas de interação da interface gráfica que exibem as métricas de produtividade e qualidade do projeto por meio de gráficos de colunas responsivos e dinâmicos. Tais gráficos, detêm o objetivo de melhor demonstrar a composição dos dados organizando-os em períodos específicos, que facilitem a visualização dos conjuntos de informações em sua totalidade ou em partes delimitadas.

Visando melhorar a compreensão, as subseções 4.3.2.1 e 4.3.2.2 irão explanar melhor os modelos de implementações utilizados em cada um dos *Widgets* mencionados.

4.3.2.1 Implementação dos Widgets Numéricos

As métricas disponibilizadas através do *widgets* numéricos apresentam uma variação no que diz respeito a origem dos dados que as alimentam. Por exemplo, apenas as métricas Quantidade de *Bugs* Registrados no Projeto e *Bugs* Registrados por Semana, possuem seus dados oriundos do *Open Project*, já as demais métricas apresentadas utilizam dados diretamente coletados do *TestLink*.

Desta forma, as métricas consolidadas a partir de dados oriundos do *Open Project* são construídas realizando-se um filtro sobre o arquivo *data.json* (arquivo obtido na requisição de API feita diretamente à ferramenta e que retorna os dados de um projeto, salientando que parte deste arquivo pode ser analisada no Apêndice C deste trabalho de pesquisa). Este filtro buscará por uma *Tag* específica dentro do arquivo .JSON e a manipulará conforme seja a necessidade. Tudo isso pode ser melhor compreendido analisando a Figura 19, que apresenta a codificação de obtenção da quantidade total de *bugs* registrados em um projeto a partir do arquivo *data.json* obtido.

Figura 19 – Código utilizado para contagem da quantidade total de *bugs* registrados no projeto.

```
const data = require('../public/data.json')

// count bugs
const { _embedded: { elements: bugs } } = data
console.log(bugs.length)
```

Fonte: Produzido pelo autor.

Como pode ser observado na imagem acima, o resultado pela busca da quantidade total de *bugs* registrados em um projeto e que porventura são dados advindos da ferramenta *Open Project*, acaba gerando uma constante dentro do código que é utilizada diretamente no *front-end* da aplicação, objetivando a plena exibição do conteúdo obtido conforme detalhado na Figura 20.

Figura 20 – Código utilizado para exibição da quantidade total de *bugs* registrados no projeto.

```
<CCol xs={4}>
          <CWidgetStatsE
            className="mb-4"
            title="Quantidade de Bugs registrados"
            value={<h1>{bugs.length}</h1>}
          />
</CCol>
```

Fonte: Produzido pelo autor.

Já para as métricas Quantidade de Planos de Teste Registrados no Projeto, Total Testes Executados que Foram Aprovados no Projeto, Total Testes Executados que Foram Reprovados no Projeto e Média de Tempo Gasto Executando Testes, faz-se necessário a obtenção de dados junto a ferramenta *TestLink* para que seja possível medir e exibir as informações desejadas.

Neste cenário, a Plataforma **AMP** executa uma *Query* de consulta SQL, via conexão direta estabelecida com o banco de dados MySQL da ferramenta desejada. A consulta criada visa obter os dados necessários para a medição das quatro métricas a serem postadas via *cards* e que são dependentes das informações exclusivamente contidas e registradas no *TestLink*.

O resultado da requisição feita ao banco de dados é salvo em um arquivo. JSON nomeado de *executions.testlink.json*. Tal sequência lógica de implementação, pode ser vista no código da aplicação exibido na Figura 21 (parte deste arquivo .JSON mencionado, contendo dados exclusivos da ferramenta *TestLink*, pode ser analisado no Apêndice D deste trabalho).

Figura 21 – Código utilizado na obtenção de dados do *TestLink* para métricas que são exibidas em *widgets* numéricos.

```
function getExecutionsTest() {
    const sql = `SELECT t2.id as "plans", e.status, e.execution_duration, us.first
                from testprojects t
                join testplans t2 on t2.testproject_id = t.id
                join executions e on e.testplan_id = t2.id
                JOIN users_sanitized us on us.id = e.tester_id
                where t.prefix like "%ENERSH%"`;
    connection.query(sql, function (error, results, fields) {
        if (error) console.log(error);
        let reduceData = results.reduce((acc, item, index) => {
            if (!acc['plans'].includes(item.plans))
                acc['plans'].push(item.plans)

            if (!acc['status'][item.status.toLowerCase()])
                acc['status'][item.status.toLowerCase()] = 0

            acc['status'][item.status.toLowerCase()] += 1

            acc['executions'] += item.execution_duration || 0
            acc['executions_qtd'] += 1
            return acc
        },
       { plans: [], status: [], executions: 0, executions_qtd: 0 })
        const resultData = [
            { display: "Quantidade de planos de testes", value:
reduceData.plans.length },
            { display: "Quantidade de execuções que passaram", value:
reduceData.status.p },
            { display: "Quantidade de execuções que não passaram", value:
reduceData.status.f },
            { display: "Media de tempo de execucão em minutos", value:
(reduceData.executions / reduceData.executions_qtd).toFixed(2) }
        ]

        fs.writeFile(`executions.testlink.json`, JSON.stringify(resultData),
function (err) {
            if (err) return console.log(err);
        });
    });
}
```

Fonte: Produzido pelo autor.

O arquivo .JSON obtido como resultado da consulta executada junto ao banco de dados da ferramenta *TestLink*, é organizado de forma a conter o nome da métrica e o valor de sua respectiva medição. Este nível de organização só é possível graças ao uso de consultas SQL, que permitem um maior refinamento na realização da análise e síntese dos dados desejados. Porventura, esta estruturação do arquivo *executions.testlink.json* permite ao *front-end* da aplicação utilizá-lo de forma direta, realizando um laço de repetição que coleta e publica em *cards* distintos o nome das métricas e seus valores associados, assim como exposto no trecho de código mostrada na Figura 22.

Figura 22 – Código utilizado para exibição em *widgets* numéricos das métricas obtidas através de dados analisados do *TestLink*.

```
<>
          {
            testExecutions?.map((test, i) =>
              <CCol
                key={i}
                xs={4}>
                <CWidgetStatsE

                  className="mb-3"
                  title={test.display}
                  value={<h1>{test.value}</h1>}
                />
              </CCol>
            )
          }
 </>
```

Fonte: Produzido pelo autor.

4.3.2.2 Implementação dos Widgets Gráficos

Para melhorar a explanação e consequentemente o entendimento, optou-se neste projeto em dividir os *Widgets* Gráficos em dois subconjuntos, que são: Gráficos de *Rankings* e Gráficos de Dados Distintos Associados.

4.3.2.2.1 GRÁFICOS DE RANKINGS

As métricas mencionadas nesta categoria específica de gráficos, são obtidas a partir da análise exclusiva dos dados oriundos da ferramenta *Open Project* e seguem a mesma lógica arquitetural de codificação dos *Widgets* Numéricos. Estratégia na qual, realiza-se um filtro sobre o arquivo *data.json* (arquivo obtido na requisição de API feita diretamente ao *Open Project*), onde este determinado filtro buscará por uma *Tag* específica contida no arquivo .JSON e a manipulará conforme desejado.

Porém, no cenário dos gráficos que representam *ranking*, o retorno do filtro realizado sobre o arquivo .JSON será uma lista de dados organizada no formato de chave e valor, conforme pode ser melhor entendido na visualização da Figura 23.

Figura 23 – Código utilizado para contabilizar os *rankings* de profissionais que mais registraram e resolveram *bugs* no projeto.

```
  // Bugs opened by member
  const bugOpenedByMember = toArray(_.countBy(bugs.map(bug =>
bug._links.author), 'title')).sort((a, b) => b.value - a.value).slice(0, 3)

  // Bugs resolved by member
  const bugResolvedByMember = toArray(_.countBy(bugs.filter(bug =>
bug._links.status.title === 'Resolved').map(bug => bug._links.assignee),
'title')).sort((a, b) => b.value - a.value).slice(0, 3)
```

Fonte: Produzido pelo autor.

O fato de ter-se uma lista simplificada já contendo o nome e o valor associado aos colaboradores que mais abriram e resolveram defeitos só é possível, pois o filtro realizado para esta consulta utiliza recurso da linguagem de programação que visa otimizar ao máximo a busca pelos dados ideais. Desta forma, tal filtragem além de procurar as *Tags* desejadas, contabiliza as incidências das mesmas e organiza todos estes dados em um *ranking*. Tudo isso, permite que o código *front-end* utilize diretamente a lista obtida e a exiba em um gráfico no qual destrincha-se os valores de cada *index* da variável, conforme exibido na Figura 24.

Figura 24 – Código utilizado para exibição dos gráficos de *ranking*.

```
{/* Bugs abertos por membros */}
        <CCol xs={6}>
          <CCard className={"mb-3"}>
            <CCardBody>
              <CChart
                type="bar"
                data={{
                  labels: bugOpenedByMember.map(b => b.name),
                  datasets: [
                    {
                      label: "Top 3 Bugs abertos por membro",
                      backgroundColor: "#90be6d",
                      borderColor: "#90be6d",
                      data: bugOpenedByMember.map(b => b.value)
                    }
                  ],
                }}
              />
            </CCardBody>
          </CCard>
        </CCol>
```

Fonte: Produzido pelo autor.

4.3.2.2.2 GRÁFICOS DE DADOS DISTINTOS ASSOCIADOS

As métricas mencionadas nesta categoria específica de gráficos, são obtidas a partir da análise associativa entre os dados oriundos da ferramenta *Open Project* e os dados registrados no *TestLink*, seguindo uma combinação das lógicas arquiteturais de codificação apresentadas nas seções anteriores.

Para obtenção de dados do *Open Project* visando a consolidação das referidas métricas, efetuou-se um filtro sobre o arquivo *data.json* onde este determinado filtro buscou ativamente por uma *Tag* específica contida no arquivo .JSON e a manipulou conforme desejado. Sendo assim, o número total de *bugs* abertos para cada área do sistema e a identificação de seus respectivos *status* dentro do *worflow* do projeto, foram facilmente identificados e salvos em uma lista de dados organizada no formato de chave e valor, conforme pode ser melhor entendido e visualizado na Figura 25.

Figura 25 – Código utilizado para contabilizar as áreas do sistema que mais registraram *bugs* no projeto.

```
 // Bugs opened by area
const bugOpenedByArea = toArray(_.countBy(bugs.filter(bug => ['New', 'On
hold', 'Reopen'].includes(bug._links.status.title)).map(bug =>
bug._links.parent), 'title')).filter(f => f.name !== 'null').sort((a, b) =>
b.value - a.value).slice(0, 5)

const testsByOpenedArea = GetSuiteCasesOnTestLink(bugOpenedByArea,
'bugOpenedByArea') // get name to used on extract on database
```

Fonte: Produzido pelo autor

A presença de uma lista simplificada contendo o nome das áreas do sistema e os respectivos valores dos *bugs* associados, de acordo com o ponto de vista analisado, faz-se útil como pré-requisito para a obtenção de dados junto à ferramenta *TestLink*. Possibilitando assim, medir e exibir informações específicas, como por exemplo: o número de Casos de Teste existentes para cada área do sistema. Área esta que se encontra registrada na variável presente na lista proveniente da medição feita junto ao *Open Project*.

Para este cenário descrito, cria-se no código da Plataforma **AMP** um método típico para mapeamento de quais áreas do sistema faz-se necessário coletar o número de documentos de testes existentes. Tal método mencionado e chamado de *GetSuiteCasesOnTestLink*, interpreta a lista de dados fornecida e baseada nas informações coletadas da ferramenta *Open Project*, para auxiliar no direcionamento da criação de filtros com foco na extração dos dados que irão consolidar determinadas métricas e que por obséquio darão maior amplitude na análise, principalmente na análise de qualidade do projeto observado.

Sendo assim, em posse das áreas a serem filtradas para coleta das medições necessárias, se executa uma *query* de consulta SQL via conexão direta estabelecida com o banco de dados MySQL da ferramenta *TestLink*. A consulta criada visa obter a contabilização dos números de Casos de Testes registrados para cada área do sistema através da manipulação recursiva da tabela *nodes_hierarchy*, fazendo com que o *script* identifique e acesse as suítes de testes relacionadas e consequentemente contabilize o número de casos de teste registrados.

O resultado da requisição feita ao banco de dados é salvo em um arquivo .JSON, e nomeado de maneira a referenciar o escopo da métrica que solicitou a consulta dos dados. Por fim, o método *GetSuiteCasesOnTestLink* retorna como resultado o arquivo .JSON proveniente da consulta, e este arquivo será associado a uma dada variável que venha a facilitar o manuseio das informações coletadas. A Figura 26, expõe a codificação criada para a manipulação de dados do *TestLink* através do fluxo de trabalho dos métodos descritos.

Figura 26 – Código utilizado para o método *GetSuiteCasesOnTestLink* e consequentemente para obtenção de dados junto ao *TestLink*.

```
const GetSuiteCasesOnTestLink = (bugByArea, file) => {
 // use this to get names to find on database with file testlink/indexjs the other project
  const namesToFind = bugByArea.map(d => {
    let [, ...last] = d.name.split(" ")
    return last.join(' ')
  })
  console.log({ namesToFind, file })
  return require(`../public/${file}.testlink.json`)
}
 (async (bugArea, nameFile) => {
     let allData = []
     connection.connect();
     const runner = async function (data, acc = []) {
         let result = await new Promise((resolve, reject) => {
             connection.query(`SELECT * from nodes_hierarchy where parent_id in
(${data.map(d => d.id)}) `, function (error, results, fields) {
                 if (error) reject(error)
                 if (results.length > 0)
                     resolve(runner(results, [...acc, ...results]))
                 resolve(acc)
             });
         })
         return result
     }
     for (let bug of bugArea) {
         const initialData = await new Promise((resolve, reject) => {
             connection.query(`SELECT * from nodes_hierarchy where name like "%${bug}%"`,
function (error, results, fields) {
                 if (error) reject(error);
                 resolve(results)
             });
         });
         const dataArea = await runner(initialData)
         allData.push({ area: bug, data: dataArea.filter(f => f.node_type_id == 3).length
});
     }
     fs.writeFile(`${nameFile}.testlink.json`, JSON.stringify(allData), function (err) {
         if (err) return console.log(err);
         console.log('bugByArea.testlink.json');
     });
     connection.end();
 })
```

Fonte: Produzido pelo autor.

Para criar a visualização dos gráficos de dados distintos associados, o código *front-end* da Plataforma **AMP** utiliza de maneira simplificada a lista de informações obtidas junto ao *Open Project* e o arquivo .JSON proveniente do *TestLink*, que porventura contém o valor numérico de todos os casos de teste registrados para cada área pesquisada. O fato de utilizar-se uma estrutura de gráficos proveniente do framework *ReactJS* simplifica a criação de tais diagramas, sendo apenas necessário especificar o *dataset* a ser consumido conforme observa-se na Figura 27.

Figura 27 – Código utilizado para exibição dos gráficos de dados distintos associados.

```
          <CChart
               type="bar"
               data={{
                 labels: bugsByArea.map(b => '.'),
                 datasets: [
                   {
                     label: "Top 5 Bugs por área",
                     backgroundColor: "#fb5607",
                     borderColor: "#fb5607",
                     data: bugsByArea.map(b => b.value),
                   },
                   {
                     label: "Casos de Testes",
                     backgroundColor: "#E6B325",
                     borderColor: "#E6B325",
                     data: testsByArea.map(b => b.data),
                   }
                 ],
               }}
             />
        <CListGroup>
               {
                 bugsByArea.map((data, index) => (
                   <CListGroupItem key={index} className="d-flex justify-
content-between align-items-center">
                     {data.name}
                     <CBadge color="dark" shape="rounded-pill">
                       Bugs: {data.value} | Testes: {testsByArea[index]?.data}
                     </CBadge>
                   </CListGroupItem>
                 ))
               }
             </CListGroup>
```

Fonte: Produzido pelo autor.

5. RESULTADOS

Este capítulo expõe os resultados obtidos com a aplicação da pesquisa proposta. Primeiramente, é apresentada uma visão geral do ambiente de experimentação, com foco na demonstração do projeto ágil escolhido para medição. Em seguida é descrito o *layout* adotado no *dashboard* evidenciado pelo sistema, no qual expõe-se detalhadamente as métricas produzidas e gerenciadas pela plataforma desenvolvida. Logo após, são abordadas as experiências obtidas com base na implantação e disponibilização da Plataforma **AMP**, levando consequentemente a uma seção exclusiva para exposição das melhorias identificadas no ambiente de experimentação e que podem ser implantadas para a melhoria do processo ágil de desenvolvimento de *software* analisado.

5.1. Projeto Piloto

A versão inicial da plataforma desenvolvida foi validada em um dos projetos do Laboratório Assert. É importante salientar, que os projetos ágeis de *software* executados no contexto do Laboratório Assert usam o *framework scrum* para realização do gerenciamento de projetos. Desta forma, seguindo as boas práticas propostas são realizadas *sprints* de desenvolvimento quinzenais que ocorrem em paralelo com *sprints* de testes também quinzenais. Ou seja, durante a primeira semana da *sprint* de desenvolvimento a equipe de testes mantém-se focada nas atividades de planejamento dos *scripts* de testes, e ao término da *sprint* de desenvolvimento a equipe de qualidade de *software* dá início as validações necessárias junto ao artefato entregue. Este é um ciclo contínuo, iterativo e incremental que pode ser melhor visualizado e compreendido no Apêndice E.

O Laboratório Assert conta com times de desenvolvimento que variam de 4 a 12 desenvolvedores por iniciativa, dependendo da complexidade. Além, de possuir uma Célula de QA *cross project* que conta com aproximadamente 4 testadores, responsáveis pelas atividades de controle e garantia da qualidade de *software* em todos os projetos de desenvolvimento em curso na instituição, atuando de forma coordenada com os times de desenvolvimento para que as demandas por testes sejam atendidas dentro dos prazos estabelecidos e solicitados.

O Assert usa de maneira difundida de métricas de qualidade e produtividade que são principalmente empregadas através da consolidação de *reports* de *sprints*, que são compartilhados entres os membros dos projetos e por vezes com os clientes demandantes da solução. Porém, tais métricas sempre necessitaram ser consolidadas manualmente através de consultas diretas nas ferramentas *OpenProject* e *TestLink*, ou extraindo os dados desejados de tal ferramental mencionado e realizando-se as análises cruzadas em planilhas personalizadas. Sendo assim, é correto observar que o gestor de projetos no Assert necessita coletar e analisar dados provenientes de fontes diferentes para que assim possa realizar de forma mais objetiva suas atividades de monitoramento e controle do projeto ou de uma *sprint* específica. Além do mais, a uniformidade de acesso às métricas não lhe é garantida, pois cada projeto adota um modelo de análise dos dados, implicando desta maneira

admitir que algumas métricas empregadas em um dado projeto de desenvolvimento não é de conhecimento dos demais projetos em execução dentro da mesma instituição.

Posto isto, é possível notar que apesar destas ferramentas de gestão de projetos e de testes empregadas no Laboratório Assert serem amplamente utilizadas pelos projetos, as mesmas não compartilham dados e informações, pois não existe uma tecnologia nativa ou alguma pesquisa na literatura que estabeleça um elo de comunicação entre o *Open Project* e o *TestLink* para a efetiva troca de informações.

Para validação da plataforma, foi realizado um alinhamento com os Gestores atuantes no Laboratório Assert, visando delimitar um escopo de projetos a serem medidos pela plataforma. Tal alinhamento resultou no entendimento em utilizar-se um projeto específico já finalizado, pelos seguintes motivos:

- A utilização de um projeto ativo poderia acarretar problemas relacionados aos seus termos de privacidade e concessão em vigor;
- O alto volume de dados e as constantes mudanças de um projeto em execução poderiam atrapalhar o desenvolvimento da Plataforma **AMP**;
- A maioria dos projetos em execução não seguiam as boas práticas ágeis quanto ao uso das ferramentas de gestão *Open Project* e *TestLink*. Diferentemente do que foi observado em alguns dos projetos já finalizados, que inclusive foram reconhecidos pelo laboratório como *cases* de sucesso quanto ao uso das boas práticas ágeis.

Foi assim definida e formalizada a utilização de um projeto, aqui denominado de projeto *Delta*, tendo em vista questões de confidencialidade requeridos por projetos executados no âmbito do Polo de Inovação – IFPB. Tal Projeto selecionado foi desenvolvido em parceria com uma grande multinacional do setor elétrico, tendo por objetivo o desenvolvimento de um sistema ***web*** para a integração de três estações de produção distintas, que permitiria o acompanhamento e monitoramento em tempo real da produção através de um ***dashboard***. O processo de desenvolvimento levou aproximadamente 12 meses, sendo concluído em meados de dezembro de 2020, seguindo à risca os princípios da metodologia ágil *scrum* e aplicando também tais princípios no uso das ferramentas de gestão de projetos *Open Project* e *TestLink*, prezando sempre pela organização dos pacotes de trabalho registrados nas mesmas.

5.2. Dashboard de Métricas

Como fruto da implantação da solução proposta pela pesquisa, associada ao projeto piloto imerso no contexto do Laboratório Assert, foi possível analisar o funcionamento integrado dos *dashboards* de métricas disponibilizadas pela Plataforma **AMP** e assim averiguar o quão torna-se mais fácil a obtenção de acesso aos dados numéricos desejados e a possibilidade de lhes atribuir metas específicas. Tal constatação, visa sobretudo possibilitar que os gerentes de projeto atuantes

no laboratório possam tomar ações embasadas pelo máximo de informações que lhe são disponibilizadas, levando em conta os dados registrados tanto no *Open Project* quanto no *TestLink*, ambas as ferramentas utilizadas em ambiente real de produção do projeto piloto observado.

Sendo assim, em sua primeira versão, a plataforma disponibiliza para acompanhamento e análise as seguintes métricas de qualidade e produtividade referentes ao projeto *Delta*: Quantidade de *Bugs* Registrados no Projeto; *Bugs* Registrados por Semana; Quantidade de Planos de Teste Registrados no Projeto; Total Testes Executados que Foram Aprovados no Projeto; Total Testes Executados que Foram Reprovados no Projeto; Média de Tempo Gasto Executando Testes; Top 3 Colaboradores que mais Registraram *Bugs* no Projeto; Top 3 Colaboradores que mais Resolveram *Bugs* no Projeto; Top 5 Áreas (*Features*) do sistema com mais *Bugs* Registrado x Número de Testes Registrados Para a Referida Área (*Features*); Top 5 Áreas (*Features*) do sistema com mais *Bugs* em *Status* de Aberto x Número de Testes Registrados Para a Referida Área (*Features*) e Top 5 Áreas (*Features*) do sistema com mais *Bugs* em *Status* de Fechado x Número de Testes Registrados Para a Referida Área (*Features*).

Figura 28 –Tela de acompanhamento de métricas da plataforma desenvolvida.

Fonte: Produzido pelo autor.

Tais métricas descritas e apresentadas foram previamente escolhidas, conforme expostas na Tabela 7 da seção 4.1. Porém, a escolha destas métricas considerou o ambiente controlado do projeto Delta, visto que tal projeto se apresenta como um modelo de estruturação do *backlog* na ferramenta *Open Project* e de organização de documentos de testes na ferramenta *TestLink*. Posto isto, é possível reiterar que não é possível coletar algumas destas métricas mencionadas em alguns dos outros projetos pertencentes ao mesmo laboratório, visto que nestes casos pelo fato dos projetos não seguirem as boas práticas na utilização das ferramentas de gestão de projetos e de testes, a coleta das informações é severamente impactada.

O que leva a reafirmação da importância do projeto Delta como sendo um modelo de estruturação e de uso ferramental a ser seguido no Laboratório Assert. Modelo este, que comprovadamente facilita a gestão, possibilitando inclusive a medição automática dos indicadores que qualidade e produtividade do projeto.

Outro tópico importante a ser realçado, é que essa arquitetura e aparato tecnológico desenvolvido visando a integração da Plataforma de Métricas com os Repositórios de gestão de projetos utilizados pelo Laboratório Assert, só são efetivamente funcionais quando executados dentro da infraestrutura de servidores do próprio Polo de Inovação ou se executados via VPN (*Virtual Private Network*), com as devidas permissões concedidas pelo setor de infraestrutura e segurança da instituição. Esta cautela, têm por foco proteger os dados sensíveis e os acordos de privacidade firmados pelo Polo, evitando assim o vazamento de informações.

5.2.1. Detalhamento do Dashboard

Os *dashboards* disponibilizados pela Plataforma **AMP**, têm por objetivo primordial disponibilizar um painel contendo informações consolidadas e de fácil interpretação a respeito do monitoramento no tocante às medidas de qualidade e produtividade do projeto analisado. Sendo assim, os *dashboards* representam a camada direta de interação entre os usuários e as métricas pré-definidas na Tabela 7, servindo analogamente como um *report* integrado e automatizado de qualidade e produtividade para o contexto do Laboratório Assert.

Desta forma, buscando seguir o alinhamento lógico empregado na implementação das camadas arquiteturais, o *layout* adotado para visualização das métricas na camada de apresentação do sistema varia de acordo com a complexidade de análise dos dados apresentados, podendo o usuário deparar-se com duas grandes áreas no *dashboard* da Plataforma **AMP**, que são elas: a região dos *Widgets* Numéricos e a região dos *Widgets* Gráficos.

5.2.1.1 Widgets Numéricos

Contendo o título da métrica exibida e o seu respectivo valor conforme pode ser melhor observado na Figura 29.

Figura 29 – Métricas que são exibidas e estruturadas através de *widgets* numéricos.

Fonte: Produzido pelo autor.

Os *widgets* numéricos ocupam uma posição de destaque dentre os *dashboards* disponibilizados, ficando agrupados na posição superior da tela. Essa é uma ação estratégica, que visa melhorar a experiência do usuário fornecendo um painel com informações bem organizadas e facilmente localizáveis, além de dar destaque a tais métricas que são extremamente dinâmicas.

Estas métricas mencionadas são assim exibidas em *cards* no Painel da Plataforma **AMP**, pelo simples fato de suas medições retornarem um único valor numérico consolidado e que pode ser armazenado dentro de uma determinada constante, tornando-se assim muito mais fácil o manuseio e consequentemente a exibição destas informações. Por isso, as métricas que se adequam a este contexto são: Quantidade de *Bugs* Registrados no Projeto; *Bugs* Registrados por Semana; Quantidade de Planos de Teste Registrados no Projeto; Total Testes Executados que Foram Aprovados no Projeto; Total Testes Executados que Foram Reprovados no Projeto; Média de Tempo Gasto Executando Testes.

5.2.1.2 Widgets Gráficos

Como já mencionado neste trabalho, o termo *Widgets* Gráficos refere-se a áreas que têm por foco informar a evolução de uma determinada métrica em um dado período e que podem ser utilizadas de forma comparativa. Esta região do *dashboard* visa melhorar o entendimento da composição dos dados organizando-os em períodos específicos e utilizando gráficos de barras para tal propósito.

Também buscando seguir o alinhamento lógico empregado na implementação das camadas arquiteturais, o layout adotado para visualização dos *Widgets* Gráficos na camada de apresentação divide-se em dois subconjuntos, que são: Gráficos de *Rankings* e Gráficos de Dados Distintos Associados.

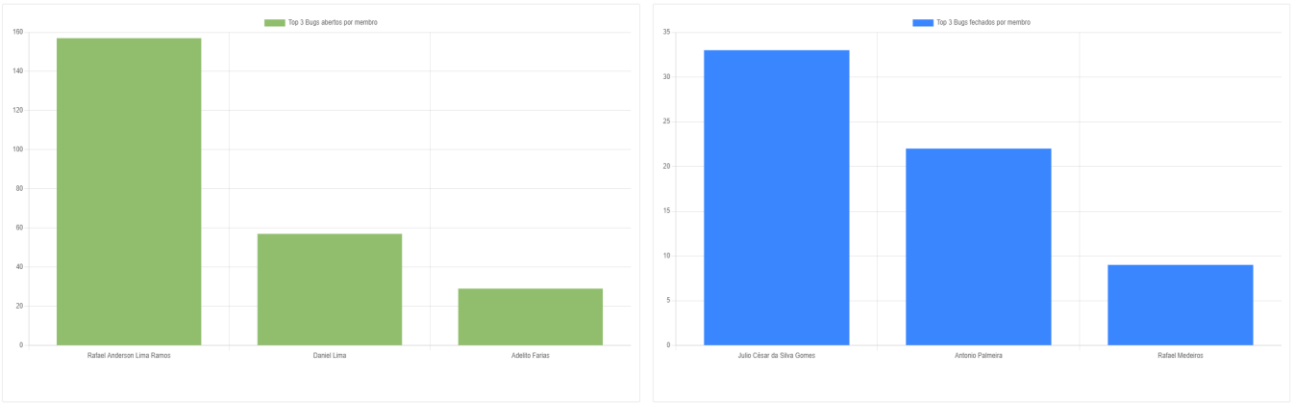
5.2.1.2.1 GRÁFICOS DE RANKINGS

Estes tipos de diagramas são responsáveis por permitir a visualização das métricas Top 3 Colaboradores que mais Registraram *Bugs* no Projeto e Top 3 Colaboradores que mais Resolveram *Bugs* no Projeto. Pois, o conjunto de informações obtidas com a medição e o dinamismo de tais dados possibilitam a utilização de gráficos de barras que melhor facilitam a interpretação das

informações referenciadas. A Figura 30 mostra claramente como tais gráficos são apresentados aos usuários quando os mesmos acessam o *dashboard* principal da Plataforma **AMP**.

Vale salientar que, estes gráficos detêm por maior foco a análise de produtividade da equipe e apenas são constituídos a partir da síntese de informações extraídas exclusivamente da ferramenta *Open Project.*

Figura 30 – Métricas que são exibidas e estruturadas através de gráficos de *rankings*.



Fonte: Produzido pelo autor.

5.2.1.2.2 GRÁFICOS DE DADOS DISTINTOS ASSOCIADOS

Estes tipos de diagramas permitem ao usuário visualizar através de gráficos de barras as métricas cujos dados são provenientes de diferentes repositórios (*Open Project* e *TestLink*), permitindo assim uma associação primária da informação que têm por consequência a ampliação na profundidade de análise dos índices de qualidade e produtividade observados. Bem como, estimula uma maior acessibilidade das informações visto que reúne em um mesmo local de visualização, dados de um mesmo projeto e que se encontram salvos em distintas ferramentas. Ferramentas estas, que não se comunicam nativamente devido às limitações tecnológicas e de pesquisas embasadoras que englobam tal escopo na literatura.

As métricas Top 5 Áreas (*Features*) do sistema com mais *Bugs* Registrado x Número de Testes Registrados Para a Referida Área (*Features*); Top 5 Áreas (*Features*) do sistema com mais *Bugs* em *Status* de Aberto x Número de Testes Registrados Para a Referida Área (*Features*) e Top 5 Áreas (*Features*) do sistema com mais *Bugs* em *Status* de Fechado x Número de Testes Registrados Para a Referida Área (*Features*), compões tal cenário apresentado e a Figura 31 mostra claramente como tais gráficos são exibidos aos usuários quando os mesmos acessam o *dashboard* principal da Plataforma **AMP**.

Figura 31 – Métricas que são exibidas e estruturadas através de gráficos da dados distintos associados.

Fonte: Produzido pelo autor.

5.3. Disponibilização e Implantação da Plataforma

O código fonte da Plataforma **AMP** encontra-se publicamente disponível para análise via repositório *GitHub*, através do seguinte endereço: https://github.com/faellima/AMP-REPO.

Porém, tal código disponibilizado publicamente possui a supressão de algumas informações tidas como sensíveis e que se divulgadas seriam consideradas uma ação de violação crítica à segurança do Polo de Inovação – IFPB, pois tal vazamento de informações poderia pôr em risco a infraestrutura dos projetos executados na instituição. Dentre tais dados classificadas como sensíveis e que precisam ser protegidas, têm-se por exemplo: credenciais de acesso a banco de dados, *token* de acesso a API, Id de Projetos de PD&I e mapeamento das tabelas contidas nas ferramentas de gestão de projetos.

Tendendo evitar o vazamento de informações, para o desenvolvimento da plataforma foi utilizado um repositório *GitHub* privado, o qual foi interconectado diretamente com a ferramenta *Netlify*.

O *Netlify* é o responsável pelo *deploy* automático da aplicação sempre que um novo *Pull Request* acontece na *branch* principal do repositório privado monitorado. Sendo assim, uma nova versão da Plataforma **AMP** é sempre disponibilizada de forma automática para seus usuários.

Posto isto, o *link* da primeira versão da Plataforma **AMP** que se encontra hospedado na ferramenta *Netlify*, foi disponibilizado aos Gerentes de Projeto do Laboratório Assert para que fosse possível ter-se acesso ao *software* desenvolvido e possibilitando também a experimentação da proposta ferramental disponibilizada. Bem como, dando vistas a instituição sobre a possibilidade de se vencer os desafios da unificação dos dados contidos em diferentes ferramentas de gestão de projeto que até então não eram interconectadas.

A Plataforma **AMP** foi desenvolvida e implantada tendo acesso direto a infraestrutura de dados do Laboratório Assert, realizando medições em um projeto real de PD&I que apesar de ter sido concluído a um determinado tempo, forneceu dados e *insights* necessários para o desenvolvimento de uma solução disruptiva que por muito tempo era desejada e aguardada pelo centro de pesquisa.

5.4. Análise da Plataforma no Projeto Piloto

Ao longo do avanço deste projeto de pesquisa que culminou no desenvolvimento da Plataforma **AMP**, foi possível observar e catalogar uma série de sugestões de melhorias a serem implantadas no ambiente de experimentação, tanto no tocante às questões de uso das ferramentas de gestão de projeto, quanto no que se refere ao processo adotado para desenvolvimento de projetos ágeis de *software* pelo Laboratório Assert.

Desta maneira, tais melhorias foram registradas visando postular-se como um pequeno guia de boas práticas tendo por objetivo otimizar e maximizar o uso da Plataforma **AMP**, bem como pretendendo dar maior estruturação e agilidade ao processo de desenvolvimento já em curso na instituição. Sendo assim, torna-se notória a obtenção de mais uma contribuição alcançada a partir do desenvolvimento e da implantação do sistema aqui implementado.

Posto isto, as melhorias elencadas e plausíveis de serem aplicadas ao ambiente de experimentação são:

- O *backlog* completo do projeto deve ser cadastrado na ferramenta *Open Project* e consequentemente obedecer às hierarquias estipuladas para os Pacotes de trabalho;
- Recomenda-se que o *backlog* cadastrado no *Open Project* seja quebrado em pacotes de trabalho do tipo *Epic*, que contenham *Features* associadas. Tais *Features* podem conter um ou mais *Acceptance Criteria* e um ou mais pacote do tipo *User Story*;
- Toda e qualquer nova *Task* cadastrada para o projeto deve estar associada a uma *User Story*, pois facilitará a aplicação de filtros, bem como o monitoramento e controle das informações;
- Para o registro de *Bugs* é recomendável criar-se um *Epic* específico para tal propósito que seja dividido em *Features* globais do sistema. Cada *Feature* conterá uma ou mais *User Story*, que representarão áreas específicas do sistema analisado e estas porventura possuirão os *Bugs* propriamente ditos associados (tal recomendação pode também ser utilizada para o registro das melhorias do sistema);
- O *Open Project* permite a criação de vários quadros *Kanban* com o objetivo de representar *Sprints*, estes quadros são chamados de versões. Porém, para o registro de *Bugs* e Melhorias é importante manter apenas um quadro de versão para cada, centralizando o registro das inconsistências em uma mesma versão;

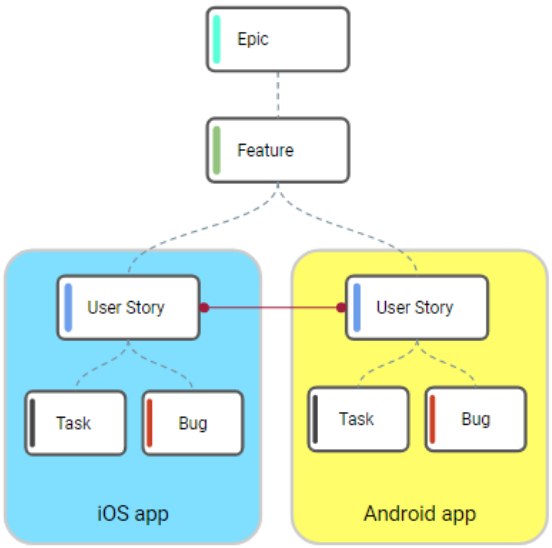
- É uma boa prática que toda *Task*, *Bug* e Melhoria criados no *Open Project* contenha seu registro de horas trabalhadas atualizado;
- No planejamento da *Sprint* ou antes de realizar a implementação de um *Bug* ou Melhoria, deve-se registrar a *Sprint* (versão) correspondente no formulário do pacote de trabalho;
- O nome da *User Story* deve representar seu propósito. Não se deve ter duas *User Story* com o mesmo nome.
- Não registar *Tasks* nos *Kanbans* de *Bugs* e Melhorias
- Recomenda-se que os nomes das suítes de testes criadas no *TestLink* sejam exatamente iguais aos nomes das *User Story* criadas no *Open Project* para agrupamento dos *Bugs*, pois isso implica diretamente no funcionamento da Plataforma **AMP** que porventura trabalha com a associação dos dados coletados;
- Todo Casos de Teste registrado no *TestLink* deve ter o seu campo Tempo Estimado de Execução preenchido;

É importante salientar que segundo Dias (1997), um *Epic* é uma *User Story* (US) que levaria mais de uma *Sprint* para ser concluída. Neste contexto, Épicos devem ser quebrados em *Features* e consequentemente em US's menores que possam ser concluídas dentro de uma *Sprint*. Já a *Feature*, ao contrário do *Epic*, é algo um pouco mais concreto daquilo que será desenvolvido, possuindo mais detalhes de uma interação ou ação do sistema.

Ainda segundo Dias (1997), uma US deve descrever da melhor forma possível uma dada funcionalidade que possui valor ao usuário. A escrita deve ser objetiva e de fácil compreensão pelo usuário, por tanto, faz-se necessário utilizar uma linguagem simples, clara e direta. Sendo exatamente esta as diretrizes e boas práticas que devem ser utilizadas nos registros de tais informações nas ferramentas de gestão de projetos utilizadas pelo Laboratório Assert (Polo de Inovação - IFPB).

Visando facilitar o entendimento da estratégia hierárquica mencionada e adotada como boa prática para o rastreamento das atividades de projeto ágil de desenvolvimento de *software*, expõe-se então a Figura 32 com o intuito de dar maior visibilidade sobre como um gerente de projetos conseguiria planejar e acompanhar o desenvolvimento de vários pacotes de trabalho de um mesmo projeto.

Figura 32 – Hierarquia dos pacotes de trabalho.



Fonte: Targetprocess (2022).

5.5. Outros Resultados da Pesquisa

Visando também agregar ainda mais valor para a Plataforma **AMP**, foi realizado o registro junto ao INPI[10] do código fonte da solução proposta. O Processo foi finalizado e expediu-se o certificado de registro de programa de computador de número **BR 51 2022 003304-6**.

É também importante realçar, que dentre outros resultados obtidos com o desenvolvimento desta pesquisa podem ser citados os seguintes itens:

- Apresentação deste referido projeto na XV Semana de Educação, Ciência, Cultura e Tecnologia do IFPB-Campus João Pessoa (**XV SECT/2020**), que ocorreu no dia 20 de novembro de 2020 às 14:30h;
- Submissão de um artigo, tendo por base o conteúdo exposto neste projeto de pesquisa, na **WTDSoft2021**;
- Aprovação no **Edital 01/2020 – Interconecta**, permitindo que parte do projeto fosse desenvolvido seguindo as normas e regulamentações do Programa Interconceta, instituídas pela Pró-Reitoria de Pesquisa, Inovação e Pós-Graduação. Atestando dessa forma, a relevância da pesquisa proposta, bem como aumentando as responsabilidades na geração de resultados de valor para a comunidade acadêmica.

[10] INPI: https://www.gov.br/inpi/pt-br

6. CONSIDERAÇÕES FINAIS

O estudo aqui apresentado, teve por foco propor uma plataforma de monitoramento de métricas de qualidade e produtividade em projetos de desenvolvimento ágeis de *software* a partir da integração dos dados obtidos junto às ferramentas de gestão de projetos e testes. Para alcançar tamanho objetivo, buscou-se responder às seguintes questões de pesquisa estabelecidas:

- QP1: Quais são as estratégias utilizadas para extração e disponibilização de métricas de qualidade e produtividade em equipes que desenvolvem projetos ágeis de *software*? (a qual foi respondida ao longo do **Capítulo 3**);
- QP2: Como extrair e integrar dados do *Open Project* e o *TestLink* para disponibilizar métricas de qualidade e produtividade de projetos ágeis de *software*? (a qual foi respondida ao longo do **Capítulo 4**).

A elucidação das questões de pesquisa levantadas, bem como a realização das etapas metodológicas estabelecidas anteriormente na seção 1.3, possibilitaram expor de maneira prática e sucinta a importância no uso de métricas para avaliar o maior número de informações sobre qualidade e produtividade referentes a um dado projeto de desenvolvimento ágil de *software*, considerando em seu escopo os recursos e o tempo disponível para execução do procedimento de medição. Medição esta, que quando realizada de forma automatizada e tendo como resultado uma série de informações bem estruturadas, possibilita aos Gerentes de Projeto uma oportunidade indescritível de melhorar e aperfeiçoar os seus processos, além de também auxiliar no planejamento e no controle dos projetos monitorados e não obstante possibilita o monitoramento da qualidade do produto final implementado.

Desta forma visando alcançar o impacto propositivo planejado, foi desenvolvido em parceria com o Laboratório Assert uma versão inicial de Plataforma para Gestão de Métricas de Qualidade e Produtividade aspirando o monitoramento de projetos ágeis de *software*, cuja arquitetura de tal sistema foi idealizada para possibilitar a integração com as diferentes ferramentas utilizadas na gestão de projetos e em uso no ecossistema de desenvolvimento analisado. Por suposto, esta iniciativa proporcionou que dados de um mesmo projeto até então registrados e monitorados em diferentes *softwares* de gestão, passassem a ser unificados a um nível de assegurar uma dada padronização da informação, bem como um fácil acesso às métricas pré-estipuladas.

Com os resultados alcançados durante o desenvolvimento e a implantação da Plataforma **AMP**, torna-se plausível considerar que de fato foi viabilizada uma estratégia tangível que pudesse ser implantada junto ao contexto do Laboratório Assert, tendo por foco o monitoramento das métricas de produto e também de processo. Vale salientar, que com a implementação de tal estratégia mencionada passou-se a ter a criação e a disponibilização das medições pré-definidas a partir da coleta automatizada de dados que até então encontravam-se estritamente registrados em duas ferramentas distintas e não interconectadas (*Open Project* e *TestLink*). Sinalizando, tanto para

a gestão do Laboratório e quanto para a comunidade acadêmica, que apesar das dificuldades tecnológicas encontradas para a integração dos dados contidos em tais ferramentas de gestão e do pouco número de pesquisas na literatura que sirvam de embasamento para este tema, é sim possível abandonar a criação de métricas a partir de métodos manuais de consultas e passar a disponibilizar *dashboards* integrados a diferentes repositórios, que concedam de maneira dinâmica metrificações de qualidade sobre o produto e o processo observado.

Sem dúvidas, existem oportunidades de melhoria junto à plataforma desenvolvida, visto que o produto de *software* entregue está em sua primeira versão funcional e utiliza um conjunto limitado de métricas definidas. No entanto, tais melhorias podem ser analisadas em trabalhos futuros ou podem ser melhor exploradas quando abordadas na fase de Implantação definitiva do sistema junto ao Laboratório Assert. Porém, independente das oportunidades de melhorias identificadas, é possível afirmar que os objetivos da Plataforma **AMP** foram devidamente alcançados e que a sua arquitetura com foco na obtenção dos dados para geração das métricas é extremamente viável e funcional no contexto de aplicabilidade analisado.

6.1. Contribuições da Pesquisa

Esta pesquisa teve como foco principal contribuir de maneira significativa para a ampliação dos horizontes que englobam os trabalhos acadêmicos na área de Gestão de Métricas, além de também fornecer aos profissionais que lidam com projetos ágeis de desenvolvimento de *software* uma maior independência e capacidade, ao dar-lhes um devido direcionamento na hora de promover uma eventual implantação de plataformas específicas e dedicadas para realização de medições com foco na qualidade e produtividade de projetos.

Sendo assim, como fruto direto da execução desta pesquisa é possível detalhar e elencar as seguintes contribuições:

- Revisão sistemática da literatura com foco na identificação das estratégias utilizadas para extração e disponibilização de métricas de qualidade e produtividade em projetos ágeis de *software*, identificando e analisando os principais trabalhos relacionados a esta pesquisa;
- Levantamento e mapeamento junto ao Laboratório Assert de todas as métricas de produtividade e qualidade em uso nos mais diversos projetos ágeis de desenvolvimento de *software*, e dentre tais métricas realização da escolha de um grupo pré-definido a ser comparado e ajustado com o estado da arte, para que na sequência tais informações pudessem ser coletadas e disponibilizadas de maneira automatizada através de *dashboards*;
- Estabelecimento de uma arquitetura de *software* com foco na extração dos dados advindos das ferramentas *Open Project* e *TestLink* (utilizadas para gestão de projetos no contexto do Laboratório Assert), e consequentemente na realização da correta

análise de tais dados para que se obtenha como artefato de saída métricas de qualidade e produtividade acerca do projeto analisado;

- Formalização do fluxo de trabalho focado na extração de dados e na consolidação das métricas desejadas, permitindo assim a evolução da proposta arquitetural demonstrada para que seja possível contemplar outras fontes de dados no mesmo arcabouço da plataforma gestora de métricas. Dentre estas outras fontes de dados que podem ser utilizadas para ampliação do cenário de informações, podem-se citar: *Jira*, *GitLab*, *GitHub*, dentre outras;
- Implementação de uma Plataforma de Métricas, tendo o ecossistema do Laboratório Assert como ambiente de experimentação e servindo desta maneira como prova de conceito para a arquitetura apresentada;
- Implantação da Plataforma **AMP**, utilizando um projeto real de PD&I desenvolvido no Laboratório Assert, tendo por intuito a validação da solução em ambiente simulado e possibilitando uma melhor análise dos resultados concedidos;
- Compartilhamento de boas práticas identificadas no uso de ferramentas de gestão de projetos, como o *Open Project* e o *TestLink*, visando auxiliar no melhor ordenamento do processo de desenvolvimento ágil de *software* e bem como facilitar a implantação de uma plataforma automatizada de extração e disponibilização de métricas.

Sem dúvidas, o sistema desenvolvido para extração e disponibilização de métricas de qualidade e produtividade em projetos ágeis de *software* sobressai-se como uma contribuição de maior destaque dentre as demais mencionadas, por se tratar de um resultado palpável e muito almejado pelo mercado e pela academia. Não existindo na literatura nenhuma estratégia consolidada e de arquitetura detalhada que melhor exemplifique tamanha extração de dados de um projeto de desenvolvimento, e muito menos não é possível encontrar nenhuma proposta que utilize ferramentas amplamente difundidas no mercado, como o *Open Project* e o *TestLink*, exercendo a função de bases de extração dos dados fomentadores das métricas almejadas.

Inclusive, a carência e a consistência de estudos que exploraram de maneira ampla o uso de um suporte ferramental como auxilio na gestão e no acompanhamento dos projetos através do agrupamento de informações produzidas em diversas ferramentas utilizadas no ciclo de desenvolvimento, é uma problemática que foi identificada durante a execução da Revisão Sistemática da Literatura (RSL) e que comprova tamanha lacuna de pesquisa. Sendo assim, este trabalho contribui à área da Engenharia de *Software*, oferecendo um modelo arquitetural e de fluxo de trabalho que possibilitam a integração entre os mais diversos *softwares* de gestão de projetos com foco na obtenção das métricas de qualidade e produtividade. Além de aplicar efetivamente as métricas proposta pela literatura em um contexto real de projeto ágil, reduzindo tanto as lacunas de

pesquisas observadas no que dizem respeito as aplicações de métricas, quanto as lacunas que dizem respeito a extração de dados advindos de diferentes fontes para consolidação de medições.

Vale ressaltar que, a Plataforma desenvolvida foi disponibilizada na íntegra ao Laboratório Assert, para que a instituição realizasse seu antigo anseio de conseguir unificar e facilitar o acesso às métricas dos projetos da instituição, bem como para que consiga evoluir e melhor otimizar a atual versão do *software*.

6.2. Trabalhos Futuros

Conforme abordado ao longo do capítulo 5 e como também mencionado neste próprio capítulo, é notório e de conhecimento transparente a existência de melhorias a serem implementadas junto a Plataforma **AMP** desenvolvida e entregue como resultado final deste projeto. Visto que, o sistema disponibilizado é na verdade uma implementação de referência que teve por foco consolidar uma prova de conceito suficientemente robusta para validação da proposta de pesquisa, levando em consideração as delimitações de escopo e tempo. Mesmo assim, é indiscutível o potencial de uso comercial e acadêmico apresentado pela Plataforma **AMP**, cujo tais potencialidades podem ser maximizadas mediante a implementação de melhorias a serem concretizadas em forma de trabalhos futuros.

Dentre tais melhorias, torna-se plausível citar a ampliação no número de projetos monitorados pelo sistema, possibilitando que o usuário em uma mesma instância da plataforma selecione o projeto desejado e consequentemente tenha acesso às métricas de qualidade e produtividade disponíveis. Vale frisar que este é um aperfeiçoamento plausível de ser implementado, uma vez que a arquitetura proposta nesta pesquisa conta com dois módulos de código aberto para gerenciamento de métricas (Gerenciador de Métricas de Produtividade e Gerenciador de Métricas de Qualidade), possibilitando assim que os desenvolvedores criem novas medições e as adicionem nos gerenciadores de métricas ou então utilizem as métricas já existente em tais módulos gestores, onde neste cenário não existe a necessidade de novas implementações, sendo apenas necessário fornecer os dados esperados para os cálculos da métrica escolhida.

Outra eventual melhoria a nível de sistema e que também pode ser implementada, é o aumento na quantidade de métricas que a plataforma é capaz de processar e disponibilizar aos seus usuários, conseguindo inclusive em uma proposta futura cobrir todas as métricas de qualidade e produtividade utilizadas pelo Laboratório Assert, fazendo com que tais informações sejam geradas de forma automatizada e possibilitando uma maior acessibilidade aos dados. Tal aumento na quantidade de métricas disponibilizadas pela Plataforma **AMP** pode se dar tanto pelo aumento no número de métricas implementadas no gerenciadores de métricas, conforme citado anteriormente, quanto pela a ampliação no número de fonte dados. No tocante a captação de novas fontes, é possível expandir a camada de acesso aos dados através da implementação novos conectores ou *plugins*, para que assim a plataforma consiga extrair informações de outras ferramentas utilizadas no ciclo de

desenvolvimento ágil de *software*, possibilitando inclusive a criação de novas métricas que podem ser anexadas a plataforma.

Com o intuito de ampliar a usabilidade do sistema pode-se existir a implementação de *dashboards* personalizados, nos quais os usuários consigam selecionar quais métricas desejam visualizar e consequentemente tenham autonomia para montar a forma mais propícia de enxergar as informações de um dado projeto. Também é possível a nível de infraestrutura implementar um serviço de atualização automática dos dados junto aos repositórios utilizados, que por meio de um *script* ou por meio de alguma outra estratégia tecnológica sejam disparadas requisições que solicitem a extração de novas informações, pois atualmente essa ação é feita de maneira manual uma vez que a Plataforma **AMP** não encontra-se hospedada na estrutura de servidores do Polo de Inovação – IFPB, mas apenas possui acesso restrito de consulta a determinadas base de dados e por um tempo de conexão estipulado.

Ainda no que diz respeito aos fatores de segurança, especificamente ao controle de acesso dos usuários a Plataforma **AMP**, neste primeiro momento não foi implementado um mecanismo responsável por gerenciar o acesso dos usuários. Ou seja, a contenção do acesso é realizada através da disponibilização do *link* da plataforma diretamente a um grupo de usuários pré-selecionados, não havendo a necessidade de autenticar-se no *software* para que se tenha acesso às métricas desejadas. Sendo assim, é perceptível que um módulo de gestão de usuários pode ser implementado para garantir um melhor controle de acesso e de permissões, aumentando consideravelmente os níveis de segurança da aplicação.

É importante pontuar, que a arquitetura de *software* proposta para a plataforma de gestão de métricas de produtividade e qualidade em projetos ágeis de desenvolvimento foi concebida com o propósito de ser expandida, permitindo assim que a camada de acesso aos dados seja maximizada através de uma estrutura de módulos de conectividade, a qual possibilite a adição de um número ainda maior de conectores ou *plugins* visando a integração com outras ferramentas utilizadas no ciclo de desenvolvimento de *software*, como por exemplo o *GitLab*, *GitHub*, *Jira*, dentre outras. Ampliando desta maneira, a extração de dados que hoje concentra-se nas ferramentas *Open Project* e *TestLink*. Porém, é importante salientar que esta é uma proposta de melhoria que demandará muito tempo e esforço em detrimento a amplitude do escopo.

Ainda no que diz respeito a otimização da arquitetura de *software* proposta, é também possível vislumbrar uma melhoria em potencial no módulo responsável por realizar a conexão direta com a base de dados da ferramenta *TestLink*, sendo possível cogitar a criação de uma nova API para tal propósito ou sugerir a realização de um *refactoring* na API nativa disponibilizada pelos desenvolvedores da ferramenta. Tal esforço é justificado pela necessidade em se manter uma padronização nas estratégias de acesso aos dados das ferramentas nas quais a Plataforma **AMP** realiza a obtenção das informações necessárias, pela redução na dependência de acesso direto a uma base de dados de uma dada ferramenta e pelo aumento significativo no reuso do código utilizado.

Por conseguinte, baseando-se nos tópicos expostos acima, fica evidente que este trabalho pode ser ampliado das mais diversas formas. Entretanto, além das propostas de melhorias de cunho técnico, também se vislumbra como um aprimoramento futuro a realização de uma análise qualitativa da Plataforma **AMP**, tendo por foco analisar a usabilidade do sistema através de testes de experimentação executados com auxílio de usuários reais constituintes do público alvo.

REFERÊNCIAS BIBLIOGRÁFICAS

AKTUNC, Ozgur. *Entropy Metrics for Agile Development Processes. In: 2012 IEEE 23rd International Symposium on Software Reliability Engineering Workshops*. IEEE, 2012. p. 7-8.

ALVES, Kedson. Gestão de Indicadores/Métricas com *Grafana*. *In:* . [s.n.], 2018. Disponível em: https://www.profissionaisti.com.br/2018/10/gestao-de-indicadoresmetricas-com-grafana/. Acesso em: 05 dez. 2019.

AMBLER, Tim; CLOUD, Nicholas. *JavaScript Frameworks for Modern Web Dev*. Apress, 2015.

BECK K. Programação Extrema Explicada. São Paulo: Ed. Bookman, 1999.

BIOLCHINI, J. *et al*. *Systematic Review in Software Engineering*. Rio de Janeiro: UFRJ, 2005.

BOA MORTE, A. *et al*. Uma Análise Sobre o Uso de DLTs no Tratamento de Dados Pessoais: Aderência aos Princípios e Direitos elencados na. W *Blockchain* SBRC - Simpósio Brasileiro de Redes de Computadores e Sistemas Distribuídos, p. 14, 2020.

BOERMAN, Martin P. *et al*. *Measuring and Monitoring Agile Development Status*. 2015 *Ieee/acm 6Th International Workshop On Emerging Trends In Software Metrics*. [s.l.], v. 1, n. 1, p. 54-62, 2015.

BRAYNNER, Thiago *et al*. Métricas de *Software*. 2008. 15 f. Monografia (Especialização) - Curso de Especialização em Qualidade de *Software*, Centro de Informática, Universidade Federal de Pernambuco, Recife, 2008.

BUILDER, Project. Gestão ágil de projetos: entenda melhor esse conceito e como fazer. 2018. Disponível em: https://www.projectbuilder.com.br/blog/gestao-agil-de-projetos-entenda-melhor-esse-conceito-e-como-fazer/. Acesso em: 05 dez. 2019.

BUKHARI, Zubaidah; YAHAYA, Jamaiah; DERAMAN, Aziz. *In:* 2018 *A Conceptual Framework for Metrics Selection: SMeS. International Journal on Advanced Science, Engineering and Information Technology*. 2018. v. 8, p. 2294-2300.

CAETANO, Cristiano. Automação e Gerenciamento de Testes: Aumentando a Produtividade com as Principais Soluções *Open Source* e Gratuitas. 1. ed. [s.l.]: [s.n.], 2007.

CARDOZO, Luiz Fernando. Um guia para aplicação de métricas ágeis de gerenciamento de projetos para organizações de desenvolvimento de *software*. 2020. 212 f. TCC (Graduação) - Curso de Bacharelado em Sistemas de Informação, Universidade Federal de Santa Catarina, Florianópolis, 2020. Disponível em: https://repositorio.ufsc.br/bitstream/handle/123456789/218148/TCC%20%20Luiz%20Fernando%20Cardozo.pdf?sequence=1&isAllowed=y. Acesso em: 20 abr. 2021.

CHO, J.; HURLEY, P.; XU, S. *Metrics and measurement of trustworthy systems*. *In: Proc. IEEE 35th Military Communications Conference (MILCOM 2016).* Baltimore, USA, 2016, p. 1237-1242.

CHORAS, Michał *et al*. *Measuring and Improving Agile Processes in a Small-Size Software Development Company. In: IEEE Access 8.* IEEE, 2020. p. 78452-78466.

COHN, M. *Agile Estimating and Planning*. Massachusetts: Pearson Education, 2006.

CORDEIRO, Marco Aurélio. Métricas de *Software*. Curitiba, 2000. Disponível em: http://www.pr.gov.br/celepar/batebyte/edições/2000/bb101/métricass.html. Acesso em: 12 maio 2021.

CORE.UI. Manual. Disponível em: https://coreui.io/react/docs/getting-started/introduction/. Acesso em: 07 set. 2022.

COSTA, A. *et al*. Utilizando métricas para dimensionar um *software*. 2013. Disponível em: https://docplayer.com.br/11078269-Utilizando-metricas-para-dimensionar-um-software.html. Acesso em: 20 maio 2021.

DAHAB, Sarah *et al*. *A Novel Formal Approach to Automatically Suggest Metrics in Software Measurement Plans*. *In: 13th International Conference on Evaluation of Novel Approaches to Software Engineering*. 2018. p. 283-290.

DALFOVO, Oscar. Sistemas de informação. Blumenau: Acadêmica, 2004.

DEUTER, Andreas; ENGELS, Gregor. (2014). *Measuring the Software Size of Sliced V-Model Projects*. *In: International Conference on Software Process and Product Measurement*. IWSM-Mensura, 2014.

DIAS, M. S.; ANDRADE, R.; TRAVASSOS, G. H. Diretrizes para Redução de Complexidade de *Software* Orientado a Objetos. *In:* VIII Conferência Internacional de Tecnologia de *Software*. Curitiba, Brasil. 1997.

DINGSOYR, T. *et al*. *A decade of agile methodologies: Towards explaining agile software development*. Journal of Systems and Software, v. 85, n. 6, p. 1213-1221, jun. 2012.

DOWNEY, S. *Scrum Metrics for Hyperproductive Teams: How They Fly like Fighter Aircraft*. *In: 46th Hawaii International Conference on System Sciences*, Wailea, 2013, p. 4870-4878. Disponível em: https://ieeexplore.ieee.org/document/6480431. Acesso em: 09 set. 2019.

DYBÅ, T.; DINGSØYR, T. *Empirical studies of agile software development: a systematic review*. *Information and Software Technology*, v. 50, n. 9, p. 833-859, 2008.

EASTERBROOK, S., SINGER, J., STOREY, M-A., DAMIAN, D. *Selecting Empirical Methods for Software Engineering Research, In: Shull F et al (ed) Guide to advanced empirical software engineering, Chapter* 11. Springer-Verlag, Lon-don, 2008.

FERNANDES, Aguinaldo Aragon. Gerência de *software* através de métricas: garantindo a qualidade do projeto, processo e produto. São Paulo: Atlas, 1995.

FUCK, Mônica Andrea. Estudo e aplicação das métricas da qualidade do processo de desenvolvimento de aplicações em banco de dados. 1995. 104 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 1995.

GUARIZZO, Karina. Métricas de *Software*. 2008. 48 f. TCC (Graduação) - Curso de Bacharelado em Ciência da Computação, Faculdade de Jaguariúna, Jaguariúna, 2008. Disponível em: https://docplayer.com.br/645247-Karina-guarizzo-metricas-de-software-jaguariuna.html. Acesso em: 20 abr. 2021

HEIMANN, D.; HENNESSEY, P.; TRIPATHI, A. *A Bipartite Empiricallt Oriented Metrics Process for Agile Sofware Development*. Software Quality Professional (SQP), v. 9, 2007.

HEPTAGON. *FDD – Feature Driven Development*. 2015.

HUMPHREY, W.S. *Managing the Software Process*. Reading, MA, USA: Addison-Wesley, 1989.

KITCHENHAM, B. A. *et al*. *Guidelines for performing Systematic Literature Reviews in Software Engineering*: Version 2.3. EBSE Technical Report. Keele, UK: Keele University, 2007.

KOZIK, R.; CHORÁS, M.; PUCHALSKI, D.; RENK, R. *Q-Rapids framework for advanced data analysis to improve rapid software development. In: J Ambient Intell Hum Comput*, 2018.

KTATA, Oualid; LÉVESQUE, Ghislain. *Designing and implementing a measurement program for Scrum teams: what do agile developers really need and want?* p. 101-107, 2010.

KUNZ, M.; DUMKE, R. R.; SCHMIETENDORF, A. *How to Measure Agile Software Development*. Berlin: Springer, 2008.

KUNZ, Martin; DUMKE, Reiner R.; ZENKER, Niko. *Software Metrics for Agile Software Development. In: 19th Australian Conference On Software Engineering (aswec 2008)*. IEEE, 2008. p.1-6. Disponível em: https://ieeexplore.ieee.org/abstract/document/4483261. Acesso em: 14 abr. 2021.

LIECHTI, Olivier; PASQUIER, Jacques; REIS, Rodney. *Beyond Dashboards: On the Many Facets of Metrics and Feedback in Agile Organizations*. 2017.

LIKERT, R. *A Technique for the Measurement of Attitudes*. Archives of Psychology, p. 1-55, 1932.

LINKE, Rudiger; LUNDBERG, Jonas; LOWE Welf. *Comparing Software Metrics Tools*. *In: ISSTA'08*. Seattle, Washington, USA, 2008.

MARTÍNEZ-FERNÁNDEZ, Silverio *et al*. *Continuously Assessing and Improving Software Quality With Software Analytics Tools: A Case Study*. *In: IEEE Access 7*. IEEE, 2019. p. 68219-68239.

MEDEIROS, Juliana Dantas Ribeiro Viana de *et al*. Engenharia de requisitos em projetos ágeis: uma revisão sistemática da literatura. Revista Principia - Divulgação Científica e Tecnológica do IFPB, [S.l.], n. 28, p. 11-24, dez. 2015. ISSN 2447-9187. Disponível em: https://periodicos.ifpb.edu.br/index.php/principia/article/view/459. Acesso em: 21 out. 2020.

MEDEIROS JUNIOR, Ivan. Polo de Inovação - Apresentação. 2022. Disponível em: https://www.ifpb.edu.br/polodeinovacao/institucional/apresentacao. Acesso em: 07 set. 2022.

MEDING, W. *Effective monitoring of progress of agile software development teams in modern software companies: An industrial case study*. *In: Proceedings of the 27th International Workshop on Software Measurement and 12th International Conference on Software Process and Product Measurement*. New York, NY, USA: ACM, 2017. (IWSM Mensura ’17), p. 23–32.

NODE.JS. Manual. Disponível em: http://nodemanual.org/latest/. Acesso em: 06 jun. 2022.

OLIVEIRA A. G. G. de. Construção de Aplicações Distribuídas Utilizando-se de APIs REST. Universidade do Estado do Rio Grande do Norte (UERN), 2018. Disponível em: <https://di.uern.br/tccs2019/html/ltr/PDF/014006456.pdf>. Acesso em: 28 mar. 2021.

OLIVEIRA, Edson *et al*. *‘Software project managers’ perceptions of productivity factors: findings from a qualitative study*. *In: 10th ACM/IEEE Int. Symp. on Empirical Software Engineering and Measurement*. New York, NY, USA: Association for Computing Machinery, 2016. (ESEM '16), p. 1-6.

PADMINI, K. V. J.; BANDARA, H. M. N. Dilum; PERERA, I. *Use of software metrics in agile software development process. In: Moratuwa Engineering Research Conference (MERCon)*. [s.n.], 2015. p. 312-317.

PARDELINHA, L. Manual da ferramenta *Open Project.* 2013. Disponível em: http://engenheironocanteiro.com.br/Openproj-cronograma. Acesso em: 07 set. 2022.

PEGORARO, Raquel Aparecida. Métricas de avaliação para abordagens ágeis em projetos de *software*, 2014. 166 f. Tese (Doutorado) - Curso de Pós-Graduação em Engenharia de Produção, Departamento de Engenharia, Universidade Federal do Rio Grande do Sul, Porto Alegre, 2014.

PRESSMAN, Roger S. Engenharia de *Software*. São Paulo: Makron Books, 1995.

PRESSMAN, Roger S. Engenharia de *Software* - Uma Abordagem Profissional. 8ª ed.: AMGH, Porto Alegre, 2016.

PULUCENO, Thiago Vieira. Estudo de Caso Sobre uma *AP REST* Utilizando a Abordagem de Programação Orientada e Eventos com a Plataforma NODE.JS. 2012. 74 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas da Informação) – Departamento de Informática e Estatística, Universidade Federal de Santa Catarina, Florianópolis.

RAMOS, Rafael Anderson de Lima. Um *Framework* Para Adaptação de Processos de *Software* Guiado por TDD em Aderência a Norma ISO/IEC e IEEE 12207. 2015. 96 f. Monografia

(Especialização) - Curso de Bacharelado em Ciência da Computação, Centro Universitário de João Pessoa - Unipê, João Pessoa, 2015.

REIS, Christian Robbotom. Caracterização de um Processo de *Software* para Projetos de *Software* Livre. 2003. 247 f. Dissertação (Mestrado) - Curso de Ciência da Computação e Matemática Computacional, Universidade de São Paulo, São Paulo, 2003.

RESMINI, Tiago Nunes. Sistema para Avaliação de Indicadores voltado para Times de Desenvolvimento de Produtos de *Software*. 2016. 53 f. TCC (Graduação) - Curso de Curso de Engenharia de Controle e Automação, Universidade Federal de Santa Catarina, Florianópolis, 2016.

ROACH, S. *White Collar Productivity: A Glimmer of Hope? Special Economic Study*, Morgan Stanley, New York, set. 1988.

SANTOS, Fernando de Souza. Uma proposta de coleta e visualização de métricas de custo, tamanho e esforço, em projetos de *software* ágeis, com apoio de ferramenta *Data Warehousing*. 2013. 94f. TCC (Graduação) - Curso de Bacharelado em Engenharia de Software, Universidade de Brasília – UnB, Brasília, 2013.

SANTOS JÚNIOR, José Renato dos. Aplicação *Web* Para Controle de Produção de Uma Fábrica de Pré-Moldados 2015. 10 f. TCC (Graduação) - Curso de Bacharelado em Engenharia de *Software*, Faculdade de Tecnologia do Piauí– Fatepi, Teresina, 2015.

SATO, Danilo Toshiaki. Uso eficaz de métricas em métodos ágeis de desenvolvimento de *software*. 2007. 155 f. Dissertação (Mestrado) - Curso de Ciência da Computação, Universidade de São Paulo, São Paulo, 2007.

SCHWABER, Ken. *The Enterprise and Scrum.* 1st. ed. Washington: Microsoft Press, 2007.

SEIBT, Patrícia Regina Ramos da Silva. Ferramenta Para Cálculo de Métricas em *Softwares* Orientados a Objetos. 2001. 99 f. TCC (Graduação) - Curso de Bacharelado em Ciência da Computação, Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau, 2001.

SHARMA, V. S; KAULGUG, V. *Agile Workbench: Tying People, Process, and Tools in Distributed Agile Delivery*. *In: 2016 IEEE 11th International Conference on Global Software Engineering (ICGSE)*. IEEE, 2016. p. 69-73.

SHIU, Alicia. *Thinking about building your own analytics? Don't*. 2020. Disponível em: https://amplitude.com/blog/2016/04/21/thinking-building-analyticsdont/. Acesso em: 25 nov. 2020.

SINGH, Gurdev; SINGH, Dilbag; SINGH, Vkram. *A Study of Software Metrics*. *In: IJCEM International Journal of Computational Engineering & Management*. [s.l.], v. 11, 2011.

SOMMERVILE, Ian. Engenharia de *Software*. 10º ed.: Pearson, São Paulo, 2019.

TARGETPROCESS. *Platform-specific User Stories under common Features*. Disponível em: https://www.targetprocess.com/guide/entity-types/data-mapping/platform-specific-user-stories-common-features/. Acesso em: 07 set. 2022.

THUNG, Ferdian *et al*. *Network structure of social coding in GitHub*. *In: Proceedings of the Euromicro Conference on Software Maintenance and Reengineering, CSMR*. IEEE, 2013. p. 323-326.

TIMÓTEO, A. L.; ÁLVARO, A.; DE ALMEIDA, E. S.; DE LEMOS MEIRA, S. R. *Software metrics: A survey. In:* . [S.l.: s.n.], 2008.

TURNER, C. Reid *et al*. *A conceptual basis for feature engineering*. Journal of Systems and Software, v. 49, n. 1, p. 3-15, 1999.

VERSIONONE. *14th Annual state of Agile Development Survey*. Atlanta: VersionOne.com, 2020.

VIPUL, A.; SONPATKI, P. *ReactJS by Example-Building Modern Web Applications with React*. [s.l.]: Packt Publishing Ltd, 2016.

YONG, Chu Shao. Tecnologia de Informação. Revista de Administração de Empresas, São Paulo, n. 32, p. 78-87, jun/mar. 1992.

APÊNDICES

APÊNDICE A – PROTOCOLO PARA REVISÃO SISTEMÁTICA.

Introdução:

As informações sobre o andamento das atividades em um projeto de desenvolvimento de *software*, juntamente com a quantidade e tipos de *bugs* encontrados são essenciais para uma gestão eficiente (PRESSMAN, 2011). Desta forma, é primordial para alcançar tal nível de eficiência em gestão, a identificação de plataformas que realizam a distribuição e visualização das métricas de qualidade e produtividade em projetos ágeis de *software*, para que assim tais dados possam de forma integrada gerar mais valor para a gestão de projetos.

Posto isto, buscando minimizar erros sistemáticos e definir de maneira clara o procedimento a ser adotado no levantamento do estado da arte, decidiu-se realizar uma Revisão Sistemática da Literatura (RSL) sobre o tema de pesquisa acerca das estratégias utilizadas para extração e disponibilização de métricas de qualidade e produtividade em projetos ágeis de *software*. Sendo assim, para chegar a tais evidências será utilizado um protocolo (BOLCHINI, 2005) como parte da RSL (KITCHENHAM, 2007).

Com o protocolo e seus critérios de seleção e exclusão de artigos a ideia é identificar o estado da arte para desenvolver uma plataforma *web* de extração e visualização personalizada das métricas de qualidade e produtividade em equipes de *software*. A plataforma pretende auxiliar na gestão e na tomada de decisão de projetos ágeis.

O protocolo apresentado a seguir, segue o que determina os "Procedimentos de Execução de Revisões Sistemáticas em Engenharia de *Software*", de Kitchenham et al. (2007). Esse documento pretende guiar a produção de uma RSL e identificar o estado da arte em métricas de qualidade e produtividade de projetos ágeis de *software*.

OBJETIVOS:

Objetivo Geral: Realizar revisão bibliográfica sobre as estratégias utilizadas para extração e disponibilização de métricas de qualidade e produtividade em projetos ágeis de *software*.

Objetivos Específicos:

- Identificar as plataformas utilizadas para a extração e apresentação personalizada de métricas de qualidade e produtividade em equipes de *software*;
- Identificar estratégias a serem utilizadas para integração com ferramentas de gestão

de projetos, controle de versão, testes, dentre outras, visando a coleta automática de dados sobre produtividade e qualidade;

- Identificar o impacto, na tomada de decisão, criado pela presença ou ausência de uma fácil visualização/acesso à métricas de qualidade e produtividade em equipes de *software*.

ETAPAS (Fluxograma):

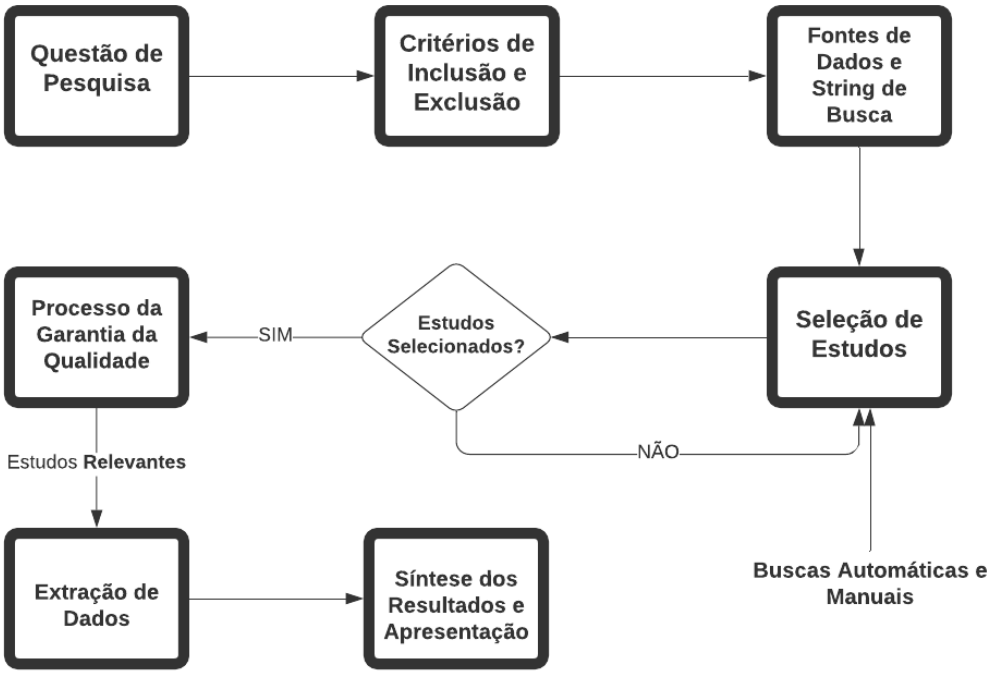


Figura 1: Elaborado pelo autor com base em Medeiro et al. (2015).

QUESTÕES DE PESQUISA:

A principal questão para este trabalho é:

Questão de Pesquisa Geral: Quais são as estratégias utilizadas para extração e disponibilização de métricas de qualidade e produtividade em equipes que desenvolvem projetos ágeis de *software*?

Para complementá-la, temos as perguntas a seguir:

QP. E1: Quais plataformas estão sendo utilizadas para facilitar a extração e visualização de métricas de qualidade e produtividade durante a execução das atividades realizadas em projetos ágeis de *software*?

QP. E2: Quais estratégias estão sendo utilizadas para integração com ferramentas de gestão de projetos, controle de versão, testes, dentre outras, visando a coleta automática de dados sobre produtividade e qualidade?

QP. E3: Quais estratégias não automatizadas de extração e visualização das métricas de qualidade e produtividade, são aplicadas durante a execução das atividades em projetos ágeis de *software*?

QP. E4: Quais os impactos (positivos/negativos) são observados na gestão e na tomada de decisão, a partir do uso das plataformas de extração e visualização das métricas de qualidade e produtividade?

O protocolo de pesquisa inclui várias atividades, que segundo Kitchenham e Charters (2007) podem ser divididas em três fases: planejamento, condução e relatório. Tais fases podem ser melhor observadas na Figura 2.

Figura 2: Atividades do processo de execução da Revisão Sistemática da Literatura cobertas pelo protocolo de pesquisa, com base em Kitchenham e Charters (2007)

Até o momento, este protocolo abordou apenas tópicos referente às atividades de planejamento da pesquisa. Sendo assim, os próximos tópicos abordarão as atividades referentes às atividades contidas nas fases de condução e relatório.

Intervenção: encontrar e aplicar o estado da arte sobre estratégias utilizadas para extração e visualização das métricas de produtividade e qualidade em projetos ágeis de *software*.

Controle:

 # Engenharia de requisitos em projetos ágeis: uma revisão sistemática da literatura - Juliana Dantas Ribeiro Viana de Medeiros, Daniela C. P. Alves, Alexandre Marcos Lins de Vasconcelos, Carla Taciana Lima Lourenço Silva Schuenemann, Eduardo Wanderley;

 # ApplicationsSoftware Development Productivity Tools and Metrics - Anne Smith Duncan**.**

População: Projetos da engenharia de *software* ágil que coletem métricas para orientar sua gestão.

Resultados: Uma visão abrangente das formas implementadas para extração e distribuição de métricas na gestão de projetos ágeis de desenvolvimento de *software*, apontando limitações e oportunidades na área.

Aplicação: Desenvolvimento, registro de *software* e publicação de artigo.

SELEÇÃO DE FONTES:

As fontes deverão estar disponíveis via *web*, preferencialmente em bases de dados científicas da área. Sendo assim, foram escolhidas as fontes de busca automáticas e manuais com acesso institucional permitida para o Instituto Federal da Paraíba - IFPB via Portal de Periódicos da CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior).

Poderão ser selecionados também trabalhos disponíveis em outros meios, como por exemplo congressos relevantes, desde que atendam aos requisitos da Revisão Sistemática.

PALAVRAS-CHAVES:

Qualidade de *Software*; Métricas; Desenvolvimento de *Software*; Produtividade no Desenvolvimento de *Software*; Gerenciamento de Projetos de *Software*; Projetos Ágeis de *Software*;

Painel de Gerenciamento em Projetos Ágeis; Tomada de Decisão; Extração e Visualização de Métricas; Integração de Plataformas; Coleta Automática de Dados.

LISTAGEM DE FONTES:

· Biblioteca Digital do IEEE (http://ieeexplore.ieee.org/Xplore/)

· Biblioteca Digital da ACM (http://portal.acm.org/)

· SCOPUS (http://www.scopus.com/home.url)

· SpringerLink (http://springerlink.com)

Para complementar a seleção dos artigos, também serão selecionados artigos de maneira manual em conferências específicas das áreas, referente aos últimos 3 anos:

- ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (https://ieeexplore.ieee.org/xpl/conhome/1001929/all-proceedings)
- International Conference on Information Resources Management (https://aisel.aisnet.org/conf-irm/).

TIPO DOS ARTIGOS:

Serão considerados estudos de acordo com os critérios de inclusão e exclusão listados abaixo.

IDIOMA(S) DOS ARTIGOS:

Inglês.

CRITÉRIOS DE INCLUSÃO E EXCLUSÃO DOS TRABALHOS:

Critérios de inclusão:

IC1. Estudos aplicados na indústria ou academia, que tratem sobre uso de métricas de qualidade e produtividade na gestão de projetos ágeis *software*;

IC2. Pesquisas qualitativas ou quantitativas;

IC3. Estudos primários.

Critérios de exclusão:

EC1. Escrito em um idioma que não seja o inglês;

EC2. Estudos duplicados ou repetidos;

EC3. Estudos que não tratem de métricas de qualidade de *software* ou produtividade de *software*;

EC4. Estudos realizados a mais de 12 anos;

EC5. Estudos incompletos, rascunhos, slides ou resumos;

EC6. Estudos secundários, terciários e meta-análises;

EC7. Estudos que não abordam pelo menos uma métrica de qualidade e produtividade em projetos ágeis de *software*;

EC8. Artigos que não estejam disponíveis gratuitamente para *download* nos ambientes institucionais do IFPB;

EC9. Estudos teóricos que não apresentem nenhum tipo de validação.

PROCESSO DE SELEÇÃO DOS ESTUDOS:

Serão construídas *strings* com as palavras-chave e seus sinônimos. As *strings* serão submetidas às *engines* de busca. Após a leitura do resumo e título, será aplicado os critérios de inclusão e exclusão.

Em uma segunda fase, os critérios serão aplicados na leitura da introdução e da conclusão dos estudos resultantes da fase anterior. Quando necessário, a leitura completa do estudo será efetuada.

O trabalho será selecionado se confirmada a sua relevância pela dupla de revisores. Se houver dúvida da relevância o Professor Orientador será consultado e servirá de mediador para o conflito.

Para o processo de seleção, será utilizada a ferramenta Parsif.al. Esta ferramenta oferece gratuitamente uma aplicação online em https://parsif.al/ e ainda tem a possibilidade de ser descarregada para dispositivos móveis. Inclusive aceita a importação de referências em vários formatos e permite ainda a condução de revisão simultânea por mais de um investigador. A aplicação tem um tutorial disponível online que fornece orientações a novos utilizadores.

CRITÉRIOS DE GARANTIA DA QUALIDADE DOS ESTUDOS:

Após aplicar os critérios de inclusão e exclusão, será avaliada a qualidade dos estudos primários através de um questionário adaptado de Medeiro et al. (2015).

Os artigos serão analisados pelos pesquisadores que terão acesso ao questionário de qualidade dos estudos, apresentado na Tabela 1. Sendo assim, para cada questão aplicada será utilizada a escala de três pontos de Likert (1932) visando obter uma pontuação do artefato validado.

- 0 (Equivalente a Não no Parsif.al): Não existe nada no artigo que atenda ao critério avaliado;
- 0.5 (Equivalente a Parcialmente no Parsif.al): O artigo não deixa claro se atende ou não ao critério;
- 1 (Equivalente a Sim no Parsif.al): O artigo atende ao critério avaliado.

Tabela 1 - Questionário de qualidade aplicado aos estudos selecionados.

Critério de Qualidade
1. Trata-se de um artigo de pesquisa?
2. Os objetivos da pesquisa são claros?
3. O contexto do estudo encontra-se claro?
4. O método da pesquisa foi justificado?
5. A amostragem foi representativa?
6. Os dados foram coletados de maneira a atender adequadamente a questão de pesquisa? (Está claro como os dados foram coletados?)
7. Os dados coletados justificam os resultados obtidos na pesquisa? (O processo de análise dos dados foi descrito?)
8. Vieses foram considerados?
9. Os resultados foram claramente descritos?
10. O estudo possui/apresenta uma discussão a respeito da contribuição para prática ou para literatura?

Fonte: Produzido pelo autor.

A partir do somatório das notas de todos os critérios, os artigos foram classificados em quatro faixas de qualidade de acordo com a pontuação obtida, conforme apresentado na Tabela 2. Os artigos

com somatório classificado nas faixas Alta e Muito Alta foram encaminhados para extração, os demais foram descartados nesta etapa.

Tabela 2 - Pontuação da faixa de qualidade.

Baixa	Média	Alta	Muito Alta
$0 \leq N \leq 2,5$	$3 \leq N \leq 5,5$	$6 \leq N \leq 8,5$	$9 \leq N \leq 10$

Fonte: Produzido pelo autor.

ESTRATÉGIA DE EXTRAÇÃO DE INFORMAÇÃO:

Após definidos os trabalhos a serem incluídos, estes serão lidos na íntegra. No processo de extração de dados dos artigos, as seguintes informações deverão ser catalogadas: dados de publicação (referência), contexto (tipo de estudo, métodos de pesquisa, análise dos dados, tamanho da amostra, presença ou ausência de plataformas para extração e visualização de métricas) e evidências (trechos de texto) objetivando responder as questões de pesquisa, conforme sugerido por Cruzes e Dybå (2011), citado por Medeiros et al. (2015).

Os pesquisadores se organizarão em duas equipes, dividindo-se igualmente o número de artigos selecionados. Desta forma, cada equipe realizará a extração nos artigos utilizando a ferramenta MAXQDA. E, por fim, será feito o processo de junção de todas as extrações obtidas na ferramenta.

Vale ressaltar, que os pesquisadores devem adotar uma técnica de destacar em cores diferentes as questões de pesquisa durante a leitura e obrigatoriamente revisar o conteúdo extraído pelo outro pesquisador, procurando identificar questões de pesquisa que não estão sendo abordadas. Além de realizar uma atenta revisão em segundo plano deve-se fazer um levantamento da quantidade de empresas e quantidade de pessoas envolvidas nos estudos.

SÍNTESE E APRESENTAÇÃO DOS DADOS:

A síntese e análise dos dados ocorrerá de maneira paralela utilizando uma abordagem qualitativa, buscando implementar uma análise temática dos dados conforme recomendado por Cruzes e Dybå (2011), onde serão categorizados os trabalhos de acordo com as plataformas de extração e visualização de métricas utilizadas, limitações e benefícios das plataformas utilizadas e estratégias utilizadas para integração de ferramentas visando a coleta automática de dados sobre produtividade e qualidade.

Em seguida, deverá ser feito um relacionamento dos códigos com as áreas temáticas identificadas (MEDEIROS et al., 2015). Vale ressaltar, que também se têm a intenção de através de uma abordagem quantitativa, analisar a frequência de ocorrência dos códigos mapeados

Nesta etapa, de síntese e apresentação dos resultados, utilizaremos a ferramenta *MAXQDA* para a geração de relatórios, onde será possível identificar a relação entre os estudos e as questões de pesquisa.

Atributos a serem extraídos dos artigos incluídos: Das questões de pesquisa apresentadas, foram extraídos constructos exibidos na Tabela 3 visando nortear os pesquisadores na identificação e codificação das principais características encontradas durante a realização deste estudo.

Tabela 3 - Constructos das questões de pesquisa.

QP	**Constructos (Códigos)**
QP 1	1. Estratégia para extração de métrica (EEM). 2. Estratégia para disponibilização de métrica (EDM). 3. Metodologia ágil empregada no projeto (MAE).
QP. E1	1. Plataforma de extração de métrica de produtividade (PEMP). 2. Plataforma de visualização de métrica de produtividade (PVMP). 3. Plataforma de extração e visualização de métrica de produtividade (PEVMP). 4. Plataforma de extração de métrica de qualidade (PEMQ). 5. Plataforma de visualização de métrica de qualidade (PVMQ). 6. Plataforma de extração e visualização de métrica de qualidade (PEVMQ). 7. Plataforma de extração de métrica de produtividade e qualidade (PEMPQ). 8. Plataforma de visualização de métrica de produtividade e qualidade (PVMPQ). 9. Plataforma de extração e visualização de métrica de produtividade e qualidade (PEVMPQ).
QP. E2	1. Estratégias de integração (EI). 2. Coleta automática de dados (CAD). 3. *Software*/ferramenta integrado (S/FI).

QP. E3	*1.* Dados obtidos sem plataformas específicas (DSP).
QP. E4	1. Impactos positivos (IP). 2. Impactos negativos (IN).

Fonte: Produzido pelo autor.

STRING DE BUSCA:

((“software” OR “information system engineering” OR “information system development”) AND (“agile” OR “agility”) AND ("project management" OR "decision making" OR “software quality” OR “quality of software” OR “dashboard” OR "management panel" OR "software development productivity" OR "software team productivity" OR "developer productivity") AND ("metrics" OR “indicators”))

REFERÊNCIAS:

BIOLCHINI, J. et al. Systematic Review in Software Engineering, UFRJ: Rio de Janeiro, 2005.

CRUZES, D. S.; DYBÅ, T. Recommended steps for thematic synthesis in software engineering. In: INTERNATIONAL SYMPOSIUM ON EMPIRICAL SOFTWARE ENGINEERING AND MEASUREMENT, 5, 2011, Banff, Canada. Proceedings… Banff: IEEE, 2011.

KITCHENHAM, B. A.; CHARTERS, S. Guidelines for performing Systematic Literature Reviews in Software Engineering: Version 2.3. EBSE Technical Report. Keele, UK: Keele University, 2007.

MEDEIROS, Juliana Dantas Ribeiro Viana de et al. Engenharia de requisitos em projetos ágeis: uma revisão sistemática da literatura. Revista Principia - Divulgação Científica e Tecnológica do IFPB, [S.l.], n. 28, p. 11-24, dez. 2015. ISSN 2447-9187. Disponível em: <https://periodicos.ifpb.edu.br/index.php/principia/article/view/459>. Acesso em: 21 Out. 2020.

NERI DE SOUZA, F.; COSTA, A. P.; MOREIRA, A. WebQDA: Software de Apoio à Análise Qualitativa. In.: ROCHA, A. (Ed.). Conferência Ibérica de Sistemas e Tecnologias de Informação, 5 CISTI’2010. Santiago de Compostela, Espanha: Universidade de Santiago de Compostela, 2010.

OUZZANI, Mourad et al.. Rayyan — a web and mobile app for systematic reviews. Systematic Reviews (2016) 5:210, DOI: 10.1186/s13643-016-0384-4.

PRESSMAN, R. S. Engenharia de Software. 7.ed. Porto Alegre:AMGH, 2011, 779p.

OLSEN, K; POSTHUAM, m.;ULRICH, S. - Brazilian Software Testing Qualifications Board, Foundation Level Syllabus, 201

APÊNDICE B – CERTIFICADO DE REGISTRO DE PROGRAMA DE COMUTADOR CONCEDIDO PELO INPI

REPÚBLICA FEDERATIVA DO BRASIL
MINISTÉRIO DA ECONOMIA
INSTITUTO NACIONAL DA PROPRIEDADE INDUSTRIAL
DIRETORIA DE PATENTES, PROGRAMAS DE COMPUTADOR E TOPOGRAFIAS DE CIRCUITOS INTEGRADOS

Certificado de Registro de Programa de Computador

Processo Nº: **BR512022003304-6**

O Instituto Nacional da Propriedade Industrial expede o presente certificado de registro de programa de computador, válido por 50 anos a partir de 1° de janeiro subsequente à data de 30/08/2021, em conformidade com o §2°, art. 2° da Lei 9.609, de 19 de Fevereiro de 1998.

Título: Assert Metrics Panel: uma plataforma para acompanhamento de métricas de qualidade e produtividade em projetos ágeis de software

Data de publicação: 30/08/2021

Data de criação: 23/03/2020

Titular(es): INSTITUTO FEDERAL DE EDUCACAO, CIENCIA E TECNOLOGIA DA PARAIBA

Autor(es): JULIANA DANTAS RIBEIRO VIANA DE MEDEIROS; RAFAEL ANDERSON DE LIMA RAMOS; HEREMITA BRASILEIRO LIRA

Linguagem: JAVA SCRIPT; NODEJS

Campo de aplicação: AD-01; AD-06

Tipo de programa: AP-01

Algoritmo hash: SHA-512

Resumo digital hash:
9102e960222ef455299ba11693d2c47d0d8e445dce4ed656d93e318145b381c8e1ea6213062028edbbef380812fcc96d7cdb82d0feb5e9be3b21ab631c6ae466

Expedido em: 06/12/2022

Aprovado por:
Carlos Alexandre Fernandes Silva
Chefe da DIPTO

APÊNDICE C – MODELO DE ARQUIVO .JSON CONTENDO DADOS OBTIDOS DA FERRAMENTA OPEN PROJECT


```
{
        "description": {
          "format": "markdown",
          "raw": "O campo $Nome não está validando a quantidade máxima de
caracteres, que é de 50.",
          "html": "<p>O campo $Nome não está validando a quantidade máxima de
caracteres, que é de 50.</p>"
        },
        "spentTime": "PT1H30M",
        "laborCosts": "0.00 EUR",
        "materialCosts": "0.00 EUR",
        "overallCosts": "0.00 EUR",
        "_type": "WorkPackage",
        "id": 3195,
        "lockVersion": 11,
        "subject": "Campo $Nome não valida qtd de caracteres máximo",
        "startDate": "2019-07-31",
        "dueDate": null,
        "estimatedTime": null,
        "percentageDone": 0,
        "createdAt": "2019-07-31T13:04:31Z",
        "updatedAt": "2019-09-05T16:03:00Z",
        "remainingTime": null,
        "customField9": "59",
        "customField2": {
          "format": "markdown",
          "raw": null,
          "html": ""
        },
        "customField13": {
          "format": "markdown",
          "raw": null,
          "html": ""
        },
        "customField3": {
          "format": "markdown",
          "raw": null,
          "html": ""
        },
          "logTime": {
            "href": "/work_packages/3195/time_entries/new",
```

```
      "type": "text/html",
      "title": "Log time on Campo $Nome não valida qtd de caracteres máximo"
    },
    "move": {
      "href": "/work_packages/3195/move/new",
      "type": "text/html",
      "title": "Move Campo $Nome não valida qtd de caracteres máximo"
    },
    "copy": {
      "href": "/work_packages/3195/copy",
      "title": "Copy Campo $Nome não valida qtd de caracteres máximo"
    },
    "pdf": {
      "href": "/work_packages/3195.pdf",
      "type": "application/pdf",
      "title": "Export as PDF"
    },
    "atom": {
      "href": "/work_packages/3195.atom",
      "type": "application/rss+xml",
      "title": "Atom feed"
    },
    "availableRelationCandidates": {
      "href": "/api/v3/work_packages/3195/available_relation_candidates",
      "title": "Potential work packages to relate to"
    },
    "customFields": {
      "href": "/projects/software/settings/custom_fields",
      "type": "text/html",
      "title": "Custom fields"
    },
    "configureForm": {
      "href": "/types/7/edit?tab=form_configuration",
      "type": "text/html",
      "title": "Configure form"
    },
    "addWatcher": {
      "href": "/api/v3/work_packages/3195/watchers",
      "method": "post",
      "payload": {
        "user": {
          "href": "/api/v3/users/{user_id}"
        }
      },
      "templated": true
    },
    "removeWatcher": {
      "href": "/api/v3/work_packages/3195/watchers/{user_id}",
      "method": "delete",
```

```
            "templated": true
          },
          "addRelation": {
            "href": "/api/v3/work_packages/3195/relations",
            "method": "post",
            "title": "Add relation"
          },
          "addChild": {
            "href": "/api/v3/projects/software/work_packages",
            "method": "post",
            "title": "Add child of Campo $Nome não valida qtd de caracteres
máximo"
          },
          "changeParent": {
            "href": "/api/v3/work_packages/3195",
            "method": "patch",
            "title": "Change parent of Campo $Nome não valida qtd de caracteres
máximo"
          },
          "addComment": {
            "href": "/api/v3/work_packages/3195/activities",
            "method": "post",
            "title": "Add comment"
          },
          "previewMarkup": {
            "href": "/api/v3/render/markdown?context=/api/v3/work_packages/3195",
            "method": "post"
          },
          "timeEntries": {
            "href": "/work_packages/3195/time_entries",
            "type": "text/html",
            "title": "Time entries"
          },
          "category": {
            "href": null
          },
          "type": {
            "href": "/api/v3/types/7",
            "title": "Bug"
          },
          "priority": {
            "href": "/api/v3/priorities/8",
            "title": "Normal"
          },
          "project": {
            "href": "/api/v3/projects/11",
            "title": "XXXXXXXXXX"
          },
          "status": {
```

```
            "href": "/api/v3/statuses/13",
            "title": "Closed"
          },
          "author": {
            "href": "/api/v3/users/99",
            "title": "Daniel Lima"
          },
          "responsible": {
            "href": "/api/v3/users/99",
            "title": "Daniel Lima"
          },
          "assignee": {
            "href": "/api/v3/users/57",
            "title": "Francisco Neto"
          },
          "version": {
            "href": "/api/v3/versions/64",
            "title": "Sprint 3 - [22/07 a 02/08]"
          },
          "logCosts": {
            "href": "/work_packages/3195/cost_entries/new",
            "type": "text/html",
            "title": "Log costs on Campo $Nome não valida qtd de caracteres
máximo"
          },
          "showCosts": {
            "href": "/work_packages/3195/cost_entries",
            "type": "text/html",
            "title": "Show cost entries"
          },
          "costObject": {
            "href": null
          },
          "costsByType": {
            "href": "/api/v3/work_packages/3195/summarized_costs_by_type"
          },
          "customField7": {
            "title": "Test Plan Execution",
            "href": "/api/v3/custom_options/12"
          },
          "customField20": {
            "title": "0.2.1",
            "href": "/api/v3/custom_options/46"
          },
          "customField21": {
            "title": "0.3.1",
            "href": "/api/v3/custom_options/121"
          },
          "customField6": {
```

```
          "title": "Requirement",
          "href": "/api/v3/custom_options/7"
        },
        "customField30": {
          "href": null,
          "title": null
        },
        "customField5": {
          "title": "Medium",
          "href": "/api/v3/custom_options/4"
        },
        "customField8": {
          "title": "Requirement Issue",
          "href": "/api/v3/custom_options/13"
        },
        "customField19": {
          "title": "[Desktop] Registro de Usuários",
          "href": "/api/v3/custom_options/95"
        },
        "watch": {
          "href": "/api/v3/work_packages/3195/watchers",
          "method": "post",
          "payload": {
            "user": {
              "href": "/api/v3/users/61"
            }
          }
        },
        "ancestors": [
          {
            "href": "/api/v3/work_packages/5008",
            "title": "BUGS"
          },
          {
            "href": "/api/v3/work_packages/3363",
            "title": "BUGS - Desktop"
          },
          {
            "href": "/api/v3/work_packages/3064",
            "title": "[Desktop] Registro de Usuários"
          }
        ],
        "parent": {
          "href": "/api/v3/work_packages/3064",
          "title": "[Desktop] Registro de Usuários"
        },
        "customActions": []
      }
    },
```

APÊNDICE D – MODELO DE ARQUIVO .JSON CONTENDO DADOS OBTIDOS DA FERRAMENTA TESTLINK


```
[{"area":"Planejamento de
Produção","data":57},{"area":"Turnos","data":36},{"area":"Sistemas
Gerais","data":12},{"area":"Cadastro das Máquinas","data":31},{"area":"Importação
de Ordens de Produção do ERP","data":16}]

[{"display":"Quantidade de planos de testes","value":8},{"display":"Quantidade de
execuções que passaram","value":362},{"display":"Quantidade de execuções que não
passaram","value":152},{"display":"Media de tempo de execuxão em
minutos","value":"2.48"}]
```

APÊNDICE E – PROCESSO DE DESENVOLVIMENTO UTILIZADO NO LABORATÓRIO ASSERT